İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FİLO ATAMASI PROBLEMİ VE KARMAŞIK TAM SAYI PROGRAMLAMA İLE EN İYİLEME YÖNTEMLERİ

YÜKSEK LİSANS TEZİ

Doğan Akay

Anabilim Dalı : Uçak ve Uzay Mühendisliği

Programı : Disiplinler Arası Program

HAZİRAN 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

FİLO ATAMASI PROBLEMİ VE KARMAŞIK TAM SAYI PROGRAMLAMA İLE EN İYİLEME YÖNTEMLERİ

YÜKSEK LİSANS TEZİ
Doğan AKAY
(511071106)

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2009
Tezin Savunulduğu Tarih : 8 Haziran 2009

Tez Danışmanı : Prof. Dr. Metin Orhan KAYA (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. İbrahim Özkol (İTÜ)
Doç. Dr. Mustafa ÖZDEMİR (İTÜ)

HAZİRAN 2009

Sevgili anneme,

ÖNSÖZ

Üniversite hayatımın başından beri değerli bilgilerini esirgemeyen, bunun yanında abilik yapan sevgili hocam Prof. Dr. Metin Orhan Kaya'ya, dualarıyla hep yanımda olan sevgili anneme ve ablama, esprileriyle motivasyonuma yaptığı katkı sebebiyle babama, verdiği manevi destekle motivasyonumu hep en üst seviyede tutan sevgili Ezgi Özer'e teşekkürü bir borç bilirim.

Mayıs 2009

Doğan Akay

(Uzay Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

KKKM : Kullanılabilir Koltuk Kilometresi Masrafı
FAP : Filo Ataması Problemi
YGK : Yolcu Gelir Kilometresi
KKKG : Kullanılabilir Koltuk Kilometresi Masrafı
KKK : Kullanılabilir Koltuk Kilometresi
Eİ : En iyileme
KTP : Karmaşık Tamsayı programlama

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

FİLO ATAMASI PROBLEMİ VE KARMAŞIK TAM SAYI PROGRAMLAMA İLE EN İYİLEME YÖNTEMLERİ

ÖZET

Bu çalışmanın amacı, havayolu şirketlerinin en büyük maliyet kalemlerinden biri olan filo ataması konusunun incelenmesi ve filo ataması probleminin tüm detayları ile gösterilip, probleminin çözümü için mevcut olan tekniklerin bilgisayar kodları ile modellenmesidir.

Filo ataması problemi yöneylem araştırması literatüründe nispeten yeni bir konu olup, endüstriyel anlamda ciddi maliyet artışlarına sebep olan "doğru uçuşa-doğru uçağın atanması" sorununun çözümü için gereklidir.

Çalışmada öncelikle filo ataması problemi için önerilen iki model ve bu modellerin çözümünde kullanılan karmaşık tamsayı programlama ve karmaşık tamsayı programlama çözüm teknikleri açıklanmıştır. Teknikler için verilen örnekler el ile çözülüp, yazılan kodlar ile karşılaştırılmış, bu sayede kodun güvenilirliği teyit edilmiştir.

Filo ataması probleminin ayrıntılı olarak açıklanabilmesi için bir örnek model kullanılmıştır. Örnek model için filo ataması probleminin en başı olan maliyet analizinden, filo tiplerinin önceden belirlenmiş uçuşlara atanmasına kadar olan tüm konular ayrıntılı olarak incelenmiş, böylelikle ileride bu konu üzerinde çalışacaklar için önemli bir kaynak oluşturulmuştur.

Filo ataması probleminin modellenmesinde kullanılan karmaşık tamsayılı programlama tekniğinin çözümü için dört adet teknik kodlanmış ve sonuçlar karşılaştırılmıştır. Bu yöntemler sırasıyla yorucu numaralama, dal ve sınır algoritması, Gomory kesme düzlemi yöntemi ve genetik algoritmadır.

Çalışma aynı zamanda filo atamasından sonraki aşama olan uçak atama problemi için bir ön çalışma mahiyetindedir. Çünkü filo ataması probleminin çözüm kalitesi, uçak atama probleminin başarısı için önemli bir parametredir.

FLEET ASSIGNMENT PROBLEM AND MIXED INTEGER PROGRAMMING OPTIMIZATION TECHNIQUES

SUMMARY

The aim of this study is to examine fleet assignment problem which is one of the greatest cost factor for airline companies and show fleet assignment problem in detail and write and program the codes for the solution techniques of fleet assignment problem.

The fleet assignment problem is a relatively new subject in the operations research literature, and is about "right fleet for the right flight" motto, which causes great cost for the airline companies.

In this study the first aim is to explain the fleet assignment problem and the two different and major models from the literature. Second, the mathematical model, mixed integer programming, which is commonly used to model the fleet assignment is explained in detail. The techniques which are used for mixed integer programming are examined and coded and some developments are made upon them. The techniques are used in solution to some small problems and the results are compared to the results of the codes outcome. By this, the verification of the code is done.

To examine fleet assignment problem in detail, a sample model is used. For this sample model, fleet assignment problem is examined throughly, from the cost analysis, to the assignment of the fleet types, so this study is a great source for the subsequent researchers on this subject.

The mixed integer programming, which is used in modelling the fleet assignment problem, is solved with four solution techniques. These techniques are, in order, exhaustive enumeration, branch and bound, Gomory cuts, and genetic algorithms.

1. GİRİŞ

Koltuklar bir havayolu şirketinin ürünleridir, havayolu şirketleri koltukları pazarlar. Diğer her ürün gibi, fazla ürün satışları arttıracağı gibi, masrafı da arttırır. Her havayolu şirketi için çok fazla koltuk kapasitesine sahip olmak, işletme masrafını arttırır. Diğer bir yandan satılan koltuklar dayanıksız tüketim malzemeleridir, yani uçak kalkışa geçmeden önce satılamayan koltuklar zarar demektir. Bu nedenle ideal strateji "doğru" miktarda koltuğu, "doğru" fiyata müşterilere sağlamaktır. Bu stratejilerden ilki filo ataması problemi ile ilgiliyken, diğeri gelir yönetimi ile ilgilidir.

Filo ataması problemi (FAP) her biri farklı kapasitelere, malzeme ve yakıt gerekliliklerine, işletme masraflarına sahip uçakları önceden belirlenmiş uçuşlara atama problemidir. Atama problemi doğrudan havayolu şirketinin karlılığına etki eder: Gereken yolcu kapasitesinden daha küçük bir uçağı bir uçuşa atamak doğrudan kapasite yetersizliği sebebiyle müşteri kaybına yol açar, diğer taraftan gereken yolcu kapasitesinden daha büyük bir uçağı bir uçuşa atamak koltukların satılamamasına sebep olur ve koltukların satılamamasının yanı sıra daha büyük bir işletme masrafına sebep olur. Dolayısıyla FAP bir havayolu şirketinin planlama stratejisinin en önemli parçasıdır. Gün içinde planlanan çok sayıda uçuşa, uçakların bakım gereksinimlerine, ekip planlamasına, gelir yönetimine bağlılığından dolayı filo ataması problemini çözmek havayolu şirketleri için her zaman çok zor olmuştur. Bu sebeple FAP, Yöneylem Araştırması literatüründe fazlaca çalışılmıştır.

Filo ataması probleminin çözümü için geleneksel yöntemler filo ataması probleminin diğer havayolu karar mekanizmalarından izole şekilde çözülmesinin gerekliliğini savunur. Filo ataması probleminin diğer durumlar göz önüne alınarak çözülmesi işlem yükünü arttırdığı gibi olurlu (feasible) çözümün bulunmasını da zorlaştırır.

2. FİLO ATAMASI PROBLEMİNDE GENEL KAVRAM VE TANIMLAR

2.1 Filo Tipi

Aynı kokpit konfigürasyonuna, ekip yetenek gerekliliklerine, bakım gerekliliklerine ve kapasitesine sahip uçaklardır. Örnek olarak Airbus A321 tipi verilebilir.

2.2 Filo Ailesi

Aynı gerekliliklere sahip uçakların oluşturduğu kümedir. Aynı ekip bir ailenin tüm uçakları ile uçabilir. Örnek olarak Boeing 757/767 ailesi verilebilir. Bu ailede 757-200 ve 767-300 gibi kapasiteleri 186 ile 225 arasında değişen uçaklar vardır.

Çizelge 2.1 : A320 ve B757/767 filo aileleri.

Filo Ailesi	A320					B757/767				
Filo Tipi	A32A	A32B	A32C	A319	A321	757-200	757-300	C-32	767-200	767-400ER

2.3 Rota

Kalkış ve iniş arasında bir veya birden çok uçuş ayağını birleştiren belirli bir zamanda başlayan uçuş kümesine rota denir. Bir kalkış ve iniş çiftinde birden çok rota olabilir.

2.4 Direkt Uçuş

Birden fazla uçuş ayağını aynı uçakla uçmaya direkt uçuş denir. Direkt uçuşlar müşteriler tarafından tercih edilir zira uçak uzun süreli duraklamalar yapsa bile gidecekleri noktaya aynı uçak içinde varırlar.

2.5 Devir Süresi

Uçağın inişinden bir sonraki kalkışına kadar gerek duyduğu süredir. Bu süre içinde küçük denetlemeler, bir sonraki uçuş için hazırlıklar(yakıt ikmali, ikram ikmali vb.) yapılır. Uçağın pistte koştuğu süre de devir süresine dahildir. Devir süresi uçağa ve havaalanına bağlıdır, iç hat uçuşları için bu süre yaklaşık 30-40 dakikadır.

2.6 Kullanılabilir Koltuk Kilometresi

Kullanılabilir koltuk kilometresi (Available Seat Miles), yıl boyunca kullanılan toplam yolcu kapasitesi ile yıl boyunca koltukların uçtuğu mesafenin çarpımıdır.

2.7 Yolcu Gelir Kilometresi

Yolcu gelir kilometresi (Revenue Passenger Miles) tüm uçuşlarda satılan koltuk sayısı ile o koltukların uçtuğu toplam mesafenin çarpımıdır. YGK talep olarak kullanılır. Bunlara ek olarak YGK, KKK'dan nispeten düşüktür zira tüm uçuşlarda tüm koltukların satılması söz konusu değildir.

2.8 Verim

Verim bir havayolu şirketinin her satılan koltuk için kilometre başına kazandığı paradır. Verim toplam işletme gelirinin YGK' ya bölünmesi ile bulunur.

2.9 Kullanılabilir Koltuk Kilometresi Geliri

Birim gelir olarak da bilinir. Kullanılabilir koltuk kilometresi geliri tüm uçuşlar sonunda kullanılabilir koltuklardan elde edilen geliri temsil eder. KKKG toplam işletme gelirinin kullanılabilir koltuk kilometresine bölümü ile bulunur. Kullanılabilir koltuk kilometresi yolcu gelir kilometresinden nispeten büyük olduğundan; verim de kullanılabilir koltuk kilometresi değerinden nispeten büyüktür.

2.10 Kullanılabilir Koltuk Kilometresi Masrafı

Birim masraf olarak da bilinir. Kullanılabilir koltuk kilometresi masrafı bir koltuğu bir kilometre uçurabilme masrafıdır. KKKM toplam işletme masrafının KKK'na bölünmesi ile bulunur.

[1] numaralı referansta belirtildiği üzere Amerika Birleşik Devletleri'nde kullanılan filo tipleri ve KKK,YGK,KKKM değerleri şu şekildedir:

Çizelge 2.2 : Değişik filo tipleri için KKK, YGK ve KKKM değerleri

Uçak	**KKK**	**YGK**	**KKKM**
A300-600	749.266	483.523	5.50
A319	428.284	278.843	4.50
A320-200	518.301	360.241	4.50
A321	741.194	492.437	3.00
B737-200	266.352	176.271	6.20
B737-300	394.850	248.375	5.70
B737-400	412.051	273.492	7.10
B737-500	295.569	192.774	6.50
B737-700	572.830	394.469	3.10
B737-800/900	509.172	338.763	3.90
B747-200	1.551.265	1.053.479	5.50
B747-400	1.892.889	1.298.440	4.60
B757-200	686.353	463.571	4.40
B767-200	719.460	471.907	5.50
B767-300	1.016.661	648.583	4.20
B777-200	1.451.275	945.293	3.70
DC-10-30	1.224.904	882.289	4.20
DC-10-40	524.910	401.717	5.10
DC-10-10	886.418	727.173	3.70
MD-11	1.254.607	691.432	5.50
MD-80	394.273	257.716	5.90

2.11 İşletme Masrafı

Bir uçuş için işletme masrafı, o uçuşa atanan uçağa bağlı olarak değişir. İşletme masrafı şu şekilde hesaplanır:

$$OC = KKKM \times d \times f_c \qquad \textbf{(2.1)}$$

Denklem (2.1)'de "$\boldsymbol{OC}$" işletme masrafını, "$\boldsymbol{KKKM}$" kullanılabilir koltuk kilometresi gelirini, "$\boldsymbol{d}$" uçuşun mesafesini, "$\boldsymbol{f_c}$" ise koltuk kapasitesini göstermektedir.

2.12 Yolcu Kayıp Masrafı

Filo ataması probleminde önemli girdilerden biri de her uçuş için değişen taleptir. Büyük kapasiteli bir uçağı talebin düşük olduğu bir uçuşa atamak yüksek işletme masrafının yanında satılmamış koltuklara sebep olur. Aynı şekilde yüksek talepli bir uçuşa düşük kapasiteli bir uçak atamak yolcu kaybına sebep olur. Yolcu kayıp masrafı uçağa binemeyen yolculardan elde edilecek gelirin toplamıdır.

3. LİTERATÜR ARAŞTIRMASI

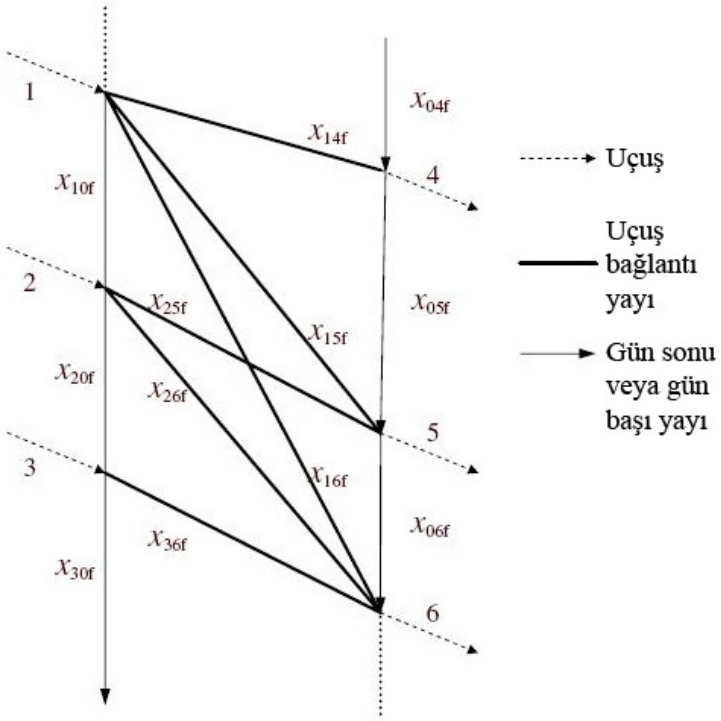
3.1 Bağlantı Tabanlı Filo Araştırması Modeli

[2] numaralı referans, gerçekçi boyuttaki filo ataması problemini ilk defa bağlantı-tabanlı ağ yapısı ile modellemeyi başarmış çalışmadır. Bu ağda, her düğüm noktası (node) uçağın kalkış ve iniş yaptığı anları simgeler. Bu düğüm noktalarına ek olarak gün başlangıcı ve sonunu modellemek üzere gerçekte var olmayan düğüm noktaları yaratılır. Böylelikle gün sonu ve başlangıcı da modeli dahil edilmiş olur.

[2] numaralı referansta kullanılan modelde farklı bağlantıları simgelemek için üç çeşit bağlantı yayı (arc) bulunur:

1. Uçuş bağlantı yayı: Bu yaylar kalkış ve iniş düğüm noktalarını birbirine bağlar.
2. Bitirici bağlantı yayı: Bu yaylar en son düğüm noktasını gün sonu düğüm noktasına bağlar. Bu bağlantı yayı en son düğüm noktasının bulunduğu yerde geceyi geçireceğini belirtir.
3. Başlatıcı bağlantı yayı: Bu yaylar ilk düğüm noktasını gün başı düğüm noktasına bağlar. Bu bağlantı yayı bağlandığı düğüm noktasında uçağın geceyi geçirdiğini belirtir.

Ağdaki bütün bağlantılar olurlu (feasible) olmalıdır. Model kurulurken uçağın minimum devir süresinde bir sonraki düğüm noktasına bağlanacağını düşünerek bağlantılar gerçekleştirilir. Filo tiplerini temsil eden ikili sisteme göre belirlenmiş karar değişkenleri, bu bağlantıları sağlayacak şekilde ayarlanır. [2] numaralı referanstan uyarlanan Şekil (3.1)'de 12 bağlantılı (6 uçuş bağlantı yayı, 3 bitirici bağlantı yayı, 3 başlatıcı bağlantı yayı) bir istasyon çizilmiştir.



Şekil 3.1 : Bağlantı tabanlı istasyon [2].

[2] numaralı referansta kullanılan matematik model şu şekildedir:

$$en\ b\ddot{u}y\ddot{u}kle \sum_{i\in L\cup\{0\}} \sum_{j\in L} \sum_{f\in F} p_{jf} x_{ijf} - c \sum_{j\in L} \sum_{f\in F} x_{0jf} \quad \textbf{(3.1)}$$

$$bağlı\ olarak \sum_{i\in L\cup\{0\}} \sum_{f\in F} x_{ijf} = 1 \quad \forall j \in L, \quad \textbf{(3.1a)}$$

$$\sum_{i\in L\cup\{0\}} x_{ilf} - \sum_{j\in L\cup\{0\}} x_{ljf} = 0, \quad \forall l \in L, \forall f \in F, \quad \textbf{(3.1b)}$$

$$\sum_{i\in D_s} x_{0if} - \sum_{i\in A_s} x_{i0f} = 0, \quad \forall s \in S, \forall f \in F, \quad \textbf{(3.1c)}$$

$$\sum_{i\in L} x_{0if} \le A_f, \forall f \in F, \quad \textbf{(3.1d)}$$

$$x\ binary \quad \textbf{(3.1e)}$$

Abara'nın modelindeki notasyona bakacak olursak, L, i ve j ile indisli uçuş ayakları kümesi; F, f ile indisli filo tipi kümesi; S, s ile indisli istasyonlar kümesidir. Diğer yandan $A_s$ve D_s s ile indislenmiş istasyonun iniş ve kalkış kümesidir. x_{ijf} İkili sisteme sahip bir karar değişkenidir: eğer i ve j ayakları arasındaki bağlantı f filo tipine atanıyorsa karar değişkeni 1 değerini alır. Aksi takdirde, yani i ve j ayakları arasındaki bağlantı f filo tipine atanmıyorsa karar değişkeni 0 değerini alır ($i = 0, j = 0$ değerleri sırasıyla gün başlangıcı ve gün bitişi yaylarını temsil eder.)

Modelde üç adet kısıt fonksiyonu bulunur:

1. Kapsama
2. Denge
3. Müsaitlik(Availability)

Modeldeki amaç kar ve işletme masrafı farkını maksimize etmektir. f filo tipini j kalkış noktasından kalkan bir uçağa atamanın getireceği kar p_{jf}ile gösterilir.

Abara modelinde c ile indislenmiş göstermelik(nominal) bir masraf kullanır. Bu masraf x_{0jf} ile indisli her j uçuş ayağı için f filo tipinin atanması sonucunda kullanılır ve sabittir. Aslında her uçuş ayağı için masraf tek tek hesaplanmalıdır ancak [2] numaralı referansta hesap yükü artacağından bu hesap yapılmamıştır[3].

Bağlantı tabanlı modelde kapsama kısıt fonksiyonu(3.1b) bir gün başlangıcı yada iniş düğüm noktasına ihtiyaç duyar. Denge fonksiyonu (3.1c), ağdaki her uçuş ayağı için akışın dengesini sağlar. Ek olarak, zamanlama denge kısıt fonksiyonu (3.1d) her gece her istasyonda kalan uçak sayısının dengede olmasını sağlar. Böylelikle her gün aynı filo ataması gerçekleştirilmiş olur. Müsaitlik kısıt fonksiyonu (3.1e) A_f ile indisli f filo tipinin uçak sayısını sınırlandırır.

[2] numaralı referanstaki çözüm yaklaşımında (3.1d) ve (3.1e) kısıtları "yumuşak" kısıtlar olarak ele alınır ve amaç fonksiyonuna eklenecek yaptırım(penalty) fonksiyonları ile rahatlatılabilir. Yani (3.1d) kısıtı bir istasyon için gerekli gün başı ve gün sonu yaylarının varlığını sağlayacak bir yaptırım terimi içerirken, (3.1e) kısıtı gün boyunca kullanılan filo sayısının üstünde uçak kullanımını mümkün hale getiren bir yaptırım terimi içerir. Dolayısıyla amaç fonksiyonu beklenen kar ve işletme masrafının yanı sıra rahatlatılmış kısıt fonksiyonlarından gelen yaptırım terimlerini de içerir.

Yukarıda da belirtildiği gibi bu modelde tüm olurlu bağlantılar ağ içinde gösterilmelidir. Bu modelde seçilen tüm bağlantıların olurlu olmasını sağlar. Bunun sonucu olarak, model çok büyük bir hal alabilir, zira bir ağda pek çok olurlu bağlantı vardır. Bu problemi aşmak için bir uçuşta kullanılacak bağlantı değişkenlerine bir sınırlama getirilir[3].

Çok benzer bir model formülasyonu kullanarak [4] numaralı referansta problemi daha efektif kullanmak için bazı ön işlem teknikleri geliştirmiştir. Her çoklu bağlantıya sahip istasyonu bölerek bağlantıları teke düşürmüş, böylece iniş ve kalkış olurlu çözümlerini tek noktada toplamayı başarmışlardır.

[4] numaralı referanstaki çözüm sezgisel(heuristic) bir çözümdür. Önerilen çözümde ilk olarak lineer programlama rahatlatması çözülür ve buradan elde edilen sonuç dallandırma ve birleştirme(branch and bound) algoritmasına aktarılır

3.2 Zaman-Uzay Ağı Tabanlı Filo Araştırması Modeli

[2] numaralı referansta teklif edilen bağlantı ağının tersine, zaman-uzay yapısı uçuş ayakları üzerinde yoğunlaşır. Bu sebeple olurlu olduğu sürece bağlantıları modelin kurmasına izin verir. Bu bağlantıları kurmada özgürlük sağladığı gibi karar değişkeni sayısını da önemli şekilde düşürür çünkü uçuş sayısı, olurlu bağlantı süresinden çok daha azdır.

[4]'te de belirtildiği üzere zaman-uzay ağı yerdeki uçaklar arasında ayırımda yetersiz kalmaktadır. Bu sebeple müteakip planlamayı zorlaştırır.

Buna rağmen 1993'te Berge ve Hopperstad ve Hane et al. [5] ile devam eden zaman-uzay ağı akımı halen devam etmektedir. Zaman-uzay ağı yapısı halen filo ataması problemlerinin çözümünde kullanılmaktadır.

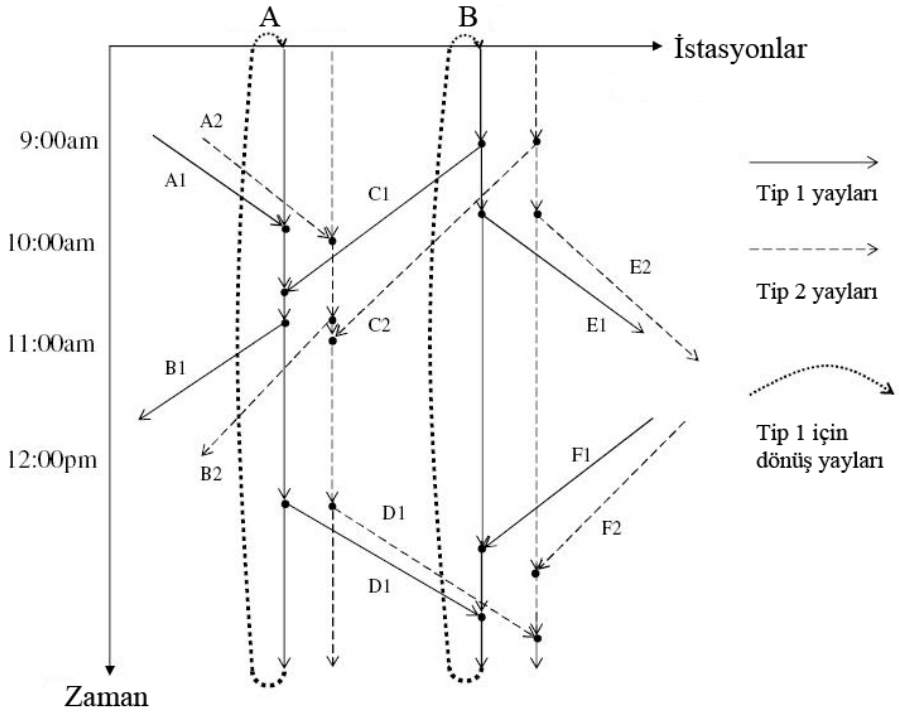
Zaman-uzay ağı gösterimi ağ kümelerini, her filo tipi için, birbirine ekler. Bu filo tipinden bağımsız uçuş ve devir zamanlarına imkan verir. Eğer uçuş ve devir zamanları birbirinden çok da farklı değil ise, benzer filo tipleri için tek bir ağ gösterimi kullanılabilir. Her filo tipinin ağında kalkış ve inişler belirli zamanlarda bir düğüm noktası ile işaretlenir. Zaman-uzay ağında da bağlantı ağı gibi üç tip yay vardır:

1. Yer yayları: Bu yaylar uçağın belirli bir istasyonda belirli bir zaman kaldığını gösterir.
2. Uçuş yayları: Uçuş ayaklarını temsil eder.
3. Dönüş yayları: Gün sonu olaylarını bir sonraki günün olaylarına bağlar. Bu "dönüş" filo atamasının her gün sürekli olmasını sağlar.

Havaalanları da her filo tipi için kopyalanır. Her filo tipinin alt ağında bir ağ zaman çizgisi oluşturulur. Bu çizgi gün içinde olan tüm olayları, gün sonu ve gün başı olaylarını içerir.

Uçuş ayağı tabanlı akışları gösterebilmek için aynı filo tipine sahip uçakların ağ zaman çizgileri kalkış ve iniş yayları ile simgelenir. Bir uçuş ayağına sadece bir filo tipinin atanması akışı düzenlediği gibi bütün ağları birbirinden bağımsızlaştırır.

Şekil (3.2)'de iki filo tipi için iki istasyonlu bir zaman-uzay ağını gösterilmektedir. Zaman çizelgesi şekilde aşağı doğru devam eder, her düğüm noktası bir istasyondaki iniş veya kalkışı simgelemektedir. Her olay zaman içinde oluş sırasına göre yukarıdan aşağıya doğru sıralanmıştır. Şekilde iki A ve B şeklinde iki istasyon bulunur. Kesiksiz çizgiler filo tipi 1'i simgelerken kesikli çizgiler filo tipi 2'yi simgeler (Şekildeki diğer detaylar şekil üzerinde anlatılmıştır).



Şekil 3.2 : Zaman-uzay ağı tabanlı istasyon [5].

Model formülasyonunda, herhangi bir yay üzerindeki akış bir ikili(binary) karar değişkeni tarafından gösterilir böylelikle ilgili uçuş ayağının sadece bir filo tipi tarafından kapsanması sağlanır. Yer yaylarındaki ve dönüş yaylarındaki akışlar ise tam sayı değerleri alırlar. Bu değerler belirli bir zamanda belirli bir istasyonda bulunan uçak sayısını temsil eder.[6]

Genel olarak modelde üç adet ana kısıt fonksiyonu vardır:

1. Kapsama
2. Denge
3. Müsaitlik

Bunlara ek olarak uçuş yaylarının yoğunluğunun azaldığı saatlerde kullanılmakta olan uçak sayılarının toplamı kısıtlanır. Böylelikle kullanılan uçak sayısı filoda bulunan uçak sayısını geçemez. Akış denge kısıt fonksiyonu müsaitlik kısıt fonksiyonunun her ağın her anında geçerli olmasını sağlar.

[5] numaralı referansta önerilen matematik model ise şu şekildedir:

$$en\ k\text{üçü}kle \sum_{i\in L}\sum_{f\in F} c_{fl}x_{fl} \tag{3.2}$$

$$bağlı\ olarak \sum_{f\in F} x_{fl} = 1,\ \ \forall l \in L, \tag{3.2a}$$

$$\sum_{o\in S} x_{fost} + y_{fst^{-}t} - \sum_{d\in S} x_{fsdt} - y_{fstt^{+}} = 0, \quad \forall\{fst\} \in N, \tag{3.2b}$$

$$\sum_{l\in O(f)} x_{fl} + \sum_{s\in S} y_{fst_n t_1} \leq A_f, \quad \forall f \in F, \tag{3.2c}$$

$$x\ binary, \quad y \geq 0. \tag{3.2d}$$

Notasyona bakılırsa

- S ağdaki istasyonlar kümesidir. s, o, d ile simgelenir.
- F filo tipi kümesidir. f ile simgelenir.
- L zamanlanmış uçuş ayakları kümesidir. l veya $\{odt\}$ ile simgelenir. Burada $o, d \in S$ ve t uçuşun başladığı anı simgeler.
- N ağdaki düğüm noktaları kümesidir ve $\{fst\}$ ile simgelenir.
- Burada $f \in F, s \in S$ ve t olayın zamanını gösterir.

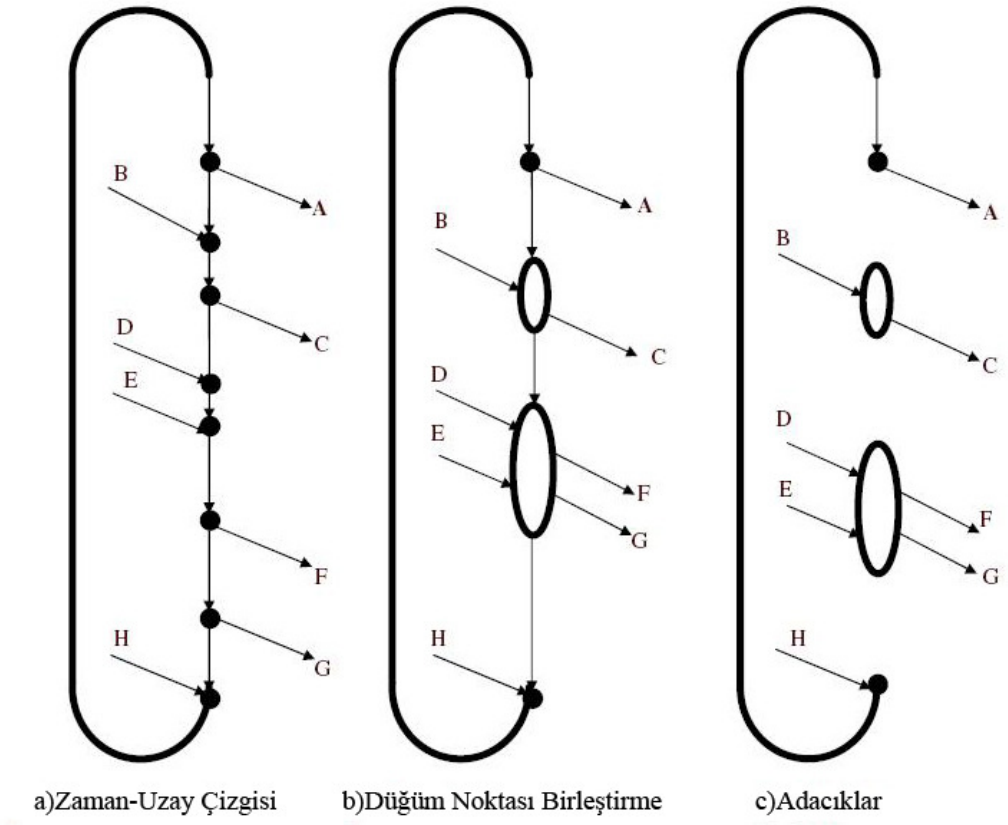
- $O(f)$ uçak zaman sayım çizgisinden geçen f filo tipine ait uçakların bulunduğu yaylar kümesidir. ($f \in F$)
- c_{fl} f filo tipini l uçuş ayağına atama masrafıdır. ($f \in F, l \in L$)
- A_f f filo tipi için müsait uçak sayısıdır. ($f \in F$)
- $x_{fl} = \begin{cases} 1, eğer\ filo\ tipi\ f\ l\ uçuş\ ayağına\ atandıysa, (f \in F; l \in L) \\ 0,\ aksi\ halde \end{cases}$
- (x_{fl} karar değişkeni $f \in F, \{odt\} \in L$ halinde x_{fodt} şeklinde de gösterilebilir)
- $y_{fstt'}$ f filo tipi için yer yayı düğüm noktasından ($\{fst\} \in N$), $s \in S$ istasyonu düğüm noktasına ($\{fst'\} \in N$) olan akıştır. Burada $f \in F$ ve genel anlamda $t' > t$, dönüş yayları için $t' \leq t$dir
- t^-, t^+ zaman çizgisinde önden gelen ve müteakip zamanları simgeler.

Daha önce de belirtildiği gibi (3.2a),(3.2b) ve (3.2c) sırasıyla kapsama, denge ve müsaitlik kısıt fonksiyonlarıdır.

Şu bir gerçektir ki yüzlerce istasyondan ve binlerce uçuştan oluşan bir ağ için filo ataması problemini çözmek zordur[3]. Ancak 1994'te [6] numaralı referans bu problemi müsaitlik kısıt fonksiyonu olmadan üç filo tipi için NP-Hard(Non deterministic polynomial-time hard) modellemiştir. Problemin çözümünün zorluğunu bilen Hane et al.[5] problemin çözümünü kolaylaştırması için bir dizi önişlemler ortaya koymuştur. Bu ön işlemler şu anda literatürde standart kabul edilir ve neredeyse yapılan her çalışmada kullanılır.

İlk önişlem gözleme dayalıdır. Ağda bütün bağlantıların doğru olup olmadığı, her düğüm noktasının doğru noktaya yerleştirilip yerleştirilmediği bu aşamada incelenir.

Bunlara ek olarak birbirini izleyen inişler ve kalkışlar aynı düğüm noktası ile gösterilebilir. Bu işleme düğüm noktası birleştirme(node aggregation) denir.



Şekil 3.3 : Düğüm noktası birleştirme [5].

Şekil (3.3) bu tekniği göstermektedir. Şekil (3.3)'ün (a) kısmı bir istasyon için bir filo tipine göre ağ zaman-çizgisini temsil eder. Bir önceki şekille aynı olarak düğüm noktaları zaman içindeki konumlarına göre yukarıdan aşağıya doğru sıralanmıştır. Şeklin (b) kısmı ise bir istasyonda düğüm noktası birleştirme işlemini göstermektedir. Ancak şu göz önünde bulundurulmalıdır ki şekildeki istasyon için başka düğüm noktası birleştirme işlemi yapılamaz. Örnek olarak B geliş yayı A gidiş yayı ile aynı düğüm noktasını paylaşamaz zira A kalkışı B kalkışından daha öncedir ve B,A arasında yapılacak bir bağlantı olursuz (infeasible) olacaktır. [5] referansındaki deneylerde düğüm birleştirme işleminin matristeki satır sayısını üç ila altı kat, sütün sayısını ise bir ila üç kat arasında azaltmayı başarmıştır.

İkinci önişlem ise bazı istasyonların, özellikle seyrek uçuş olayı olan ve belirli zamanlarda yerde hiç uçak bırakmayan istasyonların incelenmesine dayanır. Böyle durumlarda yer arkları sıfır akışa sahip olacağından ağdan silinebilir. Silinen her yer yayı için eşit miktarda iniş ve kalkış sebebiyle değeri sıfır olan bir başka yer yayı çıkar ve bu yer yayı da ağdan silinebilir. Bu basitleştirme işlemi zaman-çizgisinde

adacıklar oluşturur. Şeklin (b) kısmında çıkartılan veya silinen sıfır akışlı yer yayları sebebiyle (c) kısmında oluşan adacıklar görülebilir.

Üçüncü önişlem kayıp bağlantıları elimine eder. Birbirini izleyen iki uçuş uzun devir süreleri sebebiyle bir filo tipine atandıklarında kayıp bağlantıya sebep olabilir ve ağdan çıkartılabilir. Örnek olarak bir önceki şekilde eğer C2 ve B2 uçuş yayları birbiri ardına uçuluyorsa kayıp bağlantıya sebep olurlar ve ikinci filo tipinin ağından çıkartılmalıdırlar.

Bu üç önişlem problem boyutunu büyük ölçüde düşürür. [5] tarafından da belirtildiği üzere 48982 satır ve 66942 sütuna sahip büyük boyutlu bir problem 7703 satır ve 20464 sütün sayısına düşmüştür.

Hane et al. tarafından sunulan bu model havacılık endüstrisinde bir devrim niteliği taşır. Abara'nın 1989'da sunduğu model %1.4'lük bir kar artışı sağlarken Rushmeir ve Kontogiorgis US Airways'te senelik 15 milyon$ kar sağlandığını belirtir.[4]

4. KARMAŞIK TAM SAYI PROGRAMLAMA

Doğrusal programlama hemen her tür problemi çözecek yapıda olmasına rağmen işletme uygulamalarında ortaya çıkan sonuç ya işletmenin üretim şeklinden dolayı yada doğrusal programlamanın uygulandığı sorunun yapısından dolayı istenilen şekilde olurlu sonuç vermeyebilir. Çünkü ekonomik yaşamda her zaman girdi ve çıktıların bölünmezlik sorunları ile karşılaşılmaktadır. Bölünmezlikleri ele alınan problemlerin çözümleri de tamsayı olmalıdır. Modellerin uygulanmasında, değişkenlerin tam sayı olma şartının incelenmesi ve araştırılması durumunda Tam Sayılı Doğrusal Programlama (TDP) kullanılmalıdır [7].

Tam sayılı doğrusal programlama tekniği, doğrusal programlamanın bir uzantısı olup doğrusal programlamada meydana gelebilecek gerçekçi olmayan sonuçları ortadan kaldırmayı amaçlar. Bazı doğrusal programlama modellerinde sonuçların tam sayı çıkmaması problemin gerçek hayattaki problemlere uygunluğunu bozmaktadır. Örneğin bir üretim probleminde masa ve sandalye üretimi yapılacaksa sonuçların kesirli çıkması gerçekçi olmamaktadır. Sonuçların tam sayıya yuvarlatılması bazı kısıtları bozabileceği için çözüm olmamaktadır. Tam sayılı doğrusal programlama tekniği, kısıtları bozmadan sonucun tam sayı olmasını sağlamaktadır.

Kısacası tamsayılı doğrusal programlama, değişkenlerin bazılarının veya tamamının tamsayı değeri aldığı bir doğrusal programlama türüdür.

Doğrusal programlama ile tamsayılı doğrusal programlama arasındaki temel fark, doğrusal programlama modelinde karar değişkenlerinin sıfır ve sıfırdan büyük olma koşulu aranırken, tam sayılı doğrusal programlamada değişkenler sıfıra eşit yada sıfırdan büyük tam sayı olmalıdır.

Filo ataması problemlerinde de atanacak uçak sayılarının tam sayı olması gerekir. Aksi halde çözüm olurlu olmayacağı gibi, çözüm de gerçekçi olmaz. Bu sebeple filo ataması problemleri tam sayı programlama ile modellenir.

Doğrusal tamsayı programlama(DTP) içerisinde tamsayı ve tamsayı olmayan değişkenler, doğrusal kısıtlar içeren bir modeldir. Herhangi bir KTP şu şekilde yazılabilir:

$$(DTP) \quad Z(X) = \min_{x,y}\{cx + fy : (x,y) \in X \tag{4.1}$$

Burada X, m adet doğrusal kısıtla ve x,y üzerinde pozitiflik kısıtları ile sıkıştırılmış olurlu çözümler kümesidir. Matris gösterimi ise şu şekildedir:

$$X = \{(x,y) \in R_+^n * Z_+^p : Ax + By \geq b\} \tag{4.2}$$

Burada;

- $Z(X)$ X olurlu çözüm kümesindeki amaç fonksiyonu optimum değeridir.
- x Pozitif sürekli değişkenlerin n boyutlu sütün vektörüdür.
- y Pozitif tamsayı değişkenlerin p boyutlu sütun vektörüdür.
- $c \in R^n, f \in R^p$ Amaç katsayılarının satır vektörüdür.
- $b \in R^m$ m adet kısıt fonksiyonunun sağ taraf değerleri vektörüdür.
- A, B Kısıtların (mxn) ve (mxp) boyutlu reel sayı katsayıları matrisleridir.

Tamsayı programlama modelleri için şu ana kadar dört ana çözüm yolu geliştirilmiştir. Bunlar:

- Yorucu numaralama
- Dal-sınır algoritması
- Kesme düzlemi yöntemi
- Genetik algoritma yaklaşımlarıdır.

4.1 Yorucu Numaralama

Yorucu numaralama optimizasyon tam sayı programlama problemlerinin çözümünde kullanılan en basit tekniktir. Algoritma tüm çözümler içinde en olurlu çözümü bulmaya çalışır. Bir enküçükleme probleminde tüm olurlu çözümlerin bulunduğu kümede en küçük çözüm aranır. Yapılan işlemlerin sayısı şu şekilde hesaplanır:

$$n_e = \prod_{i=1}^{n_d} p_i \qquad \textbf{(4.3)}$$

Eğer n_d ve p_i değerlerinden biri yada her ikisi de büyükse bu işlem sayısını yüksek oranda arttırır. Aynı zamanda tamsayı değerler arasındaki işlemleri de büyütür. Bir karmaşık tamsayı probleminde, n_e kadar olurlu çözüm vardır. Eğer çözülmeye çalışılan problem basit hesaplamalar içeriyorsa problemi çözmek çok da zor değildir. Eğer problem sonlu elemanlar veya sonlu türev gibi zor işlemler içeriyorsa hesaplamaların yapılması olurlu çözüm kümesindeki her eleman için çok zor ve uzun olacaktır. Böyle durumlarda tamsayı olması beklenen karar değişkenleri bir gruba toplanırsa hesaplamalarda kolaylık sağlanabilir.

Yorucu numaralama algoritmasını bir örnek üzerinde şöyle gösterebiliriz:

$$enb\ddot{u}y\ddot{u}kle\ Z = 0.2x_1 + 0.3x_2 + 0.5x_3 + 0.1x_4 \qquad \textbf{(4.4)}$$

$$bağlı\ olarak\ 0.5x_1 + x_2 + 1.5x_3 + 0.1x_4 \leq 3.1 \qquad \textbf{(4.4a)}$$

$$0.3x_1 + 0.8x_2 + 1.5x_3 + 0.4x_4 \leq 2.5 \qquad \textbf{(4.4b)}$$

$$0.2x_1 + 0.2x_2 + 0.3x_3 + 0.1x_4 \leq 0.4 \qquad \textbf{(4.4c)}$$

$$x_j = 0\ veya\ 1, \qquad j = 1, \dots, 4 \qquad \textbf{(4.4d)}$$

Olurlu çözüm kümesinde $2^4 = 16$ çözüm vardır. Bunlar bir matriste şu şekilde gösterilebilir:

x_1	x_2	x_3	x_4
0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	1	0	0
0	1	1	0
0	0	1	1
1	0	1	0
1	0	0	1
0	1	0	1
1	1	1	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

(4.5)

Yorucu numaralama tüm olurlu çözümler için kısıt fonksiyonlarına bağlı olarak olasılıkları dener ve $x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1$ enbüyüklenmiş sonucuna ulaşır.

Problem boyutu ne kadar büyürse yapılması gereken hesap sayısının da o kadar büyüyeceği açıktır. Eğer elimizde sadece iki değer alabilen 100 adet tamsayı karar değişkeni olsaydı olurlu çözüm kümesinde 2^{100} adet çözüm olurdu. Bu kadar büyük sayıda işlemin yapılması teorik anlamda mümkün olsa da pratikte imkansızdır. Ancak günümüzün bilgisayar teknolojisi bu tarz büyük problemlerin çözümüne imkan vermektedir.

4.2 Dal ve Sınır Algoritması

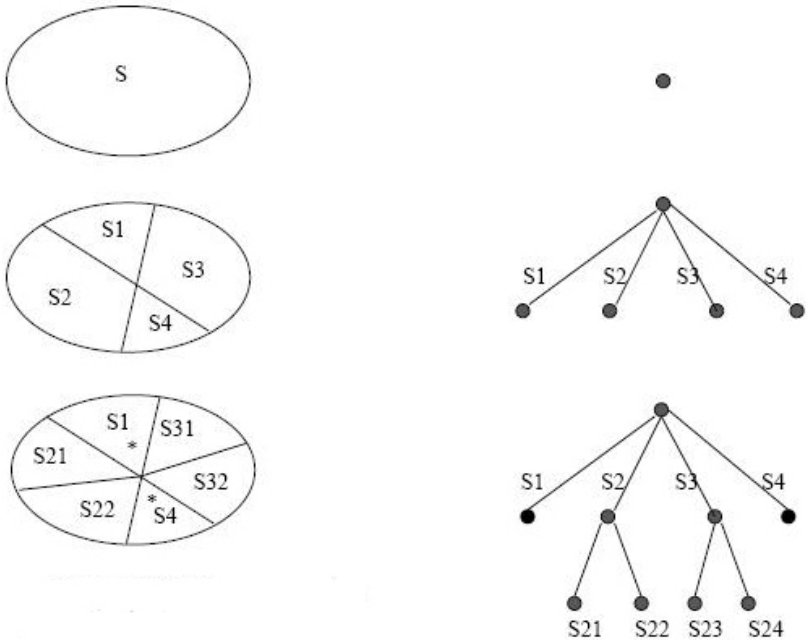
A. H. Land ve A. G. Doig tarafından 1960 yılında ortaya konulan dal ve sınır algoritması pek çok eniyileme probleminin çözümü için kullanıldığı gibi tamsayı programlama ve tümleşik eniyileme problemleri için de kullanılmaktadır.[9] Algoritma genel olarak tüm çözüm adaylarının sistemli numaralanmasına dayanır; algoritma olursuz çözüme sebep olan adayları üst ve alt sınırlar belirleyerek çözüm kümesi dışında tutar, böylelikle yorucu sayılamadan daha çabuk yakınsar.

Geleneksel dal ve sınır algoritması kısaca şu şekilde özetlenebilir: Her adımda pek çok alt problem dallandırılır ve dalların içerdiği problemler çözülür. Dallar arasında olurlu bir çözüm bulunduktan sonra bulunan en olurlu çözüm bir sonraki dallandırma için üst sınır olarak belirlenir. Bir sonraki dallandırma belirlenen üst sınır için yapılır,

ve işlem en iyi çözüme ulaşana kadar tekrarlanır. Dal ve sınır algoritması enküçükleme problemleri üzerine kurulur ancak algoritmayı enbüyükleme problemleri için de uyarlamak mümkündür.

$$(enb\ddot{u}y\ddot{u}kle\ f(x) \equiv enk\ddot{u}\varsigma\ddot{u}kle\ g(x),\ \ g(x) = -f(x)) \quad \textbf{(4.6)}$$

Dal ve sınır algoritması arama metodu şu şekilde gösterilebilir:



Şekil 4.1 : Dal ve sınır algoritması.

S çözüm kümesi ilk aşamada dört adet alt kümeye ayrılır ve bu kümeler için problemler çözülür. Olurlu çözüme sahip S2 ve S3 kümeleri tekrar dallandırılır. Şekilde "*" ile işaretli S1 ve S4 çözüm kümeleri olurlu çözüm içermediklerinden bir sonraki dallandırma aşamasında kullanılmamıştır.

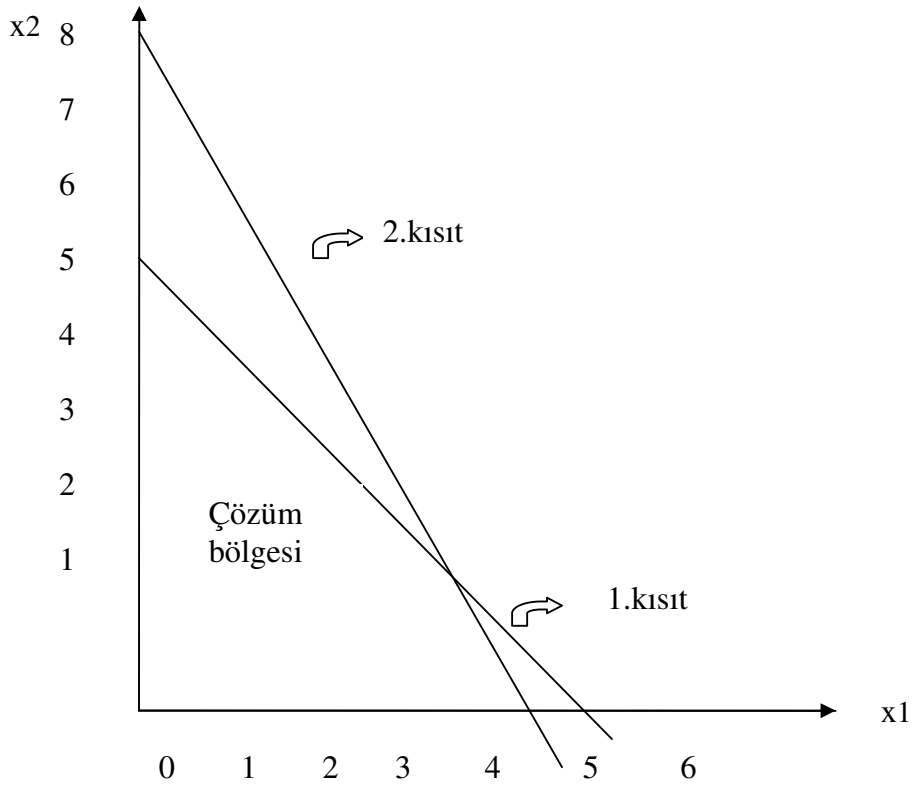
Aşağıdaki doğrusal programlama probleminin grafik çözümünden sonuçların tam sayı çıkmadığı görülmektedir.

$$enb\ddot{u}y\ddot{u}kle\ Z = 5x_1 + 4x_2 \quad \textbf{(4.7)}$$

$$ba\breve{g}l\imath\ olarak\ x_1 + x_2 \leq 5 \quad \textbf{(4.7a)}$$

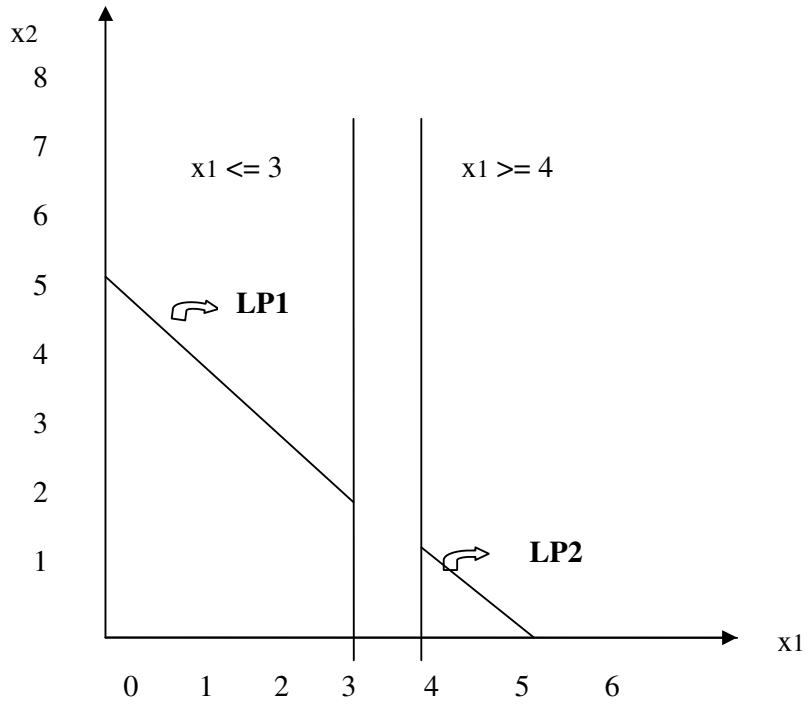
$$10x_1 + 6x_2 \leq 45 \quad \textbf{(4.7b)}$$

$x_1, x_2 \geq 0$ **(4.7c)**



Şekil 4.2 : Birinci çözüm uzayı.

Problemin tamsayı kısıtları yok sayılıp problem çözüldüğünde lineer çözüm Z=23.75, x_1=3.75, x_2=1.25 bulunur. Değişkenler tam sayı çıkmadığı için dal-sınır algoritması ile eniyilenmiş tam sayılı çözümü buluncaya kadar çözüm uzayının düzenlenmesi yapılmalıdır. İlk aşamada $LP0$ çözümünde (Doğrusal programlama çözümü) tam sayı değer almayan bir değişken rasgele seçilir. x_1 değişkenini seçelim (x_1=3.75) $LP0$ çözüm uzayının $3 < x_1 < 4$ bölgesinde tamsayılı değerler olmayacaktır dolayısıyla bu bölge elemine edilebilir.



Şekil 4.3 : İkinci çözüm uzayı.

$$LP1\ uzayı\ =\ LP0\ uzayı\ +\ (x_1\ \leq\ 3) \quad \textbf{(4.8)}$$

$$LP2\ uzayı\ =\ LP0\ uzayı\ +\ (x_1\ \geq\ 4) \quad \textbf{(4.8a)}$$

Eniyilenmiş çözüm ya $LP1$ uzayında yada $LP2$ uzayında olacaktır. Her iki alt problem ayrı ayrı çözülmelidir. Önce $LP1$ problemi $(x_1 \leq 3)$ kısıtı eklenerek çözülürse:

$$enbüyükle\ Z = 5x_1 + 4x_2 \quad \textbf{(4.7)}$$

$$bağlı\ olarak\ x_1 + x_2 \leq 5 \quad \textbf{(4.7a)}$$

$$10x_1 + 6x_2 \leq 45 \quad \textbf{(4.7b)}$$

$$x_1 \leq 3 \quad \textbf{(4.7c)}$$

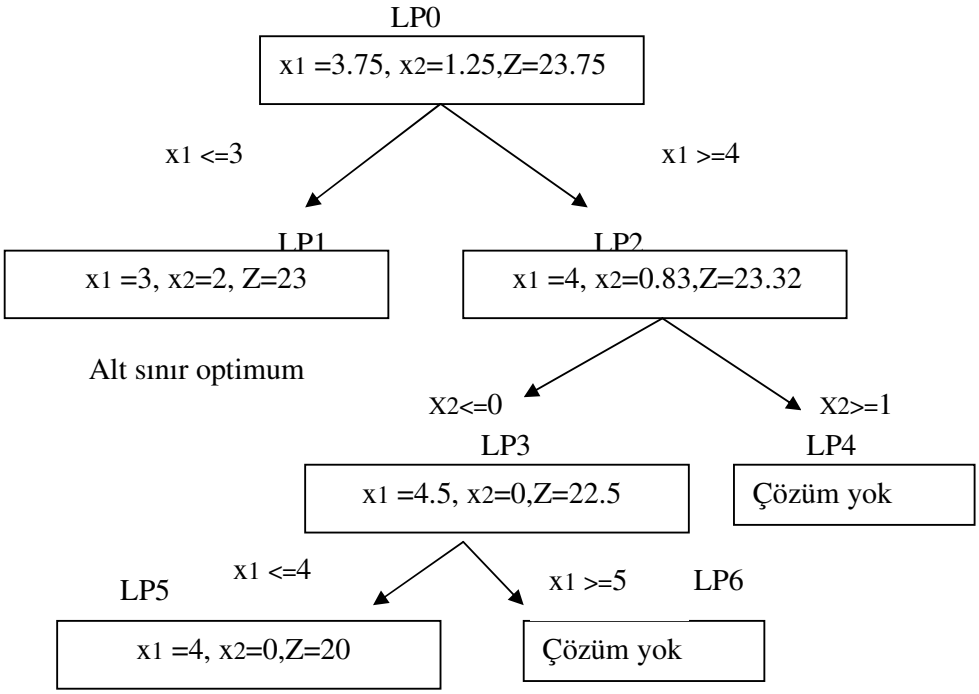
$$x_1, x_2 \geq 0 \quad \textbf{(4.7d)}$$

Problem çözüldüğünde $Z = 23,\ x_1 = 3,\ x_2 = 2$ çıkmaktadır. Bu çözümde değişkenler tamsayı değeri aldıklarından $LP1$ uzayı eniyilenmiş denebilir. Çözümdeki $Z = 23.75$ değeri $LP1$ de $Z = 23$ olarak çıktığına göre bu bir alt sınır olarak alınabilir. Bir tamsayılı çözüm elde edildiği için daha fazla ilerletmeye gerek yoktur. $LP2$ ye ait çözümde de $Z = 23.32\,, x_1 = 4,\ x_2 = 0.83$ çıkmaktadır. $x_2 = 0.83$ olduğundan yeniden bir dallanma yapılarak; $x_2 \leq 0\ ve\ x_2 \geq 1$ kontrolü yapılabilir. Buradan:

$$LP3 = LP2\ + (x_2 \leq 0) = LP0 + (\ x_1 \leq 4) + \ (x_2 \leq 0) \qquad \textbf{(4.8)}$$

denilebilir.

LP5 çözümünde de sonuç tamsayılı çıkmaktadır. Ancak $LP1$ çözümünde $Z = 23$ alt sınır olarak alınırsa(en büyük alt sınır) bu çözümün eniyilenmiş olmadığı söylenebilir. Burada hangi dalın seçilip önce çözülmesi konusunda kesin bir kural olmayıp seçim tahmini yapılmaktadır. Problem aynı şekilde devam eder. Şu ana kadar örnek problem için dallandırma aşamaları Şekil (4.4)'de gösterilmiştir.



Şekil 4.4 : Örnek problem için dal ve sınır gösterimi.

4.3 Gomory Kesme Düzlemi Yöntemi

Ralph E. Gomory tarafından ortaya atılan kesme düzlemi yöntemi çözüm ağını bazı kısıtlarla (kesmelerle) daraltmaya verilen isimdir. Tam sayı programlamada kesme düzlemi yöntemleri sıklıkla kullanılmaktadır.

Kesme düzlemi yönteminde ilk önce verilen tam sayı programlama probleminin tam sayı kısıtları kaldırılarak lineer rahatlatılmış çözümü bulunur. Eğer bulunan çözüm tam sayı ise algoritma durdurulur. Eğer değilse eniyilenmiş çözümüm bulunduğu olurlu çözüm kümesi bir kesme ile sınırlandırılır. Bu kesmenin nereye çizileceği problemine ayırma problemi denir. Aynı şekilde ilk doğrusal çözümde tamsayı değeri elde edilmezse tamsayı olmayan çözüm de bir kesme ile olurlu çözüm kümesinden ayrılabilir. Bu işlem eniyilenmiş tamsayı çözüm bulunana kadar tekrarlanır.

Kesme düzlemi matematiksel formülasyonu şu şekildedir:

X geçerli bir çözüm ve B'de geçerli çözümün olduğu alt küme olsun.

$$[B\ F]\begin{bmatrix} x_b \\ x_f \end{bmatrix} = b \tag{4.9}$$

Buradan,

$$x_b = B^{-1}b - B^{-1}Fx_f = \overline{b} - \overline{A}x_f \rightarrow x_b + \overline{A}x_f = \overline{b} \tag{4.10}$$

Buradan çıkan sonuç kesirli ise x n. dereceden kesirlidir denir.

$$x_n + \sum_{j\in\varphi}^{N} \overline{a_{n,j}}\, x_j = \overline{b_n} \tag{4.11}$$

$$[x_b] + \left[\overline{A}\right]\left[x_f\right] = [\overline{b}] \tag{4.12}$$

Her çözüm için $x_i \geq 0 \forall i$ olduğundan:

$$x_n + \sum_{j\in\varphi}^{N} \left\lfloor \overline{a_{n,j}} \right\rfloor x_j \leq \overline{b_n} \tag{4.13}$$

Olur.

Kesirli eniyilenmiş çözümü, tam sayı çözümleri kaybetmeden çıkartabilmek için sağ taraf değeri en yakın tam sayı değerine yuvarlanır:

$$x_n + \sum_{j\in\varphi}^{N} \lfloor \overline{a_{n,j}} \rfloor x_j \leq \lfloor \overline{b_n} \rfloor \qquad \textbf{(4.14)}$$

(4.11) ve (4.14) numaralı denklemler birbirinden çıkartıldığında kesme düzlemi yani Gomory'nin tamsayı kesme düzlemi formülü ortaya çıkar:

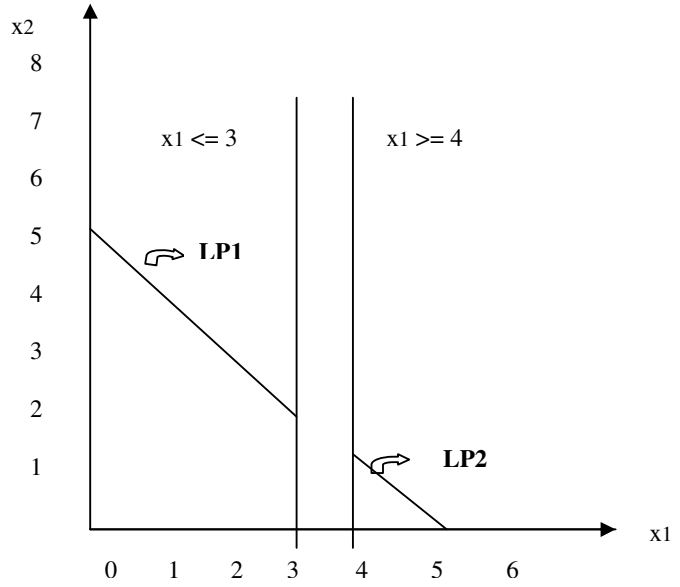
$$\sum_{j\in\varphi}^{N} (\overline{a_{n,j}} - \lfloor \overline{a_{n,j}} \rfloor) x_j \geq \overline{b_n} - \lfloor \overline{b_n} \rfloor \qquad \textbf{(4.15)}$$

Yöntemi grafik olarak göstermek için bir amaç fonksiyonu ve bir kısıt ele alalım.

$$enküçükle\ Z = 2x_1 + 5x_2 \qquad \textbf{(4.16)}$$

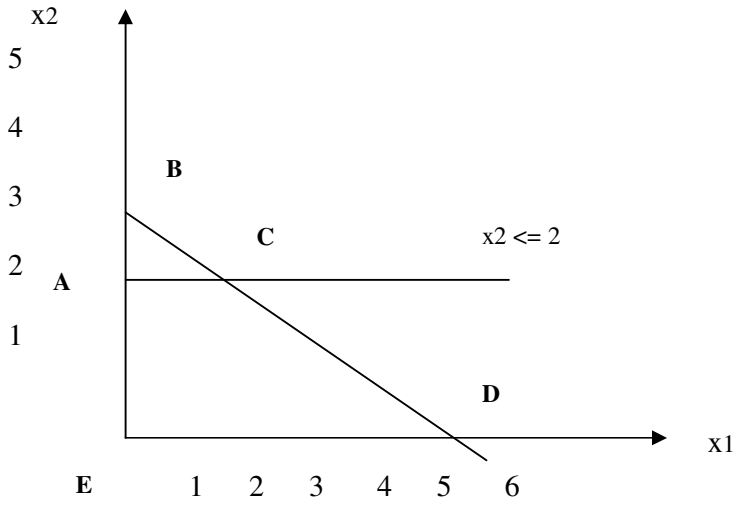
$$bağlı\ olarak\ 3x_1 + 6x_2 \leq 16 \qquad \textbf{(4.16a)}$$

$x_1 = 16/3 = 5.3\,, x_2 = 16/6 = 2.6$ doğrularını çizdiğimizde EDB alanı eniyilenmiş çözüm bölgesini oluşturmaktadır. Bu bölgedeki B noktası ($x_1 = 0, x_2 = 2.6$) amaç fonksiyonunu maksimum yapan noktadır. Ancak sonuç tamsayı çıkmadığı için probleme bir kısıt daha eklenip işleme devam edilir.



Şekil 4.5 : Gomory Kesme Düzlemi İçin 1. Çözüm Uzayı.

Yeni kısıt : $x_2 \leq 2$ kısıtı olsun.



Şekil 4.6 : Gomory Kesme Düzlemi İçin 2. Çözüm Uzayı.

ABC ile gösterilen alanda tamsayılı çözüm olmadığından yeni çözüm bölgesi EACD alanı olup problemin tamsayılı çözümlerini içermektedir. Bu bölgede amaç fonksiyonunu maksimum yapan nokta $C\ (x_1 = 1,\ x_2 = 2)$ noktasıdır.

Kesme düzlemi algoritmasını simpleks tabloda daha kolay açıklanabilir. Aşağıda bir doğrusal programlama problemi ve problemin tamsayı sonuç vermeyen final tablosu verilmektedir.

$$enb\ddot{u}y\ddot{u}kle\ Z = 3\,x_1 + 5\,x_2 \qquad \textbf{(4.17)}$$

$$ba\breve{g}lı\ olarak\ \ x_1 + 4\,x_2 \leq 9 \qquad \textbf{(4.17a)}$$

$$2\,x_1 + 3\,x_2 \leq 11 \qquad \textbf{(4.17b)}$$

Çizelge 4.1 : Simpleks Çizelgesi.

amaç fonk.katsa.			3	5	0	0
	taban değişk.	kapasite	x1	x2	*S1*	*S2*
5	*x2*	7/5	0	1	2/5	-1/5
3	*x1*	17/5	1	0	-3/5	4/5
	Zj	86/5	3	5	1/5	7/5
	Cj - Z j		0	0	-1/5	-7/5

Yeni kısıt eklemek için eniyilenmiş çözümdeki tamsayı olmayan herhangi bir değişken seçilebilir. x_2 değişkenini seçip bu değişkenin olduğu satırı tekrar yazarsak, tam sayı olmayan sayılar;

$$(tamsayı) + (1den\ küçük\ pozitif\ kesir) \quad \textbf{(4.18)}$$

Şeklinde tekrar yazılabilir. Örneğin;

- 4/3 ------→ 1 + 1 / 3
- 5/4------→ 1 + 1 / 4
- 2/3------→ 0 + 2 / 3
- -2/3 ------→ -1 + 1 / 3

$x_2$değişkeni (4.18)'e göre yazılırsa:

$$x_2\ (\ 1, \frac{2}{5}, -\frac{1}{5}, \frac{7}{5}) \quad \textbf{(4.19)}$$

$$(1+0)x_2 + \left(0+\frac{2}{5}\right)S1 + \left(-1+\frac{4}{5}\right)S2 = (1+\frac{2}{5}) \quad \textbf{(4.19a)}$$

Daha sonra tamsayılı katsayıları sağ tarafa alarak yeniden yazılırsa:

$$\frac{2}{5}\ S1\ + \frac{4}{5}\ S2\ = \frac{2}{5} + (1 -\ 1\ x_2\ +\ 1S2) \quad \textbf{(4.20)}$$

olur. Buradan tam sayılı kısım herhangi bir tamsayı olarak düşünülüp eşitlikten çıkartılırsa:

$$\frac{2}{5}\ S1\ + \frac{4}{5}\ S2\ >= \frac{2}{5} \quad \textbf{(4.21)}$$

yazılabilir.

Probleme yapay(artificial) değişken eklememek için her iki tarafı − 1 ile çarparak eşitliğin yönünü değiştirip bir slack(boş) değişken eklenirse kısıt aşağıdaki şekli alacaktır.

$$-\frac{2}{5}\ S1\ -\frac{4}{5}\ S2\ +\ S3\ =\ -\frac{2}{5} \quad \textbf{(4.22)}$$

(4.22) kısıtı son çizelgeye eklenirse;

Çizelge 4.2 : 2. Simpleks Çizelgesi.

amaç fonksiyonu katsayısı			3	5	0	0	0
	taban değişk.	kapasite	x1	x2	*S1*	*S2*	*S3*
5	*x2*	7/5	0	1	2/5	-1/5	0
3	*x1*	17/5	1	0	-3/5	4/5	0
0	*S3*	-2/5	0	0	-2/5	-4/5	1
	Zj	86/5	3	5	1/5	7/5	0
	Cj - Z j		0	0	-1/5	-7/5	0

Bundan sonraki adımda *S3* tabandan çıkacak ve yerine başka bir değişken tabana girecektir. Tabana girecek değişkenin seçimi için *Cj - Z j* satırındaki negatif elemanlar bunlara karşı gelen S3 satırındaki negatif katsayılarla oranlanır. En küçük orana sahip sütundaki değişken tabana girecek değişkendir.

(- 1/5) / (-2/5) = 1 / 2 ve (-7/5) / (-4/5) = 7/4 oranlarına bakarsak (1/2) en küçük değer olduğu için bu sütundaki *S1* değişkeni, *S3* yerine tabana girecek değişkendir.

Bir sonraki simpleks tablo şu şekilde olur:

Çizelge 4.3 : 3. Simpleks Çizelgesi.

amaç fonk.katsa.			3	5	0	0	0
	taban değişk.	kapasite	x1	x2	*S1*	*S2*	*S3*
5	*x2*	1	0	1	0	-1	1
3	*x1*	4	1	0	0	2	-3/2
0	*S1*	1	0	0	1	2	-5/2
	Zj	17	3	5	0	1	1/2
	Cj - Z j		0	0	0	-1	-1/2

Bu tablodaki sonuçlara bakarsak; $x_1 = 4$, $x_2 = 1$ ve $Z = 17$ bulunur.

Tamsayılı sonuç elde edildiği için iterasyona son verilir. Ele alınan örnekte tek kısıt ilavesi ile eniyilenmiş sonuca ulaşılmaktadır. Bu her zaman gerçekleşmeyebilir.

Eklenen ilk kısıttan sonra elde edilen sonuç hala tamsayı değilse yeniden bir kısıt daha eklenerek tam sayılı sonuç alınıncaya kadar işlemler tekrarlanır.

Gomory kesme düzlemi yöntemini 60'lı yıllarda ortaya koymasına rağmen algoritma neredeyse 30 sene hiç kullanılmamıştır. Konudaki pek çok uzman, Gomory'de dahil, bu algoritmanın pek pratik olmadığını, sayısal tutarsızlıklar sebebiyle pek çok yuvarlama işlemi yapılması gerektiğini söylemişlerdir. Ancak 90'lı yılların ortasında Cornuejols dal ve kesme algoritmasında dal ve sınır algoritması ile kesme düzlemi yöntemini birleştirmiş ve sayısal tutarsızlıkların önüne geçmeyi başarmıştır. Şu anda pek çok ticari eniyileme programı bir şekilde kesme düzlemi yöntemini kullanmaktadır.[4]

Gomory kesme düzlemi yönteminin yanında pek çok farklı kesme düzlemi yöntemi geliştirilmiştir. Ancak Gomory tarafından geliştirilen kesme düzlemi yöntemi simpleks tabloları gösterimi için çok uygundur. Diğer kesme düzlemi yöntemleri ya çok zor yakınsar yada olurlu çözüm bulamaz.[10]

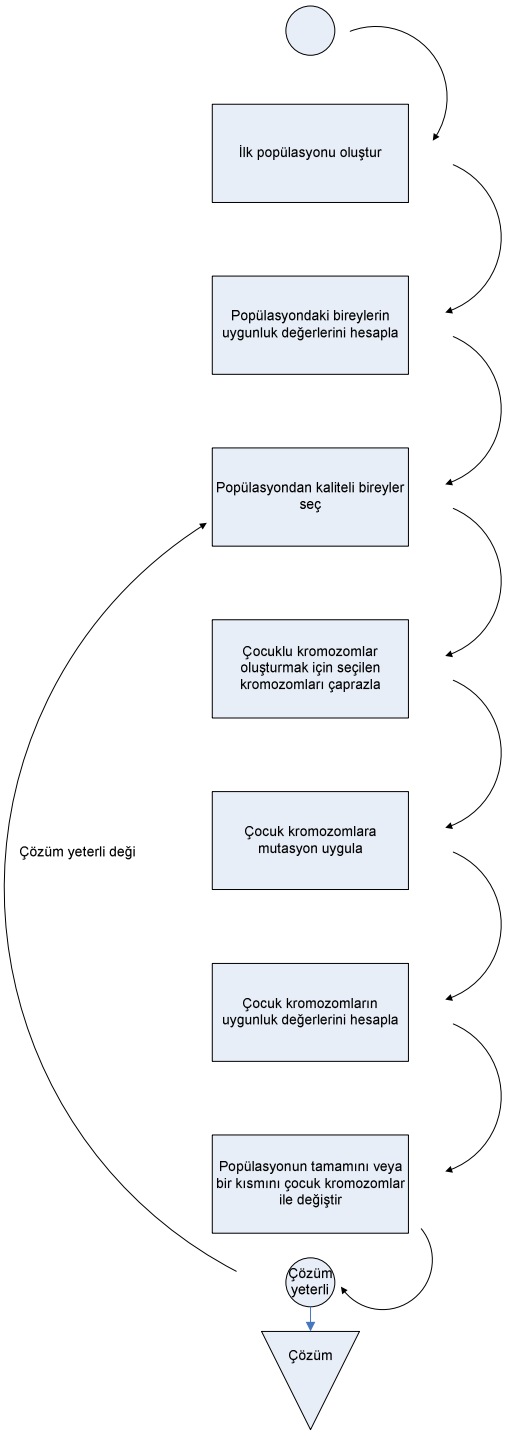
4.4 Genetik Algoritmalar

Genetik algoritmalar, bir çok değişik eniyileme problemine uyarlanabilen, olasılıklara dayanarak çalışan zeki arama-tarama algoritmaları olarak tanımlanabilirler. Teorik temelleri Holland tarafından ortaya atılan genetik algoritmalar doğadaki biyolojik organizmaların, geçirdikleri evrimsel süreçlerden esinlenilerek geliştirilmiştir. Evrimsel süreç ilerledikçe, popülasyon doğal seleksiyon prensibine göre evrimleşmektedir. Popülasyon içinde, diğerlerine nazaran çevrelerine daha iyi adapte olmuş olan bireyler daha yüksek yaşama ve çoğalma şansına sahiptirler ve daha sağlıksız bireyler elenirler. Bu şekilde evrimsel süreç boyunca sağlıklı olan bireylerin genleri nesiller boyunca bir sonraki nesile aktarılır. Sağlıklı bireylerin gen kombinasyonları, kendilerinden daha sağlıklı yeni bireyler oluşmasını sağlayabilmekte ve bu şekilde evrim süreci boyunca popülasyonda daha sağlıklı bireyler oluşturulabilmektedir. [11]

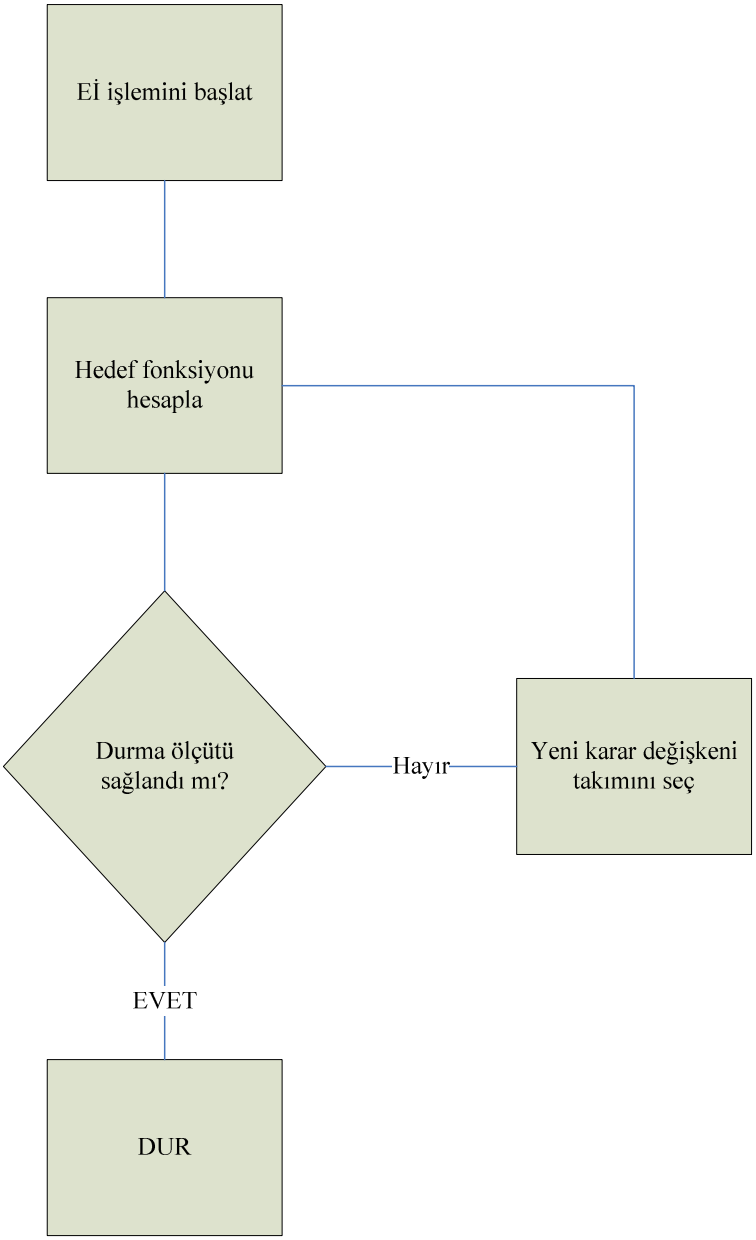
Genetik algoritmalar rastgele üretilen ilk nesil ve ondan sonra gelen tüm nesiller üzerinde genetik operatörleri uygulayarak bu evrimsel eniyileme sürecini simüle eder. Popülasyondaki her birey, problem için mümkün bir çözümü ifade etmektedir ve probleme özgü bir şekilde kodlanarak kromozomlarla temsil edilirler. Her

kromozomun(bireyin) çözüm kalitesi, kullanılan amaç fonksiyonu ile hesaplanır. Yüksek kalitede olan bireyler, genlerindeki bilgiyi, diğer yüksek kalitedeki bireyler ile çaprazlama operatörü vasıtasıyla üreyerek yeni nesillere aktarırlar. Bu şekilde, ebeveynlerin genleri değişik kombinasyonlarda çocuk nesillere aktarılır ve sürekli yeni çözümler üretilir. Yerel maksimum ve yerel minimuma yakalanmamak için her nesilde, kromozom üzerindeki bazı genlerin değiştirilmesi yolu ile belirli bir oranda mutasyon operatörü uygulanır. Ve sonuç olarak yeni nesil ya tamamen eski nesil ile yer değiştirilir yada kalitesiz olan eski nesildeki bireyler ile yeni nesildeki bireyler yer değiştirilir. Bu döngü tatmin edici bir çözüm bulana kadar tekrarlanır. Temel genetik algoritma yaşam döngüsü Şekil (4.7)'de verilmiştir.

Pratik çalışmalarda en iyilenmiş çözümün bulunması kesin şekilde başarılamayabilir. Bir çok durumda en iyi çözüme en yakın olan çözümlerle yetinilmek zorunda kalınabilir. Bunun nedenleri arasında zaman ve işlem hacminin çok olması, sorunun karar değişkenlerinin fazla ve karmaşık ilişkiler halinde bulunması gelebilir. Tüm en iyileme çözümlemelerinde biri karar değişkeni veya çözüm uzayı alanını tarif eden değişkenler, diğeri de bu değişkenlerin bir fonksiyonu olan hedef değişkeni bulunur. Bunlara ilave olarak karar değişkenlerini fizik ve matematik olarak sınırlayıcı şartlar da bulunabilir. En iyileme karar değişkenlerinin bir takımından başlayarak, önce buna karşı gelen hedef değerlerinin hesaplanması, daha sonra da bu hedef değişkenleri optimum noktaya yaklaştıracak biçimde ardışık karar değişkenleri yenilenmesi ile hedef değeri yeniden belirleyerek yola devam edilir. Bu şekilde güzergahı önceden belli olmayan bir yol boyunca karar uzayında seyahat edilir. Her yolun veya seyahatin bir sonu olduğuna göre en iyileme probleminin de bir şekilde sona ermesi gerekmektedir. Mutlak son en iyi noktaya %100 varmakla olur ama bu teorik bir beklentiden başka bir şey değildir. Bu nedenle, en iyileme yolculuğunun durması için bir ölçütün ortaya konması gerekir. Bu ölçüt ya adım sayılarının sınırlanması, ya uzman görüşüyle bulanık biçimde karar vermek yada objektif olarak ardışık ilerlemelerdeki hedef değerleri arasındaki farkın önceden belirlenen bir miktar veya yüzdeden daha küçük olmasının istenmesidir. Bütün bu söylenenlerin göz önünde tutulması halinde genel olarak bir en iyileme çalışmasında işlemlerin mantık akışı Şekil (4.8)'de gösterilmiştir.[11]



Şekil 4.7 : Genetik algoritma yaşam döngüsü.



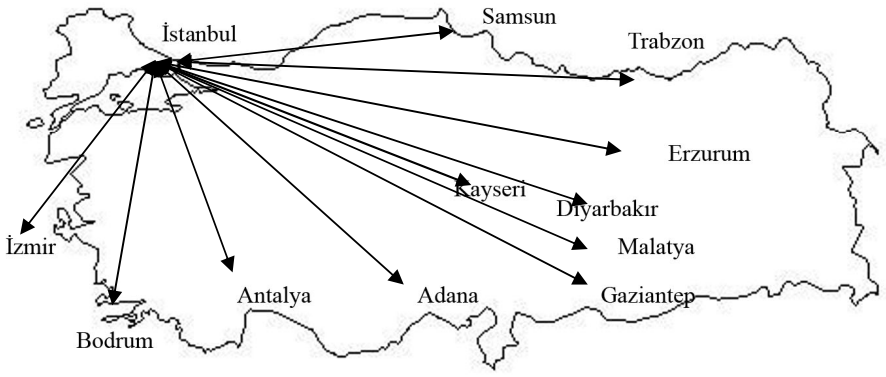
Şekil 4.8 : En iyileme akış diyagramı.

5. ÖRNEK MODEL

Filo ataması problemindeki en büyük sorun belirli zamanlarda belirli havaalanlarında hangi tip uçakların bulunduğunun takibinin zorluğudur. Bu problemi aşmak için Hane et al. tarafından 1995 yılında teklif edilen zaman-uzay ağı gösterimi oldukça uygundur. Dolayısıyla literatürde pek çok farklı araştırma olsa da halen 1995'te Hane tarafından ortaya konulan çalışma baz alınır. Bu çalışmada da Hane et al. tarafından teklif edilen filo ataması modeli baz alınarak farklı algoritmaların filo ataması probleminin çözümünde ne kadar etkili olacağı araştırılacaktır.

İşlem yükü değişken sayısının artması ile katlanarak arttığından ve kısıt fonksiyonları el ile girilmek zorunda olduğundan örnek model filo ataması modelinin özelliklerini yok etmeyecek şekilde seçilmiştir.

Öne sürülen bu modelde dokuz MD-83, sekiz A321, altı A300-600 ve iki A300-B4-200 olmak üzere toplam 25 uçağa sahip ve günde Türkiye'deki büyük şehirlere seferler düzenleyen bir havayolu şirketi bulunmaktadır. Havayolunun uçuşları şu şekildedir:



Şekil 5.1 : Örnek model için uçuş noktaları.

Çizelge 5.2 : Örnek model için kalkış ve iniş saatleri.

Uçuş No	Kalkış		Varış	
101	İstanbul	06.45	Adana	08.15
102	İstanbul	06.45	Diyarbakır	08.30
103	İstanbul	06.50	Trabzon	08.25
104	İstanbul	07.15	Kayseri	08.30
105	Antalya	07.30	İstanbul	08.30
106	İzmir	07.30	İstanbul	08.25
107	İstanbul	07.45	İzmir	08.45
108	İstanbul	08.15	Antalya	09.15
109	Adana	09.10	İstanbul	10.40
110	Kayseri	09.15	İstanbul	10.30
111	Trabzon	09.20	İstanbul	10.55
112	İstanbul	09.30	Gaziantep	11.15
113	İstanbul	09.30	Samsun	10.45
114	Diyarbakır	09.30	İstanbul	11.15
115	Antalya	10.45	İstanbul	11.45
116	İzmir	10.50	İstanbul	11.45
117	Samsun	11.45	İstanbul	13.00
118	İstanbul	11.55	Bodrum	12.55
119	İstanbul	12.15	Malatya	13.45
120	Gaziantep	12.15	İstanbul	14.00
121	İstanbul	12.30	Erzurum	14.20
122	Bodrum	13.45	İstanbul	14.45
123	Malatya	14.30	İstanbul	16.00
124	İstanbul	15.00	Adana	16.30
125	Erzurum	15.20	İstanbul	17.10
126	İstanbul	16.30	İzmir	17.30
127	İstanbul	16.45	Antalya	17.45
128	Adana	17.30	İstanbul	19.00
129	İstanbul	18.45	Diyarbakır	20.30
130	İstanbul	18.45	Trabzon	20.20
131	Antalya	18.45	İstanbul	19.45
132	İstanbul	19.00	Adana	20.30
133	İstanbul	20.30	Antalya	21.30
134	İstanbul	20.45	İzmir	21.45
135	Trabzon	21.15	İstanbul	22.50
136	Adana	21.30	İstanbul	23.00
137	Diyarbakır	21.30	İstanbul	23.15
138	İzmir	18.50	İstanbul	19.45

5.1 İşletme Masrafı

Bir uçuş için işletme masrafı genel olarak o uçuşa atanan filo tipi ile alakalıdır ve şu şekilde hesaplanır:

$$Uçuş\ işletme\ masrafı = KKKM \times koltuk\ sayısı \times mesafe \quad \textbf{(5.1)}$$

Yukarıda belirtilen modelde dört filo tipi bulunmaktadır: MD-83, A321, A300-600 ve A300-B4-200. Bu uçakların koltuk sayıları sırasıyla 165, 220, 316 ve 317'dir. Bunlara ek olarak [1]'den alınan değerler ise şu şekildedir:

- KKKM Kullanılabilir koltuk kilometresi masrafı (Cost per available seat mile-CASM): MD-83, A321, A300-600 ve A300-B4-200 için sırasıyla 5.90¢, 3.00¢, 5.50¢ ve 5.50¢'dir.
- KKKG Kullanılabilir koltuk kilometresi geliri (Revenue per available seat mile-RASM): 15¢'tir.

Yukarıdaki bilgiler kullanılarak dört filo tipi için de her uçuş için işletme masrafı hesaplanabilir. Örnek olarak modeldeki 115 numaralı Antalya-İstanbul uçuşu için toplam mesafe 535 kilometredir. Dolayısıyla 115 numaralı uçuşta dört filo tipi için işletme masrafı şu şekildedir:

- $MD-83\ için\ işletme\ masrafı = 0.059\$ \times 165 \times 535 = 5210\$$
- $A321\ için\ işletme\ masrafı = 0.030\$ \times 220 \times 535 = 3530\$$
- $A300-600\ için\ işletme\ masrafı = 0.055\$ \times 316 \times 535 = 9300\$$
- $A300-B4-200\ için\ işletme\ masrafı = 0.055\$ \times 317 \times 535 = 9330\$$

Diğer tüm uçuşlar için uçuş işletme masrafları Çizelge (5.2)'de görülebilir.

Çizelge 5.2 : Tüm uçuşlar için işletme masrafları.

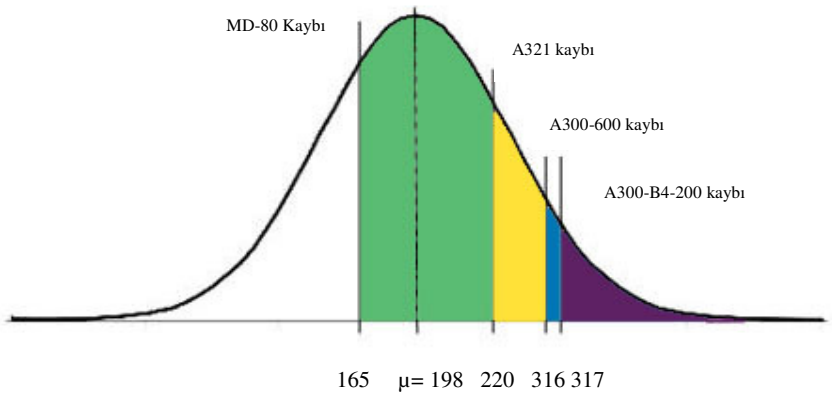
Uçuş #	Kalkış	Varış	Mesafe	1.Filo	2.Filo	3.Filo	4.Filo
101	İstanbul	Adana	824	12910	8752	22902	22280
102	İstanbul	Diyarbakır	1087	17030	11546	30211	29391
103	İstanbul	Trabzon	974	15260	10346	27071	26336
104	İstanbul	Kayseri	696	10904	7393	19344	18819
105	Antalya	İstanbul	535	8382	5683	14869	14466
106	İzmir	İstanbul	419	6564	4450	11645	11329
107	İstanbul	İzmir	419	6564	4450	11645	11329
108	İstanbul	Antalya	535	8382	5683	14869	14466
109	Adana	İstanbul	824	12910	8752	22902	22280
110	Kayseri	İstanbul	696	10904	7393	19344	18819
111	Trabzon	İstanbul	974	15260	10346	27071	26336
112	İstanbul	Gaziantep	983	15401	10441	27321	26579
113	İstanbul	Samsun	696	10904	7393	19344	18819
114	Diyarbakır	İstanbul	1087	17030	11546	30211	29391
115	Antalya	İstanbul	535	8382	5683	14869	14466
116	İzmir	İstanbul	419	6564	4450	11645	11329
117	Samsun	İstanbul	696	10904	7393	19344	18819
118	İstanbul	Bodrum	526	8241	5587	14619	14222
119	İstanbul	Malatya	861	13489	9145	23930	23280
120	Gaziantep	İstanbul	983	15401	10441	27321	26579
121	İstanbul	Erzurum	1091	17093	11588	30323	29499
122	Bodrum	İstanbul	526	8241	5587	14619	14222
123	Malatya	İstanbul	861	13489	9145	23930	23280
124	İstanbul	Adana	824	12910	8752	22902	22280
125	Erzurum	İstanbul	1091	17093	11588	30323	29499
126	İstanbul	İzmir	419	6564	4450	11645	11329
127	İstanbul	Antalya	535	8382	5683	14869	14466
128	Adana	İstanbul	824	12910	8752	22902	22280
129	İstanbul	Diyarbakır	1087	17030	11546	30211	29391
130	İstanbul	Trabzon	974	15260	10346	27071	26336
131	Antalya	İstanbul	535	8382	5683	14869	14466
132	İstanbul	Adana	824	12910	8752	22902	22280
133	İstanbul	Antalya	535	8382	5683	14869	14466
134	İstanbul	İzmir	419	6564	4450	11645	11329
135	Trabzon	İstanbul	974	15260	10346	27071	26336
136	Adana	İstanbul	824	12910	8752	22902	22280
137	Diyarbakır	İstanbul	1087	17030	11546	30211	29391

5.2 Yolcu Kayıp Masrafı

Filo atamasında önemli olan diğer bir konu ise belirli taleplere göre filo tiplerini uçuşlara atamaktır. Düşük talepli uçuşlara yüksek kapasiteli filo tiplerini atamak doluluk oranını düşürdüğü gibi uçuş masrafını da arttırır. Diğer yandan yüksek talepli uçuşlara düşük kapasiteli uçakları atamak da yolcu kaybına sebep olur. Yolcu kaybı ortalama talebin uçak kapasitesi ile farkıdır. Yolcu kayıp masrafı ise dolayısıyla, yetersiz yolcu kapasitesi sebebiyle kaybedilen yolculardan alınamayan ücretlerdir.[13]

Modelde kullanılan havayolu şirketleri için talep ve standart sapma değerleri bulunmadığından, bu değerler varsayımsal olarak üretilmiştir.

Modeldeki 115 numaralı Antalya-İstanbul uçuşuna bakılacak olursak, üretilen talep ve standart sapma değerlerinin sırasıyla 198 ve 38 olduğu görülür. Şekil (5.2) bu uçuş için talep dağılımını göstermektedir. Renkli alanlar olası yolcu kayıplarını göstermektedir. Yolcu kaybı basitçe talep dağılımından uçak kapasitesi kısmının çıkartılmasıdır.



Şekil 5.2 : Yolcu kayıp normal dağılımı.

Beklenen yolcu kayıp masrafı şu şekilde hesaplanır:

$$Yolcu\ kayıp\ masrafı = yolcu\ kaybı \times KKKG \times mesafe \quad \textbf{(5.2)}$$

Beklenen yolcu kaybı ise şu şekilde hesaplanır:

$$Beklenen\ yolcu\ kaybı\ sayısı = \int_{c}^{\infty} (x - c) f(x) dx \quad \textbf{(5.3)}$$

Yukarıdaki denklemde "c" filo kapasitesini, $f(x)$ talebin olasılık dağılım fonksiyonunu temsil eder.

Beklenen yolcu kayıplarını hesaplamak için MATLAB'da bir kod yazılmıştır. Daha sonra beklenen yolcu kayıp sayılarından beklenen yolcu kayıp masrafları hesaplanmıştır. Formül (5.2) ve (5.3)'e göre, örnek model için kurgulanan standart sapma, ortalama talep gibi veriler herhangi bir şirketten ticari bilgi olduğu için alınamadığından, bu veriler rastgele bir algoritma ile üretilmiştir. Bunlara ek olarak, veriler bağlantılı olarak kullanıldıkları şehire göre tahmini olarak hesaplanmıştır. Örneğin Adana, Antalya gibi büyük şehirler için talep oranları yüksek iken, Erzurum gibi küçük şehirler için düşük olarak belirlenmiştir. Aynı şekilde sabah ve akşam uçuşlarına talep yüksek olarak belirlenirken gün içindeki uçuşlarda talep düşük olarak belirlenmiştir.

Çizelge 5.3 : Tüm uçuşlar için yolcu kaybı masrafları

Uçuş No	MD-80 beklenen yolcu kaybı masrafı	A321 beklenen yolcu kaybı masrafı	A300-600 beklenen yolcu kaybı masrafı	A300-B4-200 beklenen yolcu kaybı masrafı
101	102.887.876,00	43.762.025,00	3.201.584,00	1.883.442,00
102	126.321.194,00	4.324.417,00	523.652,00	1.352.472,00
103	4.441.797,00	-	-	-
104	2.693.435,00	-	-	-
105	2.628.635,00	2.857.012,00	-	-
106	-	-	-	-
107	123.503,00	-	-	-
108	104.861.663,00	59.046.121,00	5.399.762,00	7.672.168,00
109	7.363.034,00	8.558.904,00	123.341,00	180.276,00
110	3.914.691,00	131.062,00	-	-
111	1.235.913,00	399.611,00	-	-
112	127.755.142,00	55.255.396,00	1.190.061,00	2.291.546,00
113	-	-	-	-
114	148.531.047,00	61.574.605,00	2.592.334,00	4.246.595,00
115	28.554.223,00	3.744.192,00	-	-
116	9.116,00	-	-	-
117	-	-	-	-
118	4.910.632,00	133.127,00	-	-
119	122.875.517,00	60.225.018,00	935.959,00	1.138.502,00
120	143.780.701,00	64.018.167,00	2.298.869,00	5.837.367,00
121	55.310.755,00	12.771.233,00	-	-
122	3.551.621,00	-	-	-
123	83.867.518,00	29.644.637,00	165.785,00	87.424,00
124	142.967.719,00	78.052.559,00	7.872.775,00	10.423.965,00
125	36.815.468,00	5.054.851,00	-	-
126	1.701.764,00	-	-	-
127	44.540.029,00	11.230.352,00	158.616,00	220.549,00
128	150.500.648,00	85.895.404,00	11.440.721,00	10.687.118,00
129	17.535.046,00	106.307.738,00	4.785.359,00	3.464.214,00
130	22.433.329,00	1.157.781,00	-	-
131	38.151.516,00	9.522.065,00	-	-
132	161.370.734,00	101.193.767,00	1.498.193,00	17.883.111,00
133	62.686.597,00	22.124.125,00	93.358,00	406.097,00
134	21.599.761,00	4.182.653,00	-	-
135	64.601.872,00	13.345.641,00	-	-
136	171.990.328,00	112.134.871,00	16.833.985,00	16.985.087,00
137	258.939.653,00	17.910.631,00	49.512.343,00	46.711.778,00
138	-	-	-	-

5.3 Tekrar Alım Oranı

Yolcu kaybı masrafına ek olarak göz önünde bulundurulması gereken bir diğer konu da tekrar alım oranıdır. Tekrar alım oranı, kaybedilmiş yolcuların aynı havayolu şirketnin başka uçuşlarını kullanma oranıdır. Eğer bir yolcu istediği uçuşa yer bulamazsa, havayolu şirketi belirli şartlarla ve indirimlerle yolcuya diğer uçuşları kullanmasını önerebilir. Yolcu önerilen uçuşu kabul ederse, yolcu kaybedilmiş sayılmaz. Büyük havayolu şirketleri için tekrar alım oranı çok yüksektir. Bunun sebebi yüksek sıklıkla yapılan uçuşlardır.[13]

Modeldeki havayolu şirketi için uçuşlar pek sık olmadığından tekrar alım oranı %15 olarak belirlenmiştir. Yani kaybedilen yolcuların %85'inin diğer havayolu şirketlerini tercih ettiği varsayılmıştır.

112 numaralı Antalya İstanbul uçuşu için beklenen yolcu kaybı masrafı şu şekilde olur:

- $MD\ 83\ beklenen\ yolcu\ kaybı\ masrafı = 127755\$ \times \%.85 = 108591\$$
- $A321\ beklenen\ yolcu\ kayıp\ masrafı = 55255\$ \times \%.85 = 46966\$$
- $A300-600\ beklenen\ yolcu\ kayıp\ masrafı = 1190\$ \times \%.85 = 1011\$$
- $A300B4200\ beklenen\ yolcu\ kayıp\ masrafı = 2291\$ \times \%.85 = 1947\$$

Beklenen yolcu kaybı masrafları da hesaplandıktan sonra toplam masraf şu şekilde hesaplanır:

$$\sum masraf = işletme\ masrafı + yolcu\ kaybı\ masrafı \quad \textbf{(5.4)}$$

Tüm uçuşlar için hesaplanan toplam masraflar şu şekildedir:

Çizelge 5.4 : Tüm uçuşlar için toplam masraflar

Uçuş No	MD-80	A321	A300-B4	A300-600
101	22927,69173	13355,24813	23041,28491	22414,12106
102	29405,65336	16160,67496	30298,92197	29451,7919
103	15605,20409	10345,50697	27070,74324	26335,58524
104	11051,9228	7392,682598	19344,18613	18818,85763
105	10350,90877	6103,029941	14869,45342	14465,64487
106	6564,457849	4450,479898	11645,4224	11329,1686
107	6584,592947	4450,479898	11645,4224	11329,1686
108	18531,16282	11929,93085	15653,29388	15308,53209
109	19503,7663	10341,81323	22907,76648	22288,43537
110	11173,31551	7392,682598	19344,18613	18818,85763
111	16786,83248	10421,00538	27070,74324	26335,58524
112	27344,92528	15361,29521	27545,35405	26691,87942
113	10904,20683	7392,682598	19344,18613	18818,85763
114	32550,94791	17835,17524	30823,70221	29737,72744
115	10929,55775	6267,379564	14869,45342	14465,64487
116	6566,392133	4450,479898	11645,4224	11329,1686
117	10905,07356	7392,682598	19344,18613	18818,85763
118	8662,380867	5597,240569	14619,31308	14222,29757
119	25720,36264	14419,03924	24061,78656	23408,42199
120	28992,22609	18010,28034	27514,78769	26770,51016
121	22354,72058	12698,25889	30322,56763	29512,32633
122	8560,384917	5586,99863	14619,31308	14222,29757
123	22460,67875	12036,71295	23930,09233	23287,44397
124	26921,38634	16482,14323	23547,46922	22896,41971
125	20766,87081	11983,12734	30322,56763	29499,1001
126	6708,215302	4450,479898	11645,4224	11329,1686
127	12653,21807	6525,753916	14869,45342	14489,49339
128	28039,18069	16946,64752	24126,18485	23428,40289
129	34730,70627	20195,11606	31208,07257	29752,24057
130	17486,05078	10417,79737	27070,74324	26335,58524
131	12065,89252	6818,983871	14869,45342	14465,64487
132	29150,13884	19046,20892	24768,69898	23901,06074
133	15307,01684	7852,449975	14880,47174	14510,60797
134	8730,48312	4796,767029	11645,4224	11329,1686
135	22486,12812	12234,27053	27070,74324	26335,58524
136	29749,93705	18998,22962	24635,44915	24336,28667
137	41798,27105	25467,63775	33782,3316	32428,11733

5.4 Amaç Fonksiyonu

Örnek model için amaç fonksiyonunu kurmak için öncelikli olarak hangi filo tipinin hangi uçağa atandığını belli edecek karar değişkenleri belirlenmelidir. Filo atama problemi için [5] referansında teklif edilen karar değişkenleri şu şekildedir:

$$x_{i,j} = \begin{cases} 1, eğer\ i\ uçuşu\ f\ filo\ tipine\ atandıysa \\ 0, aksi\ halde \end{cases} \quad \textbf{(5.5)}$$

$$G_{k,j} = k\ \ noktasında, j\ filo\ tipi\ için\ yerde\ bulunan\ uçak\ sayısı \quad \textbf{(5.6)}$$

İkili karar değişkeni $x_{i,j}$'de i indisi uçuşları(örnek model için 38 adet), j indisi ise filo tipini temsil eder(örnek model için dört adet). İleriki aşamalarda kolaylık sağlaması için filo tiplerine şu indisler verilmiştir:

- MD-83 için "1"
- A321 için "2"
- A300-600 için "3"
- A300-B4-200 için "4"

Bu bilgiye göre $x_{101,1}$ ikili karar değişkeni "101" numaralı uçuşa "1"indisli filo tipinin atandığını gösterir. $x_{101,2}$ ikili karar değişkeni ise "101" numaralı uçuşa "2" indisli filo tipinin atandığını gösterir. $G_{k,j}$ tam sayı karar değişkeni ise uçak denge kısıt fonksiyonları belirtilirken kullanılacaktır.

Amaç fonksiyonu kısaca uçuşlara en uygun filo tiplerinin atanması ile toplam maliyetin en küçüklenmesidir:

$$en\ küçükle\ 22927{,}69173\ \mathrm{x}_{101,1} + 13355x_{101,2} + 23041x_{101,3} + 22414x_{101,4} + \cdots + 8071{,}103055\mathrm{x}_{138,1} + 4668{,}352688\mathrm{x}_{138,2} + 11645{,}4224\mathrm{x}_{138,3} + 11329{,}1686\mathrm{x}_{138,4}$$

5.5 Kısıt Fonksiyonları

[5] referansında kullanılan filo ataması modelinde üç adet kısıt bulunmaktadır ve matematiksel olarak şu şekilde ifade edilirler:

$$\sum_{f \in F} x_{fl} = 1, \quad \forall \in L, \quad \textbf{(5.7)}$$

$$\sum_{o \in S} x_{fost} + y_{fst^- t} \sum_{d \in S} x_{fsdt} - y_{fstt^+} = 0, \quad \forall \{fst\} \in N, \quad \textbf{(5.8)}$$

$$\sum_{l \in O(f)} x_{fl} + \sum_{s \in S} y_{fst_n t_1} \leq A_f, \quad \forall \in F, \quad \textbf{(5.9)}$$

Bu kısıtlardan (5.7) kapsama, (5.8) denge ve (5.9) ise müsaitlik kısıt fonksiyonlarıdır. Modelin çözümü için amaç fonksiyonu belirlendikten sonra yapılması gereken kısıt fonksiyonlarını sırası ile belirlemektir.

5.5.1 Uçuş kapsama kısıtları

Hane et al tarafından teklif edilen ilk kısıt kümesi uçuş kapsama kısıtları kümesidir. Uçuş kapsama kısıtlarının amacı her uçuşun mutlaka bir filo tipi tarafından uçulmasını sağlamaktır. Bir uçuşun uçulduğundan emin olmak için uçuşu temsil eden tüm karar değişkenlerinin toplamının bire eşit olması gerekir. Böylelikle aynı uçuşa birden fazla uçak atanamadığı gibi uçuşa bir uçağın mutlaka atanması sağlanır. Uçuş kapsama kısıtı aşağıdaki formülle gösterilir:

$$\sum_{f \in F} x_{fl} = 1, \quad \forall l \in L \tag{5.10}$$

Örnek modeldeki 101 numaralı İstanbul-Adana uçuşu için kapsama kısıt fonksiyonu şu şekilde yazılır:

$$x_{101,1} + x_{101,2} + x_{101,3} + x_{101,4} = 1 \tag{5.11}$$

Bu kısıt 112 numaralı uçuşun kapsanmasını sağlar yani bir filo tipi mutlaka bu uçuşu yapacaktır. Buna ek olarak 112 numaralı uçuşa birden fazla filo tipi atanamaz zira ikili karar değişkenlerinden sadece biri "1" değerini alabilir ve diğer karar değişkenleri sıfır değerini almak zorunda kalır. Örnek modeldeki tüm kapsama kısıt fonksiyonları şu şekildedir:

$$x_{101,1} + x_{101,2} + x_{101,3} + x_{101,4} = 1 \tag{5.12}$$

$$x_{102,1} + x_{102,2} + x_{102,3} + x_{102,4} = 1 \tag{5.13}$$

$$x_{103,1} + x_{103,2} + x_{103,3} + x_{103,4} = 1 \tag{5.14}$$

$$x_{104,1} + x_{104,2} + x_{104,3} + x_{104,4} = 1 \tag{5.15}$$

$$x_{105,1} + x_{105,2} + x_{105,3} + x_{105,4} = 1 \tag{5.16}$$

$$x_{106,1} + x_{106,2} + x_{106,3} + x_{106,4} = 1 \quad \textbf{(5.17)}$$

$$x_{107,1} + x_{107,2} + x_{107,3} + x_{107,4} = 1 \quad \textbf{(5.18)}$$

$$x_{108,1} + x_{108,2} + x_{108,3} + x_{108,4} = 1 \quad \textbf{(5.19)}$$

$$x_{109,1} + x_{109,2} + x_{109,3} + x_{109,4} = 1 \quad \textbf{(5.20)}$$

$$x_{110,1} + x_{110,2} + x_{110,3} + x_{110,4} = 1 \quad \textbf{(5.21)}$$

$$x_{111,1} + x_{111,2} + x_{111,3} + x_{111,4} = 1 \quad \textbf{(5.22)}$$

$$x_{112,1} + x_{112,2} + x_{112,3} + x_{112,4} = 1 \quad \textbf{(5.23)}$$

$$x_{113,1} + x_{113,2} + x_{113,3} + x_{113,4} = 1 \quad \textbf{(5.24)}$$

$$x_{114,1} + x_{114,2} + x_{114,3} + x_{114,4} = 1 \quad \textbf{(5.25)}$$

$$x_{115,1} + x_{115,2} + x_{115,3} + x_{115,4} = 1 \quad \textbf{(5.26)}$$

$$x_{116,1} + x_{116,2} + x_{116,3} + x_{116,4} = 1 \quad \textbf{(5.27)}$$

$$x_{117,1} + x_{117,2} + x_{117,3} + x_{117,4} = 1 \quad \textbf{(5.28)}$$

$$x_{118,1} + x_{118,2} + x_{118,3} + x_{118,4} = 1 \quad \textbf{(5.29)}$$

$$x_{119,1} + x_{119,2} + x_{119,3} + x_{119,4} = 1 \quad \textbf{(5.30)}$$

$$x_{120,1} + x_{120,2} + x_{120,3} + x_{120,4} = 1 \quad \textbf{(5.31)}$$

$$x_{121,1} + x_{121,2} + x_{121,3} + x_{121,4} = 1 \quad \textbf{(5.32)}$$

$$x_{122,1} + x_{122,2} + x_{122,3} + x_{122,4} = 1 \quad \textbf{(5.33)}$$

$$x_{123,1} + x_{123,2} + x_{123,3} + x_{123,4} = 1 \quad \textbf{(5.34)}$$

$$x_{124,1} + x_{124,2} + x_{124,3} + x_{124,4} = 1 \quad \textbf{(5.35)}$$

$$x_{125,1} + x_{125,2} + x_{125,3} + x_{125,4} = 1 \quad \textbf{(5.36)}$$

$$x_{126,1} + x_{126,2} + x_{126,3} + x_{126,4} = 1 \quad \textbf{(5.37)}$$

$$x_{127,1} + x_{127,2} + x_{127,3} + x_{127,4} = 1 \quad \textbf{(5.38)}$$

$$x_{128,1} + x_{128,2} + x_{128,3} + x_{128,4} = 1 \quad \textbf{(5.39)}$$

$$x_{129,1} + x_{129,2} + x_{129,3} + x_{129,4} = 1 \quad \textbf{(5.40)}$$

$$x_{130,1} + x_{130,2} + x_{130,3} + x_{130,4} = 1 \quad \textbf{(5.41)}$$

$$x_{131,1} + x_{131,2} + x_{131,3} + x_{131,4} = 1 \quad \textbf{(5.42)}$$

$$x_{132,1} + x_{132,2} + x_{132,3} + x_{132,4} = 1 \quad \textbf{(5.43)}$$

$$x_{133,1} + x_{133,2} + x_{133,3} + x_{133,4} = 1 \quad \textbf{(5.44)}$$

$$x_{134,1} + x_{134,2} + x_{134,3} + x_{134,4} = 1 \quad \textbf{(5.45)}$$

$$x_{135,1} + x_{135,2} + x_{135,3} + x_{135,4} = 1 \quad \textbf{(5.46)}$$

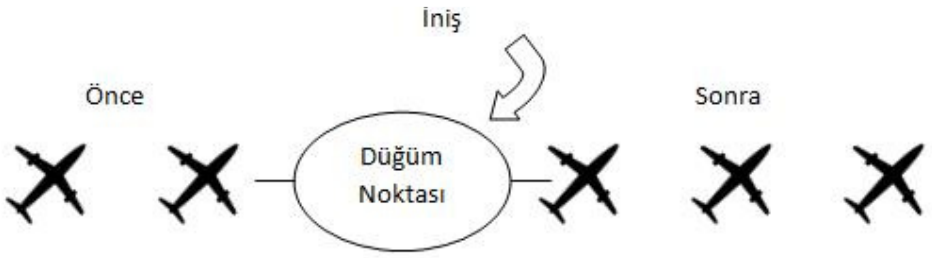
$$x_{136,1} + x_{136,2} + x_{136,3} + x_{136,4} = 1 \quad \textbf{(5.47)}$$

$$x_{137,1} + x_{137,2} + x_{137,3} + x_{137,4} = 1 \quad \textbf{(5.48)}$$

$$x_{138,1} + x_{138,2} + x_{138,3} + x_{138,4} = 1 \quad \textbf{(5.49)}$$

5.5.2 Uçuş denge kısıtları

Uçak denge kısıtları uçakların zaman uzay çizelgesindeki hareketlerinin sürekliliğini sağlar. Denge kısıtları sayesinde hangi uçağın hangi havaalanında hangi saatte bulunacağı belirlenebildiği gibi, doğru filo tipinin doğru zamanda doğru yerde olması da sağlanır. Denge kısıtları şu bir şekil ile açıklanabilir:



Şekil 5.3 : Düğüm noktası dengesi.

[5] tarafından öne sürülen modelde düğüm noktaları bir iniş yada kalkışı gösterir. Yukarıdaki şekil bir iniş düğüm noktasını göstermektedir. İnişten önce havaalanında aynı uçak tipinden iki uçak bulunmaktadır. İnişten sonra havaalanında aynı uçak tipinden üç adet uçak bulunur. Şekile göre uçak denge kısıtları için şu söylenebilir:

Düğüm noktasında yerde bulunan bir filo tipi için uçak sayısı=Düğüm noktasından hemen önce bir filo tipi için yerde bulunan uçak sayısı+aynı filo tipinden bir uçağın düğüm noktasında inişi-düğüm noktasında aynı filo tipinden bir uçağın kalkışı

Örnek olarak Şekil (5.3) ne için denge kısıtları şu şekildedir:

Düğüm noktasındaki uçak sayısı=2(düğüm noktasından önce yerde bulunan uçak sayısı)+1(bir iniş)+0(düğüm noktasından kalkış yok)=3

Kullanılan örnek model için denge kısıtları her şehir için tek tek şekillendirilmiş ve yazılmıştır.

5.5.2.1 İstanbul için denge kısıtları

İstanbul için zaman uzay ağı şu şekildedir:

Şekil 5.4 : İstanbul için zaman uzay ağı

Örnek modelde dört adet filo tipi bulunduğundan $G_{k,j}$ karar değişkeni her filo tipi için ayrı ayrı denge kısıt fonskiyonu yazmayı gerektirmektedir. İstanbul'da 1. Filo tipi için denge kısıt fonksiyonları şu şekildedir:

$$G_{L1,1} = G_{L38,1} - x_{101,1} \quad \textbf{(5.50)}$$

$$G_{L2,1} = G_{L1,1} - x_{102,1} \quad \textbf{(5.51)}$$

$$G_{L3,1} = G_{L2,1} - x_{103,1} \quad \textbf{(5.52)}$$

$$G_{L4,1} = G_{L3,1} - x_{104,1} \quad \textbf{(5.53)}$$

$$G_{L5,1} = G_{L4,1} - x_{107,1} \quad \textbf{(5.54)}$$

$$G_{L6,1} = G_{L5,1} - x_{108,1} \quad \textbf{(5.55)}$$

$$G_{L7,1} = G_{L6,1} + x_{106,1} \quad \textbf{(5.56)}$$

$$G_{L8,1} = G_{L7,1} + x_{105,1} \quad \textbf{(5.57)}$$

$$G_{L9,1} = G_{L8,1} - x_{112,1} \quad \textbf{(5.58)}$$

$$G_{L10,1} = G_{L9,1} - x_{113,1} \quad \textbf{(5.59)}$$

$$G_{L11,1} = G_{L10,1} + x_{110,1} \quad \textbf{(5.60)}$$

$$G_{L12,1} = G_{L11,1} + x_{109,1} \quad \textbf{(5.61)}$$

$$G_{L13,1} = G_{L12,1} + x_{111,1} \quad \textbf{(5.62)}$$

$$G_{L14,1} = G_{L13,1} + x_{114,1} \quad \textbf{(5.63)}$$

$$G_{L15,1} = G_{L14,1} + x_{115,1} \quad \textbf{(5.64)}$$

$$G_{L16,1} = G_{L15,1} + x_{116,1} \quad \textbf{(5.65)}$$

$$G_{L17,1} = G_{L16,1} - x_{118,1} \quad \textbf{(5.66)}$$

$$G_{L18,1} = G_{L17,1} - x_{119,1} \quad \textbf{(5.67)}$$

$$G_{L19,1} = G_{L18,1} - x_{121,1} \quad \textbf{(5.68)}$$

$$G_{L20,1} = G_{L19,1} + x_{117,1} \quad \textbf{(5.69)}$$

$$G_{L21,1} = G_{L20,1} + x_{120,1} \quad \textbf{(5.70)}$$

$$G_{L22,1} = G_{L21,1} + x_{122,1} \quad \textbf{(5.71)}$$

$$G_{L23,1} = G_{L22,1} - x_{124,1} \quad \textbf{(5.72)}$$

$$G_{L24,1} = G_{L23,1} + x_{123,1} \quad \textbf{(5.73)}$$

$$G_{L25,1} = G_{L24,1} + x_{126,1} \quad \textbf{(5.74)}$$

$$G_{L26,1} = G_{L25,1} - x_{127,1} \quad \textbf{(5.75)}$$

$$G_{L27,1} = G_{L26,1} + x_{125,1} \quad \textbf{(5.76)}$$

$$G_{L28,1} = G_{L27,1} - x_{129,1} \quad \textbf{(5.77)}$$

$$G_{L29,1} = G_{L28,1} - x_{130,1} \quad \textbf{(5.78)}$$

$$G_{L30,1} = G_{L29,1} - x_{132,1} \quad \textbf{(5.79)}$$

$$G_{L31,1} = G_{L30,1} + x_{128,1} \quad \textbf{(5.80)}$$

$$G_{L32,1} = G_{L31,1} + x_{138,1} \quad \textbf{(5.81)}$$

$$G_{L33,1} = G_{L32,1} + x_{131,1} \quad \textbf{(5.82)}$$

$$G_{L34,1} = G_{L33,1} - x_{133,1} \quad \textbf{(5.83)}$$

$$G_{L35,1} = G_{L34,1} - x_{134,1} \quad \textbf{(5.84)}$$

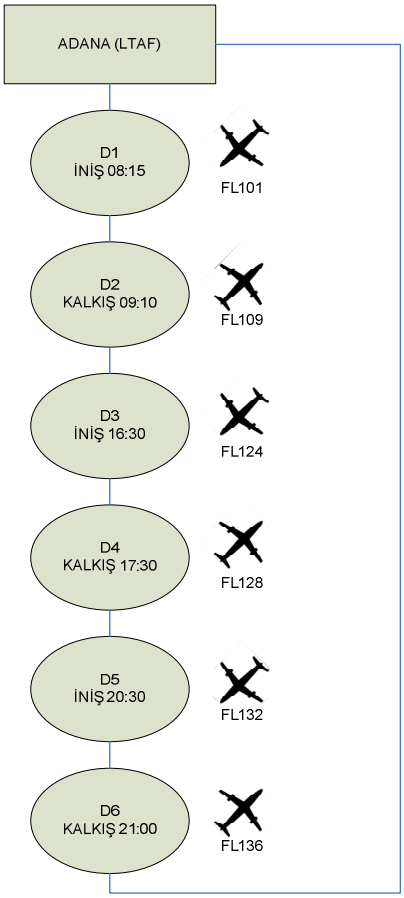
$$G_{L36,1} = G_{L35,1} + x_{135,1} \quad \textbf{(5.85)}$$

$$G_{L37,1} = G_{L36,1} + x_{136,1} \quad \textbf{(5.86)}$$

Aynı şekilde her filo tipi için denge fonksiyonları yazılır. Diğer filo tipleri için sadece alt indisler değiştiğinden yazılmamıştır.

5.5.2.2 Adana için denge kısıtları

Adana için zaman uzay ağı şu şekildedir:



Şekil 5.5 : Adana için zaman uzay ağı

Örnek modelde 4 adet filo tipi bulunduğundan $G_{k,j}$ karar değişkeni her filo tipi için ayrı ayrı denge kısıt fonskiyonu yazmayı gerektirmektedir. Adana için uçak denge kısıt fonksiyonları (1. Filo tipi için) şu şekilde yazılır:

$$G_{d1,1} = G_{d6,1} + x_{101,1} \quad \textbf{(5.87)}$$

$$G_{d2,1} = G_{d1,1} - x_{109,1} \quad \textbf{(5.88)}$$

$$G_{d3,1} = G_{d2,1} + x_{124,1} \quad \textbf{(5.89)}$$

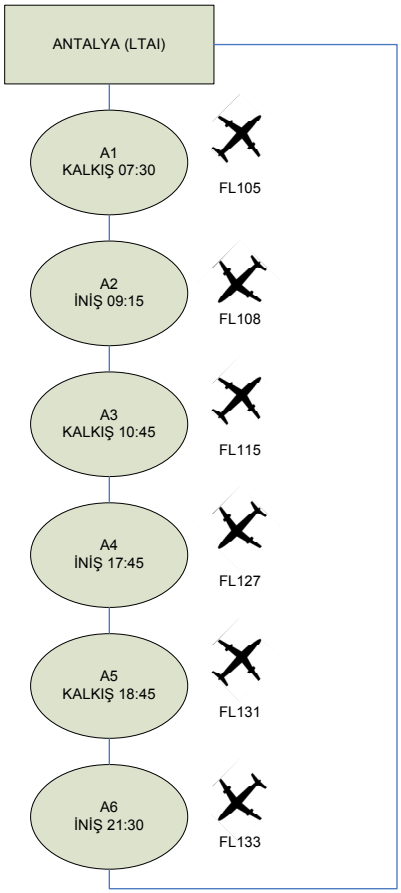
$$G_{d4,1} = G_{d3,1} - x_{128,1} \quad \textbf{(5.90)}$$

$$G_{d5,1} = G_{d4,1} + x_{132,1} \quad \textbf{(5.91)}$$

$$G_{d6,1} = G_{d5,1} - x_{136,1} \quad \textbf{(5.92)}$$

5.5.2.3 Antalya için denge kısıtları

Antalya için zaman uzay ağı şu şekildedir:



Şekil 5.6 : Antalya için zaman uzay ağı

Antalya için uçak denge kısıtları şu şekildedir:

$$G_{a1,1} = G_{a6,1} - x_{105,1} \quad \textbf{(5.93)}$$

$$G_{a2,1} = G_{a1,1} + x_{108,1} \quad \textbf{(5.94)}$$

$$G_{a3,1} = G_{a2,1} - x_{115,1} \quad \textbf{(5.95)}$$

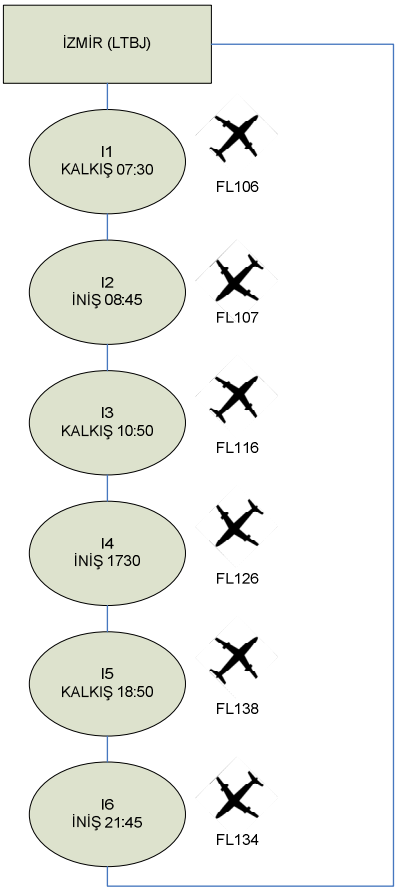
$$G_{a4,1} = G_{a3,1} + x_{127,1} \quad \textbf{(5.96)}$$

$$G_{a5,1} = G_{a4,1} - x_{131,1} \quad \textbf{(5.97)}$$

$$G_{a6,1} = G_{a5,1} + x_{133,1} \quad \textbf{(5.98)}$$

5.5.2.4 İzmir için denge kısıtları

İzmir için zaman uzay ağı şu şekildedir:



Şekil 5.7 : İzmir için zaman uzay ağı

İzmir için denge kısıtları şu şekilde yazılır:

$$G_{i1,1} = G_{i6,1} - x_{106,1} \quad \textbf{(5.99)}$$

$$G_{i2,1} = G_{i1,1} + x_{107,1} \quad \textbf{(5.100)}$$

$$G_{i3,1} = G_{i2,1} - x_{116,1} \quad \textbf{(5.101)}$$

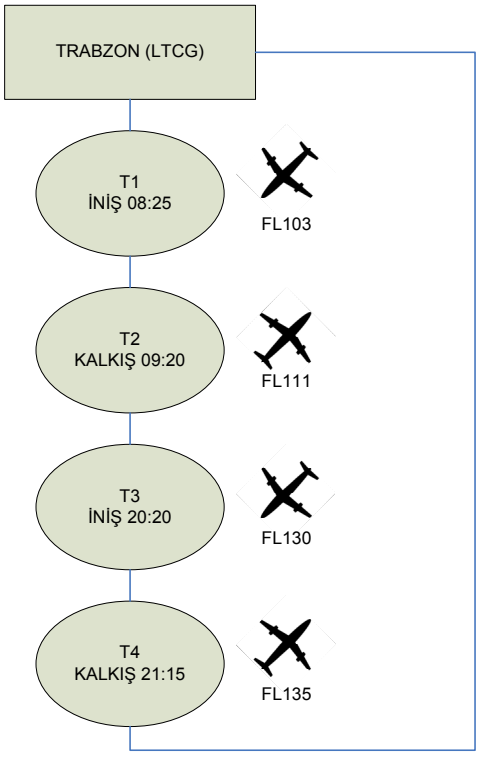
$$G_{i4,1} = G_{i3,1} + x_{126,1} \quad \textbf{(5.102)}$$

$$G_{i5,1} = G_{i4,1} - x_{138,1} \quad \textbf{(5.103)}$$

$$G_{i6,1} = G_{i5,1} + x_{134,1} \quad \textbf{(5.104)}$$

5.5.2.5 Trabzon için denge kısıtları

Trabzon için zaman uzay ağı şu şekildedir:



Şekil 5.8 : Trabzon için zaman uzay ağı

Trabzon için denge kısıtları şu şekilde yazılır.

$$G_{t1,1} = G_{t4,1} + x_{103,1} \quad \textbf{(5.105)}$$

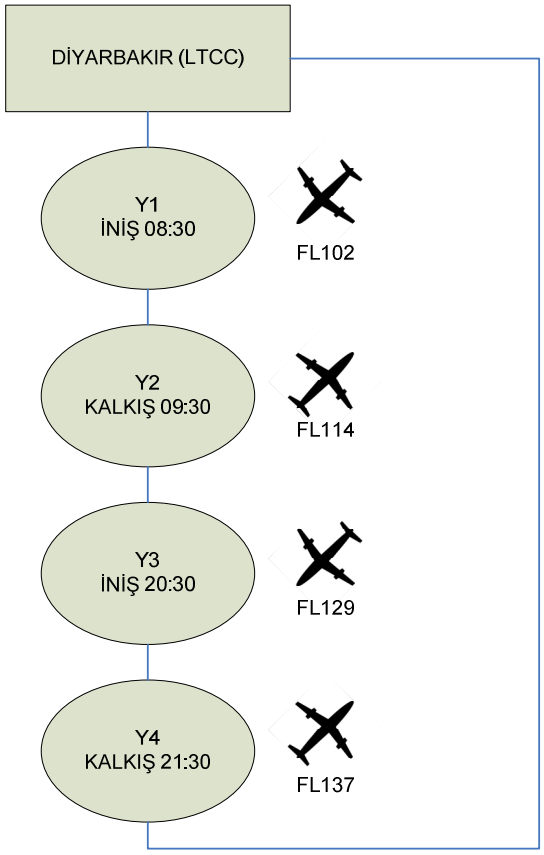
$$G_{t2,1} = G_{t1,1} - x_{111,1} \quad \textbf{(5.106)}$$

$$G_{t3,1} = G_{t2,1} + x_{130,1} \quad \textbf{(5.107)}$$

$$G_{t4,1} = G_{t3,1} - x_{135,1} \quad \textbf{(5.108)}$$

5.5.2.6 Diyarbakır için denge kısıtları

Diyarbakır için zaman uzay ağı şu şekildedir:



Şekil 5.9 : Diyarbakır için zaman uzay ağı

Diyarbakır için denge kısıtları şu şekilde yazılır:

$$G_{y1,1} = G_{y4,1} + x_{102,1} \quad \textbf{(5.109)}$$

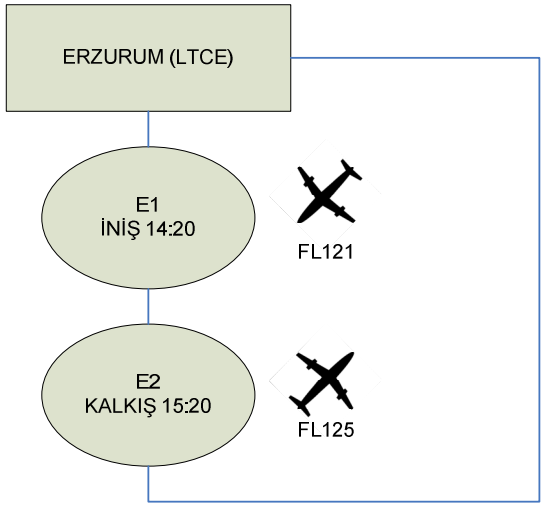
$$G_{y2,1} = G_{y1,1} - x_{114,1} \quad \textbf{(5.110)}$$

$$G_{y3,1} = G_{y2,1} + x_{129,1} \quad \textbf{(5.111)}$$

$$G_{y4,1} = G_{y3,1} - x_{137,1} \quad \textbf{(5.112)}$$

5.5.2.7 Erzurum için denge kısıtları

Erzurum için zaman uzay ağı şu şekildedir:



Şekil 5.10 : Erzurum için zaman uzay ağı

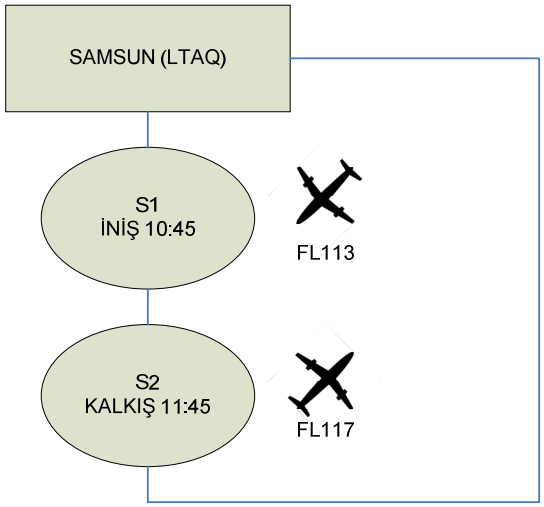
Erzurum için denge kısıtları şu şekildedir:

$$G_{e1,1} = G_{e2,1} + x_{121,1} \quad \textbf{(5.113)}$$

$$G_{e2,1} = G_{e1,1} - x_{125,1} \quad \textbf{(5.114)}$$

5.5.2.8 Samsun için denge kısıtları

Samsun için zaman uzay ağı şu şekildedir:



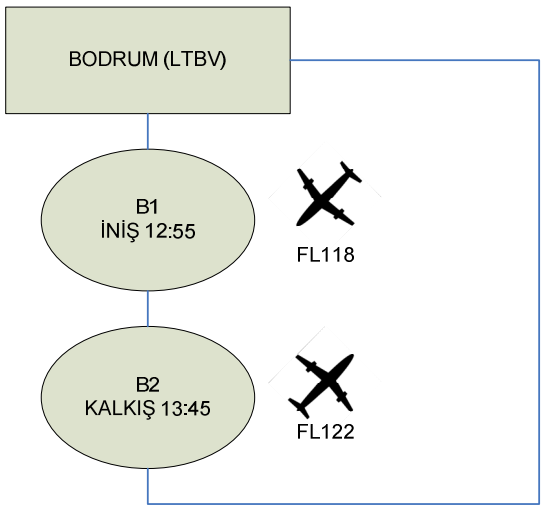
Şekil 5.11 : Samsun için zaman uzay ağı

$$G_{e1,1} = G_{e2,1} + x_{121,1} \quad \textbf{(5.115)}$$

$$G_{e2,1} = G_{e1,1} - x_{125,1} \quad \textbf{(5.116)}$$

5.5.2.9 Bodrum için denge kısıtları

Bodrum için zaman uzay ağı şu şekildedir:



Şekil 5.12 : Bodrum için zaman uzay ağı

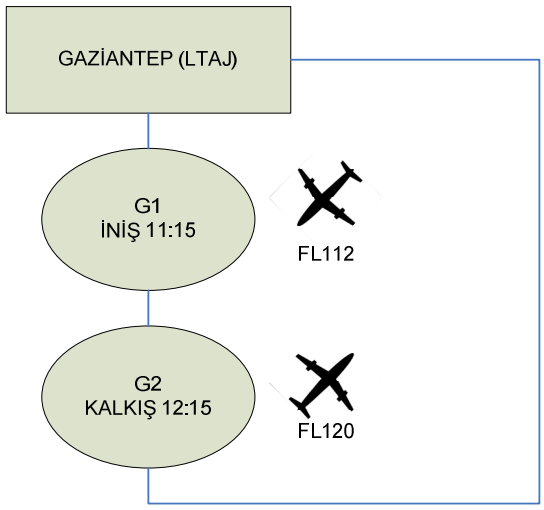
Bodrum için denge kısıtları şu şekilde yazılır:

$$G_{b1,1} = G_{b2,1} + x_{118,1} \quad \textbf{(5.117)}$$

$$G_{b2,1} = G_{b1,1} - x_{122,1} \quad \textbf{(5.118)}$$

5.5.2.10 Gaziantep için denge kısıtları

Gaziantep için zaman uzay ağı şu şekildedir:



Şekil 5.13 : Gaziantep için zaman uzay ağı

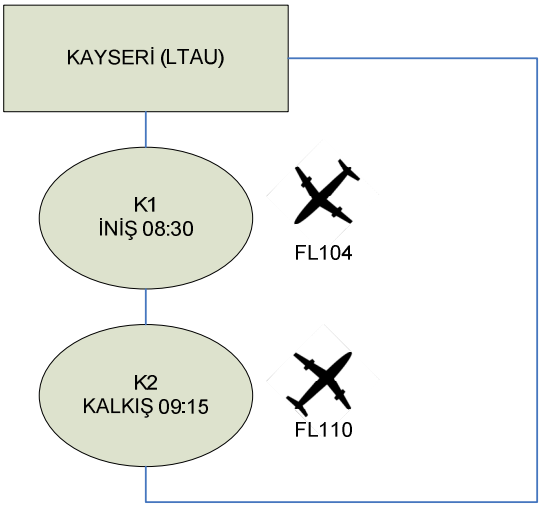
Gaziantep için denge kısıtları şu şekilde yazılır:

$$G_{G1,1} = G_{G2,1} + x_{112,1} \quad \textbf{(5.119)}$$

$$G_{G2,1} = G_{G1,1} - x_{120,1} \quad \textbf{(5.120)}$$

5.5.2.11 Kayseri için denge kısıtları

Kayseri için zaman uzay ağı şu şekildedir:



Şekil 5.14 : Kayseri için zaman uzay ağı

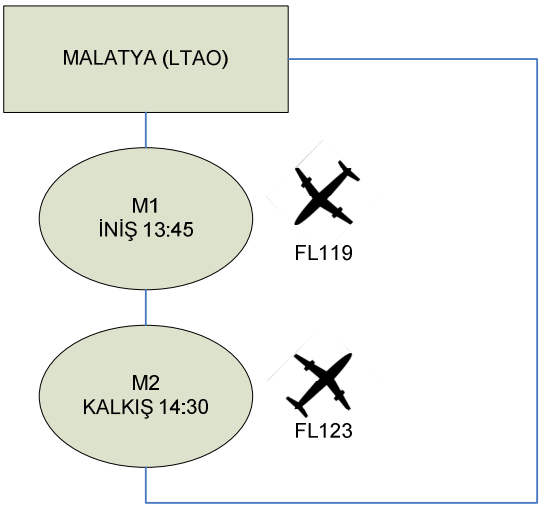
Kayseri için denge kısıtları şu şekildedir:

$$G_{k1,1} = G_{k2,1} + x_{104,1} \quad \textbf{(5.121)}$$

$$G_{k2,1} = G_{k1,1} - x_{110,1} \quad \textbf{(5.122)}$$

5.5.2.12 Malatya için denge kısıtları

Malatya için zaman uzay ağı şu şekildedir:



Şekil 5.15 : Malatya için zaman uzay ağı

Malatya için denge kısıtları şu şekilde yazılır:

$$G_{m1,1} = G_{m2,1} + x_{119,1} \quad \textbf{(5.123)}$$

$$G_{m2,1} = G_{m1,1} - x_{123,1} \quad \textbf{(5.124)}$$

5.5.3 Uçak müsaitlik kısıtları

Hane et al tarafından teklif edilen filo ataması modelindeki üçünü ve son kısıt kümesi filo müsaitlik kısıtlarıdır. Bu kısıt kümesi toplamda kullanılan uçak sayısının filoda müsait olan uçak sayısını geçmemesini sağlar.

Bu kısıt kümesini yazabilmek için her havaalanı için her filo tipinden kaç uçağın gece yerde kaldığının sayılması gerekir. Örnek olarak Antalya şeklinde son düğüm noktası A6'da (gün sonu yayı ile gün başı düğüm noktasının bağlandığı düğüm noktası) bulunan uçak sayısı o gece Antalya'da bulunan toplam uçak sayısını belirler. Bu havaalanı için $G_{A6,1}$ birinci filo tipi için Antalya'da bulunan uçak sayısını gösterir.[17]

Buna göre zaman uzay ağında birinci filo tipi için toplamda bulunan uçak sayısı şu şekildedir:

$$\sum uçak_{f1} = G_{L38,1} + G_{A6,1} + G_{D6,1} + G_{T4,1} + G_{Y4,1} + G_{B2,1} + G_{S2,1} + G_{K2,1} + G_{M2,1} + G_{E2,1} + G_{G2,1} \quad \textbf{(5.125)}$$

Yukarıdaki ifadede G değişkenleri sırasıyla İstanbul, Antalya, Adana,Trabzon, Diyarbakır, Bodrum, Samsun, Kayseri, Malatya, Erzurum ve Gaziantep için gece yerde bulunan uçak sayısını gösterir. İstanbul'da gün içinde 38 uçuş bulunduğundan İstanbul için son düğüm noktası $G_{L38,1}$ ile gösterilir.

Aynı şekilde ikinci, üçüncü ve dördüncü filo tipi için toplamda bulunan uçak sayısı şu şekildedir:

$$\sum uçak_{f2} = G_{L38,2} + G_{A6,2} + G_{D6,2} + G_{T4,2} + G_{Y4,2} + G_{B2,2} + G_{S2,2} + G_{K2,2} + G_{M2,2} + G_{E2,2} + G_{G2,2} \quad \textbf{(5.126)}$$

$$\sum uçak_{f3} = G_{L38,3} + G_{A6,3} + G_{D6,3} + G_{T4,3} + G_{Y4,3} + G_{B2,3} + G_{S2,3} + G_{K2,3} + G_{M2,3} + G_{E2,3} + G_{G2,3} \quad \textbf{(5.127)}$$

$$\sum uçak_{f4} = G_{L38,4} + G_{A6,4} + G_{D6,4} + G_{T4,4} + G_{Y4,4} + G_{B2,4} + G_{S2,4} + G_{K2,4} + G_{M2,4} + G_{E2,4} + G_{G2,4} \quad \textbf{(5.128)}$$

Kullanılan örnek modelde MD 80 filosu için dokuz, A321 filosu için sekiz, A300-600 filosu için 6 ve A300-B4-200 filosu için iki uçak bulunmaktadır. Dolayısıyla uçak müsaitlik kısıtları şu şekilde yazılabilir:

$$G_{L38,1} + G_{A6,1} + G_{D6,1} + G_{T4,1} + G_{Y4,1} + G_{B2,1} + G_{S2,1} + G_{K2,1} + G_{M2,1} + G_{E2,1} + G_{G2,1} \leq 9 \quad \textbf{(5.129)}$$

$$G_{L38,2} + G_{A6,2} + G_{D6,2} + G_{T4,2} + G_{Y4,2} + G_{B2,2} + G_{S2,2} + G_{K2,2} + G_{M2,2} + G_{E2,2} + G_{G2,2} \leq 8 \quad \textbf{(5.130)}$$

$$G_{L38,3} + G_{A6,3} + G_{D6,3} + G_{T4,3} + G_{Y4,3} + G_{B2,3} + G_{S2,3} + G_{K2,3} + G_{M2,3} + G_{E2,3} + G_{G2,3} \leq 6 \quad \textbf{(5.131)}$$

$$G_{L38,4} + G_{A6,4} + G_{D6,4} + G_{T4,4} + G_{Y4,4} + G_{B2,4} + G_{S2,4} + G_{K2,4} + G_{M2,4} + G_{E2,4} + G_{G2,4} \leq 2 \quad \textbf{(5.132)}$$

Örnek modelde dört adet filo tipi bulunduğundan bu kısıt kümesinde sadece dört adet kısıt fonksiyonu bulunmaktadır.

6. SONUÇ VE ÖNERİLER

Kurulan örnek modelde elde edilen Onur Air'in Pazartesi günü uçuşları kullanılmıştır. Bu modelde dört filo tipi için 12 merkezli toplam 38 adet uçuş bulunmaktadır.Kısıt fonksiyonları sayısı ise aşağıdaki gibidir:

- Kapsama kısıt fonksiyonları: 304 adet
- Denge kısıt fonksiyonları: 38 adet
- Müsaitlik kısıt fonksiyonları: 4 adet

Kurulan örnek modeldeki değişken sayıları ise şu şekildedir:

- 152 adet ikili değişken
- 304 adet tamsayı değişken

Model daha önce belirtilen Karmaşık tamsayılı programlama çözüm yöntemlerine göre ayrı ayrı kodlanıp incelenmiştir.

6.1 Yorucu Numaralama Algoritması İçin Çözüm

Yorucu numaralama eniyileme tekniklerinin en basiti olduğu gibi, işlem yükü bakımından en zorlu olanıdır. Değişkenlerin alabileceği değerlerin tek tek denenmesi sonucunda eniyilenmiş çözüm bulunur.[17]

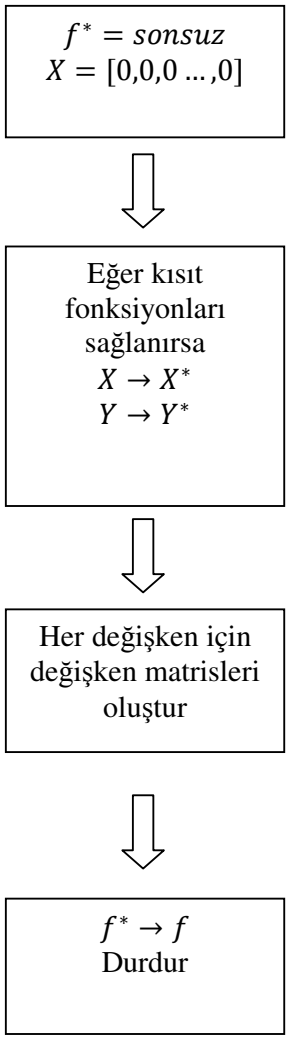
Örnek modelde toplam 152 ikili değişken ve 304 tam sayı değişken bulunduğundan yapılacak toplam deneme sayısı şu şekilde hesaplanır:

$$N_d(deneme\ sayısı) = \prod_{i=1}^{n_{de}} p_i \qquad \textbf{(6.1)}$$

Kullanılan formüldeki n_{de} değişkenlerin toplam sayısını, p_i ise alabilecekleri en büyük değerleri temsil eder. Dolayısıyla örnek model için kullanılan toplam işlem sayısı şu şekilde olur:

$$N_d = \prod_{i=1}^{152} 2 \times \prod_{i=1}^{76} 9 \times \prod_{i=1}^{76} 8 \times \prod_{i=1}^{76} 6 \times \prod_{i=1}^{76} 2 = 6.1960 \times 10^{209} \qquad \textbf{(6.2)}$$

Yorucu numaralama için ise akış diyagramı şu şekildedir:



Şekil 6.1 : Yorucu numaralama için akış diyagramı.

Yorucu numaralama için MATLAB ortamında yazılan kod, yorucu numaralama tanıtılırken el ile çözülen problemin çözümünde kullanılmıştır. Sonuç bir saniyeden kısa bir sürede tam olarak bulunmuştur.

Örnek model için kurulan yorucu numaralama kodu MATLAB ortamında Intel CentrinoDuo işlemcili 2.0 Ghz frekansında çalışan bir makinada yaklaşık 4 gün boyunca çalıştırılmıştır. Her ne kadar yapılacak işlemler gayet basit olsa da yapılacak toplam işlem sayısı 6.1960×10^{209} olduğundan olurlu çözüm bulunamamıştır.

Yorucu numaralamanın en zayıf yönü olan, olursuz(infeasible) çözümlerin de işlemler içine katılmasıdır. Karmaşık tamsayı programlama kullanan bir filo ataması

modelinde sadece bir çözüm bulunduğundan 6.1960×10^{209} değişik çözüm içinde olurlu çözümü aramak pratik olmaktan çok uzaktır.

6.2 Dal ve Sınır Algoritması İçin Çözüm

Dal ve sınır algoritmasında bulunan dallandırma şemasını kurmak çok küçük boyuttaki modeller için bile zordur. Kullanılan örnek modelde karar değişkeni sayısı 456 olduğundan dallandırma şemasının el ile gösterimi pratik olarak imkansızdır. Bu sebeple dal ve sınır algoritmasının filo atama problemindeki çözüm yeteneğini görmek amacıyla bir kod yazılmıştır.

Dal ve sınır algoritması için MATLAB Optimization Toolbox ile yazılan kod için akış diyagramı şu şekildedir:

1. Adım

   Kök düğüm noktasında bütün ayrık değişkenleri bağımsızlaştır.

   Çözüm kümesi s'i 0'a ayarla.

   Kümedeki görevli çözümde $f^{**} = \infty$ ve $X^{**} = [\infty]$ olarak ayarla, (enküçükleme problemi)

2. Adım

   Eğer olurlu kısmi çözümlerden biri mevcutsa görevli çözüm olarak bu çözümü $X^{(s)}$ olarak belirle ve 3. Adıma git

   Eğer olurlu kısmi çözüm mevcut değilse görevli çözüm eniyilenmiş çözümdür.

   Eğer olurlu kısmi çözüm ve görevli çözüm mevcut değilse problem olursuzdur.

3. Adım

   $X^{(s)}$ kısmi çözümü için tam çözümü orijinal modelde sürekli rahatlatma yöntemi ile bul.

4. Adım

   Eğer $X^{(s)}$ kısmi çözümü için tam çözüm yoksa, $X^{(s)}$ çözümünü iptal et, adım büyüklüğünü $s \rightarrow s + 1$ olarak belirle ve 2. Adıma geri dön.

5. Adım

   Eğer $X^{(s)}$ kısmi çözümü için olurlu tam çözüm görevli çözümü

geliştiremiyorsa, $X^{(s)}$ çözümünü iptal et, adım büyüklüğünü $s \rightarrow s+1$ olarak belirle ve 2. Adıma geri dön.

6. Adım

   Eğer olurlu tam çözüm görevli çözümden daha iyiyse görevli çözüm ile yer değiştir: $f^{**} \leftarrow f^{(s)},\ X^{**} \leftarrow X^{(s)}$

   $X^{(s)}$ çözümünü iptal et, adım büyüklüğünü $s \rightarrow s+1$ olarak belirle ve 2. Adıma geri dön.

Dal ve sınır algoritması kodu, dal ve sınır algoritmasının tanıtılmasında kullanılan ve eli ile çözülen aşağıdaki probleme uygulanmış el ile bulunan çözüm birebir bulunmuştur.

Filo ataması problemi için kullanılan örnek model dal ve sınır algoritmasına uyarlanmış ve MATLAB ortamında Intel CentrinoDuo işlemcili 2.0 Ghz frekansında çalışan bir makinada çalıştırılmıştır. 12 saniyede Dal ve sınır algoritması ile bulunan sonuçlar Çizelge (6.1)'de verilmiştir.(Uçuşa atanan filo tipi "X" ile işaretlidir).

Çözüm 61 iterasyonda bulunmuştur. Bulunan amaç fonksiyonu değeri 579843. $'dır.

Dal ve sınır algoritması, yorucu numaralama ile karşılaştırılırsa, numaralama sayısındaki azalma iki sebepten dolayı bulunamaz. Bunlardan birincisi ilk baştaki değişkenler kümesini sınırlarken kullanılan sınırlama yönteminde her seferinde göreceli bir sınırlama kullanılmasıdır. Bu sebeple değişkenler kümesi için kullanılan sınırlamalar çözümün daha farklı şekilde yakınsamasına sebep olabilir. İkinci sebep ise yorucu numaralamanın basit problemler için çok kolay yakınsarken, karmaşık ve yüksek değişken sayılı problemlerde çok zor yakınsaması, yada hiç yakınsamamasıdır. İyi seçilen başlangıç değerleri problemde kullanılan numaralama sayısını büyük miktarda azaltırken, nispeten kötü seçilen başlangıç değerleri problemin yakınsamasını zorlaştırır, dolayısıyla numaralama sayısını büyük miktarda arttırır.

Çizelge 6.1 : Dal ve sınır algoritması ile atanan filo tipleri

UÇUŞ NO	FİLO TİPİ			
	MD-80	A321	A300-600	A300-b4-200
101				X
102		X		
103	X			
104	X			
105			X	
106	X			
107	X			
108			X	
109		X		
110	X			
111		X		
112		X		
113	X			
114		X		
115				X
116	X			
117	X			
118	X			
119				X
120				X
121		X		
122	X			
123		X		
124				X
125		X		
126			X	
127			X	
128				X
129				X
130		X		
131				X
132				X
133			X	
134			X	
135		X		
136			X	
137				X
138			X	

6.3 Gomory Kesme Düzlemi Yöntemi İçin Çözüm

Gomory kesme düzlemi yöntemi için GAMS adlı ticari eniyileme programında Westerlund Tapio ve Poern Ray tarafından geliştirilen AlphaECP çözücüsü kullanılmıştır. Gomory kesme düzlemi yöntemi tanıtılırken el ile çözülen problem

GAMS'ta kodlanmış ve çalıştırılmıştır. Sonuç bir saniyeden kısa bir sürede bulunmuştur.

Filo ataması problemindeki örnek model GAMS ile kodlanmış ve AlphaECP çözücüsü ile 31 saniyede çözülmüştür. Çözümdeki iterasyon ve kesme düzlemi sayıları AlphaECP çıktısı ile aşağıdaki gibidir:

Çizelge 6.2 : Gomory kesme düzlemi iterasyon sonuçları

İterasyon Sayısı	Kesme düzlemi sayısı	Kısıt ihlali sayısı	Amaç Fonksiyonu Değeri
1	34	34	101223
2	42	34	101223
3	67	34	101223
4	89	33	88942
5	122	27	679467
6	130	16	498176
7	133	16	498176
8	137	13	563487
9	139	13	563487
142	141	0	579843

Gomory kesme düzlemi yöntemi tarafından bulunan sonuç dal ve sınır algoritması ile birebir örtüşmektedir. Ancak dal ve sınır algoritması tarafından 61 iterasyon ve 12 saniyede bulunan çözüm Gomory kesme düzlemi yöntemi tarafından 142 iterasyon ve 31 saniyede bulunmuştur. Bunun sebebi ise dal ve sınır algoritmasında aday çözümler üzerinde çözüme ulaşılmaya çalışılırken, Gomory kesme düzlemi yönteminde tüm çözüm uzayı üzerinden kesme düzlemleri çizilerek çözüme ulaşılmaya çalışılmasıdır.[12]

Gomory kesme düzlemi yöntemi tarafından uçuşlara atanan filo tipleri şu şekildedir:

Çizelge 6.3 : Gomory kesme düzlemi ile atanan filo tipleri

UÇUŞ NO	FİLO TİPİ			
	MD-80	A321	A300-600	A300-b4-200
101				X
102		X		
103	X			
104	X			
105			X	
106	X			
107	X			
108			X	
109		X		
110	X			
111		X		
112		X		
113	X			
114		X		
115				X
116	X			
117	X			
118	X			
119				X
120				X
121		X		
122	X			
123		X		
124				X
125		X		
126			X	
127			X	
128				X
129				X
130		X		
131				X
132				X
133			X	
134			X	
135		X		
136			X	
137				X
138			X	

6.4 Genetik Algoritma Yöntemi İçin Çözüm

Filo ataması probleminin çözümünde kullanılan son teknik genetik algoritmadır. Genetik algoritma kodu MATLAB'da yazılıp çalıştırılmıştır. Genetik algoritma için kullanılan değişken özellikleri şu şekildedir:

6.4.1 Başlangıç populasyonu

İlk populasyon genetik algoritmada iterasyona girecek ilk çözüm kümesidir ve bu kümenin üretilmesi genetik algoritma sürecinin ilk aşamasıdır. Başlangıç populasyonu üretilirken aşağıdaki yöntem uygulanmıştır:

- F= Filo tipi kümesi

  S=Ağdaki istasyon sayısı

  $x_{F,L}$ = L uçuşuna atanan F filo tipi olmak üzere

  - Başlangıç değerlerinin tümünü sıfıra eşitle.
  - F filo tipi kümesindeki her filo tipi için
    a. $x_{F,L}$=1 ise bir sonraki adıma git
    b. Herhangi bir L uçuşu için rastgele bir filo tipi ata
    c. $\forall L \supset F$ için dur.

Başlangıç populasyonunu rastgele seçmek yerine, uygun bir şekilde oluşturmak genetik algoritmaya yardımcı olur ve iterasyon sayısını azaltır.

6.4.2 Kromozom sayısı

Populasyondaki kromozom sayısı aşağıdaki formüle göre belirlenmiştir:

$$Populasyon\ sayısı = Uçuş\ sayısı \times Filo\ tipi\ sayısı \times 20 \quad \textbf{(6.3)}$$

Populasyon için kromozom sayısını arttırmak iterasyon sayısını düşürmesine rağmen işlem yükünü arttırdığından hesaplama süresinde önemli bir değişikliğe sebep olmamaktadır.

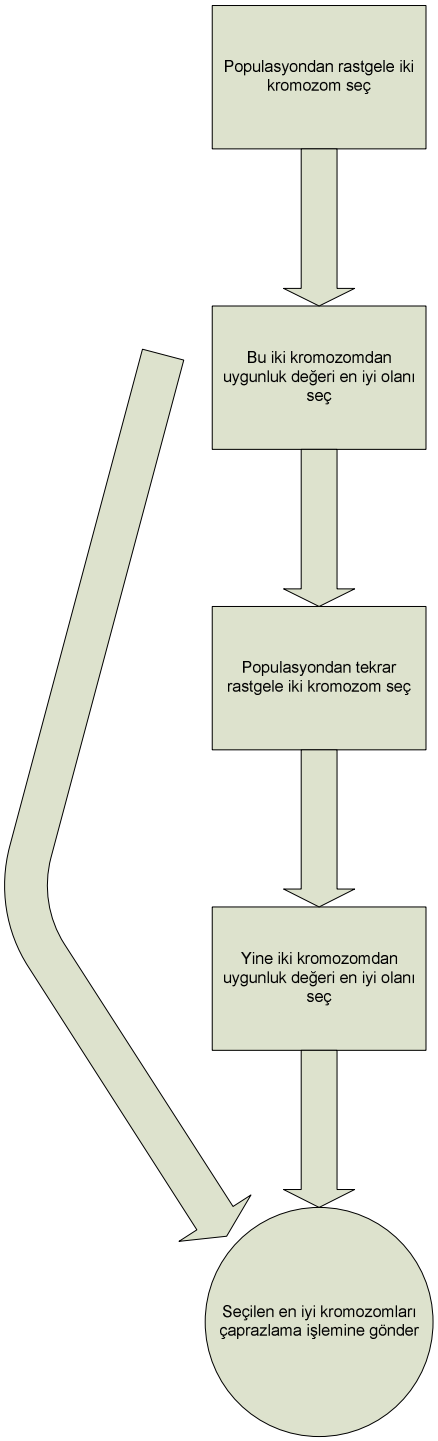
6.4.3 Seçilim operatörü

Seçilim işlemi, genetik algoritmanın her iterasyon için ilk işlemidir. Bu aşamadaki amaç bundan sonraki aşama olan çaprazlama işlemine girecek olan kromozomların

seçilmesidir. Seçilim işlemi sonunda belirlenen kromozomlar çaprazlama işlemine genetik bilgileri aktardığından bu aşamada seçilecek olan kromozomlar doğrudan yakınsama hızına etki eder. Bu çalışmada [14]'de teklif edilen ikili turnuva seçilim operatörü kullanılmıştır.

6.4.3.1 İkili turnuva seçilim operatörü

Bu seçilim operatörü fazladan bir işlem yükü getirmediğinden hızlı çalışmakta ve performans açısından bir dezavantaj getirmemektedir. Bu yöntem için akış diyagramı şu şekildedir:



Şekil 6.2 : İkili turnuva seçilim operatörü için akış diyagramı

6.4.4 Çaprazlama Operatörü

Çaprazlama operatörü, genetik bilginin yeni nesillere aktarıldığı yani çocuk kromozomların oluşturulduğu aşamadır. Seleksiyon işleminden sonra seçilen yüksek uygunluk değerine sahip iki kromozom çarprazlanarak yeni bir çocuk

oluşturulmaktadır. Bu çalışmada [16] numaralı referansta önerilen fusion çarprazlama operatörü bir kaç değişikliğe uğratılıp kullanılmıştır. Fusion çaprazlama operatörü her seferinde tam sayılı değer vermeyi garanti edemediğinden, yapılan değişiklik ile fusion çarprazlama operatörünün tam sayılı değer vermesi garantilenmiştir[18].

6.4.5 Mutasyon Operatörü

Mutasyon operatörü algoritmanın yerel enküçük ve yerel en büyük noktalara yakalanmamasını sağlamak amacıyla çarprazlama operatörü tarafından üretilen çocuk bireylerin genlerinde değişikliğe sebep olur ve küçük boyutlu bir çözüm uzayında eniyilenmiş noktaya yakınsamayı kolaylaştırır. Algoritma eniyilenmiş noktaya yaklaştıkça yakınsama hızı azalmakta ve bu aşamadan sonra mutasyon operatörü ile yapılan aramalar optimum noktaya yakınsamayı kolaylaştırmaktadır.

Karmaşık tam sayılı programlamada değişkenlerin tamsayı değerleri alması zorunlu olduğundan mutasyon operatörü tarafından oluşturulan yeni bireyler tam sayı değerleri alması için [0,10] aralığında en yakın oldukları tam sayı değerine yuvarlanmaktadır. Böylelikle üretilen bireylerin değerlerinde çok küçük bir değişim olurken, bireylerin tam sayı değerleri alması garantilenmiş olur.

6.4.6 Sonuç

Mutasyon operatörü algoritmanın yerel enküçük ve yerel en büyük noktalara yakalanmamasını sağlamak amacıyla çarprazlama operatörü tarafından üretilen çocuk bireylerin genlerinde değişikliğe Filo ataması problemi için belirlenen örnek model yukarıdaki parametrelere bağlı olarak yapılan kodun çalıştırılması ile, 20 dakika sonunda Çizelge (6.4) ile verilen sonuç bulunmuştur.

Genetik algoritma sonucunda amaç fonksiyonu değeri, yani toplam maliyet 677,892$ olarak bulunmuştur. Aynı zamanda bir adet kısıt fonksiyonu da ihlal edilmiştir. Bunun sebebi sezgisel yaklaşım temelli algoritmalarda kesin sonuca ulaşmanın rastlantısal olarak gerçekleştirilmesidir. Pek çok mühendislik probleminde sadece bir adet kısıt fonksiyonu ihlali kabul edilebilir düzeydeyken, filo ataması problemi gibi kesin sonuçlara ihtiyaç duyan problemlerde sezgisel yaklaşımların uygulanabilirliği tartışılabilir[18].

Çizelge 6.4 : Genetik Algoritma Sonuçları

UÇUŞ NO	FİLO TİPİ			
	MD-80	**A321**	**A300-600**	**A300-B4-200**
101				X
102		X		
103	X			
104	X			
105			X	
106	X			
107	X			
108			X	
109			X	
110	X			
111		X		
112		X		
113	X			
114		X		
115				X
116	X			
117	X			
118	X			
119				X
120				X
121		X		
122	X			
123		X		
124				X
125		X		
126			X	
127			X	
128	X			
129				X
130	X			
131				X
132				X
133			X	
134	X			
135	X			
136			X	
137				X
138			X	

6.5 Öneriler

Bu çalışmada havayolu filo atama problemi ayrıntılarıyla incelenmiş, karmaşık tam sayı programlama çözüm yöntemleri ile bir örnek model çözülmüştür.

Kullanılan kodlar arasında çözüme en hızlı yakınsayan dal ve sınır algoritması olmuştur. Bunun sebebi dal ve sınır algoritmasının çözüm uzayını sürekli olarak daraltması bunun yanında da aday çözümler için yapılan denemelerin gerçekçi olmasıdır.

Bu çalışma aynı zamanda havayolları için çok önemli olan uçak atama problemi için bir öncalışma mahiyetindedir.

İleride yapılacak çalışmalarda dal ve sınır algoritmasını ve kullanılan filo ataması modelini geliştirmek yararlı olacaktır. Aynı zamanda karmaşık tam sayılı programlama için tam olarak kurulamayan ve çözüm garantisi vermeyen genetik algoritma için de bazı iyileştirmeler yapılabilir.

KAYNAKLAR

[1] **ECLAT Consulting,** 2003 : First Equity Aviation And Aerospace Almanac

[2] **Abara, J.,** 1989: Applying mixed integer programming to the fleet assignment problem. *Interfaces,* Vol 19(4).

[3] **Hane, C.A., Barnhart, Marsten, R.E., Nemhauser, G., Sigismondi, G.,** 1995: The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming.*

[4] **Rushmeier, R.A., Kontogiorgis, S.A.,** 1997. Advances in the optimization of airline fleet assignment. *Transportation Science.* Vol 31.

[5] **Lohatepanont, M., Barnhart, C.,** 2004. Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*. Vol 31.

[6] **Hanif D. Sherali, Ebru K. Bish, Xiaomei Zhu,** 2006. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operations Research.*

[7] **Lohatepanont, M., Barnhart, C.,** 2004. Advances in the optimization of airline fleet assignment. Transportation Science. Vol 31.

[8] **Luenberg D. G.,** 1984. Linear and Nonlinear Programming, Addisson& Wiley, London, UK

[9]**Topçu, Y.İ.,** 2008. END 331 Yöneylem araştırması ders notları. İTÜ, İstanbul, Türkiye.

[10]**Koçak, C.K.,** 2007. Special Topics in Operations Research, Boğaziçi University Press, İstanbul, Türkiye.

[11] **J.H. Holland,** 1975. Adaptation in Natural and Artifical Systems. MIT Press.

[12] **Gomory, R.E.,** 1962. Cuts in combinatiral optimization, Princeton University Press, USA.

[13] **Şen, Z.,** 2004. Genetik Algoritmalar ve En İyileme Yöntemleri. Su Vakfı Yayınları, Istanbul, Türkiye.

[14] **Uwe Aicklein**, 2002. An Indirect Genetic Algorithm for Set Covering Problems, Journal of Operational Research Society. Vol 53:1118-1126

[15] **Klabjan, D.**, 2005. Large-scale models in the airline industry. Kluwer Academic Publishers, in press.

[16] **Bish, E.K., Suwandechochai, R., Bish, D.R.,** 2004. Strategies for managing the flexible capacity in the airline industry. *Naval ResearchLogistics* 51, 654–685.

[17] **Yu, G., Yang, J.,** 1998. Optimization applications in the airline industry. In: Du, D.Z., Pardalos, P.M. (Eds.), Handbook of Combinatorial Optimization. Kluwer Academic Publishers, Norwell, MA, pp. 635–726.

[18] **Zeren Bahadır,** 2008. Genetik Algoritmalar ile Havayolu Ekip Planlamada Ekip Rotasyon optimizasyonu. MSc Tezi, İTÜ.

ÖZGEÇMİŞ

Ad Soyad: Doğan Akay

Doğum Yeri ve Tarihi: 17.05.1986, Bakırköy - İstanbul

Lisans Üniversite: İstanbul Teknik Üniversitesi, Uzay Mühendisliği 2003.