T.C.

ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJE-1 RAPORU

Hazırlayanlar : Kadir ÇOLAK 160401037

: Kerim ULUSOY 160401026

Öğretim Elemanının Unvanı/Adı Soyadı : Prof . Dr İsmail KADAYIF

ILP (Integeter Linear Programming) kullanılarak Max/Min Problem Çözümü

Ocak,2021
Çanakkale

İçindekiler Tablosu

ŞEKİLLER TABLOSU 3

KISALTMALAR 3

Giriş 4

BİRİNCİ BÖLÜM 5

1.DOĞRUSAL PROGRAMLAMA 5

1.1 Doğrusal Programlama İle İlgili Tanımlar 5

1.2.Doğrusal Programlama Probleminin Matematiksel Yapısı ve Modelin Kurulması 6

1.3 Doğrusal Programlama Çözüm Teknikleri 10

- **Grafik çözüm yöntemi** 10
- **Grafik Çözümde Özel Durumlar** 12
- **Simpleks metot** 12
- **Büyük M yöntemi** 17
- **İki aşamalı yöntem** 18
- **Dualite** 18
- **Dual simpleks yöntem** 21

1.4 Duyarlılık Analizi 21

- **Uygunluğu etkileyen değişiklikler** 22
- **Optimumluğu etkileyen değişiklikler** 23

1.5 Doğrusal Programlama Çözücüleri 23

1.6 Doğrusal Programlama Çözücüleri için Kullanılan Platformlar 25

İKİNCİ BÖLÜM 26

2.TAMSAYILI DOĞRUSAL PROGRAMLAMA 26

2.1 Tamsayılı Doğrusal Programlama Modelleri 29

- **Tüm tamsayılı programlama modeli** 29
- **Karma tamsayılı programlama modeli** 30
- **Sıfır - bir (0–1) tamsayılı programlama modeli** 30

2.2 Tamsayılı Doğrusal Programlamada Çözüm Yöntemleri 31

- **Kesme düzlemi algoritması** 33
- **Dal-sınır (DS) algoritması** 37

2.3 Tamsayılı Programlamanın Kullanımı Örnekleri 42

- **Sanayideki uygulamalar** 42
- **Hizmet sektörü** 46

2.4 Tamsayılı Programlama Yazılımları 54

- **Baron (Branch-And-Reduce Optimization Navigator)** 54
- **Conopt** 54
- **Snopt (Sparse Nonlineer Optimization)** 55
- **AOA (AIMMS Outer Approximation)** 55
- **Xpress** 55
- **Xpress mosel** 56
- **Xpress MP** 56
- **Xpress-MP kütüphaneleri** 56
- **Knitro** 57

- **Gams (The General Algebraic Modeling System)** **57**
- **Path** **57**
- **Minos** **58**
- **Tora** **58**
- **WinQSB** **58**

2.5 Tamsayılı Programlama Yazılımlarının Bugünü ve Geleceği **58**

ÜÇÜNCÜ BÖLÜM **60**

3. KULLANDIĞIMIZ KÜTÜPHANEDEKİ ILP FONKSİYONLARI VE İŞLEVLERİ **60**

DÖRDÜNCÜ BÖLÜM **62**

4. ILP İLE MAX/MİN PROBLEMİ ÇÖZÜMÜ **62**

ŞEKİLLER TABLOSU

Şekil 1 Grafik çözüm yöntemi için bir örnek 11
Şekil 2 Başlangıç simpleks tablosu 13
Şekil 3 Simpleks algoritmasının çözüm matrisi 16
Şekil 4 Simpleks tablosu 17
Şekil 5 Prime-Dual İlişkisi 19
Şekil 6 DP-TDP çözüm farkı 26
Şekil 7 Tamsayılı doğrusal programlamanın grafik ile gösterimi 27
Şekil 8 Yuvarlaklaştırmanın grafiksel gösterimi 28
Şekil 9 Simpleks tablosu 35
Şekil 10 Kesme düzlemi algoritmasının çözüm grafiği 37
Şekil 11-Dal-Sınır Yönteminin Gösterimi 38
Şekil 12 Çözüm Karşılaştırması 41
Şekil 13 Tedarik Ağının Gösterimi 52

KISALTMALAR

CAD **:** Computer Aided Design

CAE **:** Computer Aided Engineering

CML **:** Constrained Maximum Likelihood

CNC **:** Computer Numerical Control

DP **:** Doğrusal Programlama

DS **:** Dal – Sınır Algoritması

SME : Küçük ve Orta Büyüklükteki İşletme

SQP : Sequential Quadratic Programing

TDP : Tamsayılı Doğrusal Programlama

TORA : Taha Operations Research Application

W : Watt

Giriş

Doğrusal programlama problemlerinde değişkenlerin tamsayı değerleri olması durumunda, probleme tamsayılı doğrusal programlama problemi adı verilir. Değişkenlerin yalnızca bir kısmının tamsayı değerlerine sahip olması durumunda bu model, karışık tamsayılı programlama problemi olarak adlandırılır. Eğer değişkenlerin tamamı tamsayılardan oluşuyor ise, saf tamsayılı programlama problemi olarak adlandırılır. Yalnızca 0 ve 1 ikili değişkenlerini içeren tamsayılı problemlere ise ikili (veya 0-1 tamsayı) programlama problemi adı verilir.

Bu çalışmada birinci bölümde doğrusal programlama tanımları çözüm yöntemleri ve modellerinden bahsettik.

İkinci bölümde tamsayılı doğrusal programlama başlığı altında tamsayılı doğrusal programlama model türleri olan tüm tamsayılı doğrusal programlama, karma tamsayılı doğrusal programlama ve 0-1 tamsayılı doğrusal programlama modellerini anlattık.Daha sonra çözüm yöntemi olan kesme düzlemi yöntemi ile dal-sınır yöntemlerini anlatımı ve matematiksel ifadelerle gösterimi yer almıştır. Ayrıca adı geçen bölümde, tam sayılı doğrusal programlama ile ilgili elde edilen uygulamalara dair geniş bir özet bilgi sunduk.Tam sayılı doğrusal programlama yazılımlarından bahsettik.

Üçüncü bölümde kullanacağımız olan dilin tamsayılı doğrusal programlama kütüphanesi hakkında bilgiler sunduk.

Dördüncü bölümde ise tamsayılı doğrusal programlama kullanarak max/min problemleri çözdük.

BİRİNCİ BÖLÜM

1.DOĞRUSAL PROGRAMLAMA

Doğrusal Programlama, eldeki sınırlı kaynakların optimum dağılımını belirlemek için kullanılan matematiksel bir tekniktir. Bir başka deyişle, doğrusal programlama problemleri, bir fonksiyonun belirli kısıtlayıcılar altındaki maksimizasyonundan veya minimizasyonundan ibarettir.

İşletme problemlerinin, matematik modellerin yardımı ile analiz edilmesinde doğrusal programlama önemli bir yer tutar. Özellikle matematiksel programlama teknikleri, ekonomik karar problemlerinde uygulanan doğrusal programlama aracılığı ile ortaya çıkmıştır. Doğrusal programlama metodu, amaçların ve sınırlayıcı şartların doğrusal fonksiyon ile ele alınması varsayımına dayanmaktadır. En ekonomik kararın verilmesi ise, mevcut şartlar altında ekonomik amaca optimum şekilde ulaşılmasının sağlanması anlamına gelmektedir.

Doğrusal programlama tekniği askeri problemlerden endüstriyel problemlere ve hizmet sektörüne kadar geniş bir yelpazede uygulanmaktadır.

1.1 Doğrusal Programlama İle İlgili Tanımlar

Doğrusal programlamada kullanılan ifadeler aşağıda kısaca özetlenmiştir

- Uygun Çözüm: Doğrusal programlama probleminin tüm kısıtlarını ve negatif olmama koşulunu sağlayan $x = (x_1, x_2, \ldots, x_n)$ vektörüdür.
- Optimum Çözüm: Doğrusal programlama probleminin tüm kısıtlarını sağlayan ve içlerinde amaç fonksiyonunun gereğini en üst düzeyde karşılayan çözümdür.
- Temel Çözüm: Kısıt denkleminin n-m (n'nin m'den farkı) tane değişkeninin sıfıra eşitlenerek m değişken için problemin çözümü temel çözümdür. Burada n değişken sayısını, m de kısıt sayısını temsil etmektedir.

1.2.Doğrusal Programlama Probleminin Matematiksel Yapısı ve Modelin Kurulması

Matematikte *n* bilinmeyenli bir doğrusal model ancak *n* tane birbirinden bağımsız doğrusal denklemlerle çözülebildiği halde, doğrusal programlama ile aynı sayıda bilinmeyenli bir doğrusal model bu sayıdan daha az denklem yardımıyla çözülebilmektedir. Bir problemin çözümünde doğrusal programlama yönteminin kullanılabilmesi için öncelikle problemin net bir şekilde tanımlanması gerekir. Karar verici, ilk olarak çözmek istediği problem için elindeki olanaklar ile varmak istediği sonucu belirlemelidir. Bunu yaparken problemin aşağıdaki unsurlara göre tanımlanmasına dikkat edilir:

- **Amaç fonksiyonu (Varılmak istenen hedefin matematiksel ifadesi):**

Doğrusal programlama modelinden sonuç alınabilmesi için amacın net ve açık olarak bilinmesi ve nicel olarak yazılımı gereklidir. Amaç, kâr maksimizasyonu veya maliyet minimizasyonu olabilir. Amaç fonksiyonunda x1, x2,x3 …,xn karar değişkenlerini, c1, c2, …, cj, …, cn ise kâr veya maliyet sabit kat sayılarını ifade etmek üzere, amaç fonksiyonun genel ifadesi aşağıdaki gibidir.

$$\text{Maks(Min)} Z = c_1 x_1 + c_2 x_2 + ... + c_n x_n$$

ve genel bir ifade yazılmak istenirse;

$$\text{Maks (Min) } Z = \sum_{j=1}^{n} c_j x_j \quad j=1,2,3\ldots n$$

şeklinde yazılır.Amaç Z'yi maksimize veya minimize edecek karar değişkenlerinin değerini bulmaktır. Amaç fonksiyonundaki parametreler problemin türüne göre değişir. Minimizasyon problemlerinde değişkenlerin katsayıları, birim başına maliyeti, maksimizasyon problemlerinde değişkenlerin katsayıları, birim başına elde edilen kârın katsayılarıdır.

- **Değişkenler (Kısıt ve amaç fonksiyonunu oluşturan unsurlar):**

Değişkenler, problemde yer alan bilinmeyenleri, yani sonucunu aradığımız unsurları ifade eder. İşletme problemlerinde genellikle üretim hacmi, makinelerin çalışma süreleri, üretimde kullanılan hammadde miktarları ve üretimde yapılan giderler değişkenler arasında yer alır. Bu değişkenleri belirlerken üretimde yapılacak her hangi bir değişikliğin modele de yeni değişkenler getireceğinin dikkate alınması, değişkenler için kabul edilen ölçütlerin aynı olması gerekir.

- **Parametreler (Değişkenlerin değişim aralığı):**

Parametreler belirlenirken, problemi etkileyecek üretim süresi, birim maliyet, birim hacim vb. faktörlerin modelde matematiksel olarak ifade edilmesi gereklidir. Bir birim üretim için üretimde kullanılan makinelerin, makine ile işlem süresi ilişkisinden yaralanılarak bazı parametreler belirlenir. Aynı zamanda üretim faktörlerinin bileşim oranları olan teknik üretim katsayıları yardımıyla, değişkenler arasındaki ilişkiyi kuran parametreler belirlenir.

- **Kısıtlar (Eldeki olanaklar):**

İşletmeler faaliyetlerini sınırlı kaynaklarla sürdürürler. Bu sınırlı kaynaklar; finansal olanaklar, hammadde miktarı, işçi sayısı, makine sayısı, depo hacmi vb. olup kısıt olarak adlandırılır. Topçu (2005) ise kısıdı, "soruna özgü durumların getirdiği sınırlardır" diye tanımlamaktadır.

İşletmenin sahip olduğu kaynakların miktarı b_i ile ürünlerin seçenekli üretim yolları ya da ikame oranı da a_{ij} sembolüyle gösterilir (a_{ij}, bir birim j ürünü üretmek için b_i' den gerekli olan miktarı gösterir). Buna göre kısıtların genel gösterimi aşağıdaki gibidir.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\ x_j \leq (=;\geq) b_i \quad \begin{matrix} (i = 1,2,\ldots,m) \\ (j = 1,2,\ldots,n) \end{matrix}$$

- **Negatif Olmama Koşulu:**

Doğrusal Programlama modellerinde yer alacak faaliyetler işletme için bir anlam taşıması gerektiğinden, değişkenlerin negatif olması söz konusu olamaz. Eğer işletmeler üretimde bulunurlarsa değişkenlerin değerleri pozitif, herhangi bir üretim olmaz ise değişkenlerin değeri sıfır olur. Dolayısıyla doğrusal programlamada yer alacak olan değişkenlerin değeri sıfırdan büyük veya sıfıra eşit olur ve $x_j \geq 0$ olarak denklem sisteminde yer alır .

Bu kısa bilgilerden sonra bir doğrusal programlama modelinin gösterimi ile ilgili şu bilgiler verilebilir:

Modele girecek olan değişkenler x_1, x_2,…, x_n'dir. Değişkenler arasındaki ilişkileri kuran parametreler ise a_{11}, a_{12},…, a_{ij},…, a_{mn} biçiminde gösterilir. Verilen sabit değerler b_1, b_2,…, b_m ile gösterilir. İlişkilerde kullanılan x_1, x_2, …, x_n değişkenleri pozitif veya sıfır olabilirler, ekonomide negatif üretimden söz edilemeyeceğinden değişkenlerin de negatif olmaları mümkün değildir. Değişkenler arasındaki ilişkiler,

genellikle eşitlik ve eşitsizlik halinde gösterilir. Değişkenler arasında kurulan diğer bir doğrusal denklem de amaç fonksiyonudur. Modelin bütün değişkenleri bu fonksiyonda yer alır (m: kısıt sayısı, n: değişken sayısı). Bu bilgilere göre amaç fonksiyonunun kâr maksimizasyonu veya maliyet minimizasyonu olduğu bir doğrusal programlama modeli genel olarak şöyle ifade edilebilir;

Amaç denklemi $\text{Maks}(\text{Min})Z = \sum_{j=1}^{n} c_j x_j \qquad (j=1,2,...,n)$

Kısıtlayıcılar : $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\, x_j \leq (=;\geq) b_i \quad \begin{matrix}(i=1,2,...,m)\\(j=1,2,...,n)\end{matrix}$

Negatif olmama koşulu : $x_j \geq 0$

$a_{ij} > 0$ ise imalat işleminde girdileri gösterir ve *j* faaliyetinde tüketilen *i* malın miktarıdır. $a_{ij} < 0$ ise imalat yerinde çıktıları gösterir ve *j* faaliyetinde üretilen *i* malının miktarını gösterir. $a_{ij} = 0$ ise *j* faaliyetinde *i* malının ne girdisi ne de çıktısı mümkündür. Sağ taraf sabitleri sıfırdan büyük ($b_i > 0$) ise imalat işleminde girdi olarak kullanılan *i* malının miktarını, sağ taraf sabitleri sıfırdan küçük ($b_i < 0$) ise imalat işleminde çıktı olarak kullanılan *i* malının miktarını ifade eder. $x_1, x_2, \ldots, x_n$ kontrol edilebilen karar değişkenlerini ifade etmektedir.

Matematiksel bir modelin çözümünde karşılaşılan sorun ilk önce değişkenlerin belirlenmesi, daha sonra ise amaç ve kısıtlayıcıların bu değişkenlerin bir matematiksel fonksiyonu olarak ifade edilmesi ve tanımlanmasıdır.

Bir doğrusal programlama probleminin çözümünde aşağıdaki sıra izlenir

- Verilerin toplanması,
- Probleme ait modelin kurulması,
- Modelin çözümlerinin araştırılması,
    - Modelin yapısına bağlı olarak çözümler birden fazla olabilir,
    - Modelin hiçbir çözümü olmayabilir,
    - Amaç denkleme aynı değeri kazandıran alternatif çözümler bulunabilir.

Karar problemlerinin çözümünde doğrusal programlama tekniğinin uygulanabilmesi için :

- Amaç fonksiyonu ve kısıtlayıcılar iyi bir şekilde tanımlanmalı,

- Alternatif kararların oluşturulması mümkün olmalı,
- Değişkenler kendi aralarında ilişkili olmalı,
- Kullanılacak kaynakların arzı sınırlı olmalı,
- Değişkenler arasında kurulan bağıntılar doğrusal olmalı,
- Uygulanacak problem kısa dönemli olmalıdır.

Doğrusal Programlama unsurlarının kısaca açıklanmasından sonra modelinin kurulabilmesi ve uygulanabilmesi için kabul edilen temel varsayımların neler olduğu belirtilmelidir. Temel varsayımlar genelde bütün problemlerde kullanılır, bu nedenle varsayımları aşağıdaki gibi özetlemek mümkündür.

- **Doğrusallık:** Kurulacak modelde yer alan girdi ve çıktıların katsayıları arasındaki ilişkinin doğrusal olduğu varsayılmaktadır.
- **Kaynakların sınırlılığı ve sonluluğu:** İşletmeler, sınırlı olan kaynaklarını doğru değerlendirdikleri sürece hedeflerine ulaşabilirler. Buradan yola çıkarak bu varsayım, süreç sayısının, üretim faktörlerinin, alternatif faaliyet sayısının ve kaynak sınırlarının sonlu olması üzerine kuruludur.
- **Negatif olmama (pozitiflik):** Doğrusal programlamada yer alan tüm değişkenlerin değeri sıfır ya da sıfırdan büyük olmalıdır. Bu varsayımın ortaya çıkış sebebi negatif üretim söz edilemeyeceğindendir.
- **Toplanabilirlik:** Her bir karar değişkenin amaç fonksiyonuna olan katkısı, diğer karar değişkenlerin katkısından bağımsızdır ve sistemin toplam çıktısı karar değişkenlerin katkılarının toplamına eşittir.
- **Bölünebilirlik:** Çoğu üretim faaliyetinde üretime giren kaynaklar ile üretim sonucu ortaya çıkan ürünler bölünebilme özelliğine sahiptir. Bu özellik model kurulurken dikkate alınan diğer bir varsayım olan "bölünebilirlik varsayımını" ortaya çıkarmaktadır.
- **Belirlilik:** Doğrusal programlamada birim başına kâr, her faaliyet için gerekli faktör miktarı ve elde edilecek ürün miktarı gibi ekonomik değerlerin sabit olduğu varsayılır.

Tek değerli beklentiler varsayımında ise; kaynak arzı, girdi-çıktı katsayıları ve fiyatların kesin olarak bilindiği kabul edilir.

1.3 Doğrusal Programlama Çözüm Teknikleri

Doğrusal programlama modellerinin başlıca iki temel çözüm yöntemi vardır. Bunlardan birincisi "Grafik Yöntem", ikincisi ise doğrusal programlama modellerinin özel çözüm tekniği olan "Simpleks Metottur."

Grafik yöntem teorik olarak çok sayıda değişkenin yer aldığı problemlerin çözümünde kullanılabilir olsa da pratikte üçten fazla değişken grafik yardımıyla gösterilemez ve çözülemez. Ayrıca ikiden fazla değişken için çözüme ulaşmak çok güç olacağından grafik yöntem çoğunlukla iki değişkenli modeller için kullanılır. Grafik yöntemi, kısıtların çevrelediği alan içindeki optimum çözümü bulmak için amaç doğrusu eğimi değiştirilmeksizin amaç yönünde (maksimum ya da minimum yönde) hareket ettirilmesine dayanmaktadır.

Simpleks metot ise Dantzig tarafından geliştirilmiş olup, çok sayıda karar değişkeni içeren doğrusal programlama problemlerinin genel bir çözüm metodudur.

Bir doğrusal programlama modeli çözüldüğü zaman aşağıdaki dört durumdan biri ile karşılaşılmaktadır.

- Doğrusal programlamanın tek bir optimum çözümü vardır.
- Doğrusal programlamanın alternatif optimum çözümleri vardır.
- Doğrusal programlamanın uygun çözüm alanı yoktur.
- Doğrusal programlama uygun çözüm alanı sınırsızdır.

- **Grafik çözüm yöntemi**

Grafik çözüm yöntemi problemde en fazla üç değişken olması durumunda kullanılabilen bir yöntemdir. Ancak değişken sayısı üç olduğunda üç boyutlu bir grafik ortaya çıkacağından, çözüme ulaşmak zorlaşabilir. Değişken sayısının iki olması halinde, çizilecek grafik iki boyutlu olacağından kolaylıkla çözüme ulaşılır. Dolayısıyla, değişken sayısı arttıkça boyut sayısı da artacağından grafik üzerinde çözümün elde edilmesi güçleşecektir. Bu nedenle grafik yöntemi ikiden fazla değişkenli doğrusal programlama modellerinin çözümünde tercih edilmemektedir.

Grafik yöntemin daha iyi anlaşılması için, öncelikle bu yöntemin uygulanması sırasında kullanılan terimlerin tanımlanmasında fayda vardır.

- **<u>Uygun çözüm alanı</u>:** Ortaya konan problemin kısıtlayıcı koşullarının (kısıtların) hepsini sağlayan noktalar kümesine uygun çözüm alanı denir.
- **<u>Amaç doğrusu</u>:** Amaç fonksiyonunun koordinat düzlemindeki görüntüsüdür.
- **<u>Optimum çözüm</u>:** Uygun çözüm alanı içerisindeki optimum sonucu ifade eder.

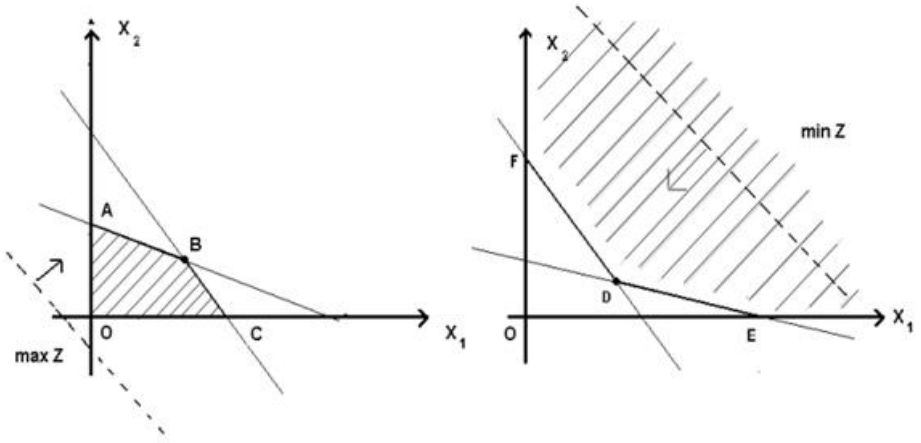
Grafik çözüm yöntemi uygulanırken; kısıtları ifade eden eşitsizlikler koordinat sisteminde gösterilerek uygun çözüm alanı belirlenir. Daha sonra amaç doğrusu problem maksimizasyon ise ∞^+ (pozitif yönde sonsuz)'a, minimizasyon ise ∞^- (negatif yönde sonsuz)'a doğru paralel olarak kaydırılır. Amaç fonksiyonun uygun çözüm alanını terk ettiği nokta optimum çözüm değerini verir. Grafik yönteminin bu özelliği dolayısıyla optimum noktası uygun çözüm alanının köşe noktalarından biridir.

Amaç Denklemi : Maks (Min) $Z = \sum_{j=1}^{n} c_j x_j$ (j: 1,2,3….n)

Kısıtlar

$$: \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\, x_j \leq (=;\geq) b_i \quad \begin{matrix} (i = 1,2,...,m) \\ (j = 1,2,...,n) \end{matrix} \qquad (3.4)$$

$$x_1, x_2 \geq 0$$



Şekil 1 Grafik çözüm yöntemi için bir örnek

Şekil 1'de; OCBA alanı bir maksimizasyon probleminin uygun çözüm alanını göstermektedir ve B noktası maksimum değer yani optimum noktadır. Diğer grafikte ise uygun çözüm alanı pozitif yönde sonsuza doğru genişleyen bir minimizasyon problemidir. Bu grafikte de minimum değer D noktası olup, optimum çözüm D olacaktır.

- **<u>Grafik Çözümde Özel Durumlar</u>**

Grafik çözüm yönteminde aranan optimum nokta her zaman bulunamayabilir. Bu durumları şöyle sıralayabiliriz:

- Birden fazla nokta optimum olabilir. Bunun nedeni ortaya çıkan dışbükey bir şekle sahip uygun çözüm alanının bir kenarı ile amaç doğrusunun eğiminin eşit olmasıdır. Bu durumda amaç doğrusunun uygun çözüm alanını terk ettiği yer, bir nokta yerine bir doğru parçası olur. Dolayısıyla doğru parçası üzerindeki tüm noktalar optimum noktadır.

- Hiçbir optimum nokta olmayabilir. Bunun nedeni de kısıtların kesişim kümesinin olmamasıdır.

- Sonsuz bir uygun çözüm alanının ortaya çıkması durumu söz konusu olabilir. Bu durum genellikle uygun çözüm alanı ∞^+ (pozitif yönde sonsuz)'a doğru genişlemektedir. Ancak pratikte böyle bir sonuca çoğunlukla rastlanmaz.

- **Simpleks metot**

Grafik yöntem sadece iki karar değişkenli problemlerin çözümünde kullanılabildiği için, birçok doğrusal problemin çözümünde ve endüstriyel sistemlerin analizlerinin yapılmasında, problem pek çok değişken ihtiva edeceğinden, yetersiz kalmaktadır. Bu nedenle ikiden fazla karar değişkeninin bulunduğu problemlerin çözümü için başka yollara ihtiyaç duyulmuştur. Bu amaçla Dantzig tarafından ilk kez ABD Hava Kuvvetleri'nin planlanmasında kullanılan Simpleks metot geliştirilmiştir.Bu yöntem günümüzde üretim programlarının hazırlanmasında (iş gücü, malzeme, siparişler, üretim araçları vb. kısıtlar altında kârı maksimize edip maliyetler

minimize etmede), verimliliğe dayalı ücretlendirme problemlerinde, stok problemlerinin çözümünde, malzeme kullanımında, endüstriyel faaliyetlerde kapasite planlarının yapılmasında, petrokimya ve metalürji gibi karışım problemlerinin çözümünde kullanılmaktadır.

Binay ve diğerleri Simpleks metodu tanımlarken, yöntemin optimal çözüm elde edilinceye kadar bir prosedürün sistematik bir şekilde tekrarlanmasından oluşan bir süreç olduğu ifade etmekte ve aslında metodun bir algoritma olduğunu belirtmektedirler. Bir başka deyişle, Simpleks metotta optimum çözüme ulaşmaya çalışırken çözüm alanı içinde farklı noktalar denenir. Bu deneme sırasında aynı standart hesaplamalar yinelenerek optimum çözüme erişinceye kadar ardı ardına çözümler geliştirilir.Standart hesaplamaların uygulandığı her bir safhaya yineleme (iterasyon) adı verilir.

Simpleks algoritmasına daha detaylı bakıldığında, algoritmanın aşağıdaki aşamalardan oluştuğu görülmektedir.

- Problemde yer alan eşitsizlikler eşitlik haline getirilir. Bunun için kısıtların değişkenler bölümüne temel çözümde yer almayan gölge değişkenler (s_i) eklenir. Bu değişkenler temel çözümde yer almadığı için amaç fonksiyonuna sıfır(0) katsayısı ile birlikte eklenir.
- Başlangıç Simpleks tablosu oluşturulur.

C_j Temel değişken x_j	x_j	s_i	Çözüm vektörü (b_i)
s_i	A	I	B
Z_j			
$Z_j - C_j$			

Şekil 2 Başlangıç simpleks tablosu

j=1,2,…m ve i=1,2…n olmak üzere;

A: Kısıtlar için katsayılar matrisi

I : Birim matris

B : Çözüm vektörü matrisi (sağ taraf sabitleri)

x_j:j. Karar değişkeni

s_i : i. Gölge değişken (kullanılmayan üretim faktörleri ve boş kapasiteyi ifade eder)

C_j: Amaç fonksiyon katsayılar kümesinin j'inci elemanı

Z_j: j'inci sütuna göre amaç fonksiyonun değeri

- Amaç fonksiyonunda çözüme girecek değişken işleme sokulur. Bu adımda anahtar (pivot) sütun belirlenir. Amaç maksimizasyon ise $Z_j - C_j$ 'nin en küçük negatif değeri, amaç minimizasyon ise $Z_j - C_j$ 'nin en büyük pozitif değere sahip sütun seçilir.
- Çözümden çıkacak değişken belirlenir. Çözüm sütunundaki (b_j) elemanlar pivot sütunda kendisine denk gelen elemanlara bölünür, bölüm sonucu negatif veya ∞ değerli elemanlar dikkate alınmadan (b_j / a_{ij}), en küçük olan seçilir ve çözümden bu satırdaki değişken çıkarılır.
- Yeni sıra hesaplanır. Bunun için anahtar satır ile anahtar sütunun kesiştiği noktada bulunan değer (pivot sayı), bütün satırı böler. Daha sonra diğer satırların aynı sütunu sıfır olacak şekilde anahtar satır elemanları bir katsayı ile çarpılarak diğer satırlardan çıkarılır. Böylece birim matris yeniden elde edilirken anahtar sütun birim matrise dâhil olur.
- Optimum çözüme ulaşılırken yukarıdaki işlemler amaç fonksiyonu maksimizasyon ise $Z_j - C_j \geq 0$ oluncaya kadar, amaç fonksiyonu minimizasyonsa $Z_j - C_j \leq 0$ oluncaya kadar tekrarlanır.

Simpleks metot, yukarıda açıklanan algoritma dışında doğrusal cebir yöntemleri kullanılarak da çözülebilmektedir. Daha öncede belirttiğimiz gibi bir doğrusal programlama modeli aşağıdaki gibi gösterilir.

Amaç fonksiyonu : Maks (Min) Z $= \sum_{j=1}^{n} c_j x_j$

Kısıtlar : $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_j \leq b_i$ i: 1,2,…..m j: 1,2,…..n

$x_j \geq 0$ (3.5)

Bu durumda n tane değişken ve m tane kısıt var demektir. m<n olduğunda kısıtların ortak çözümünü bulmak olanaksız olacaktır. Bu durumda n-m(n'nin m'den farkı) tane gölge değişken kısıtlara eklenerek eşitsizlikler eşitlik haline getirilir. Daha sonra standart haldeki problemi aşağıdaki gibi matris halde gösterebiliriz.

Maks (Min) Z = CX

(AI)X=b

X ≥ 0

I: Birim matrisi (m×m boyutlu)

A: Katsayılar matrisi

B: Sağ taraf sabitleri matrisi

X: Değişkenler kümesi

C: Amaç fonksiyonu katsayılar kümesi olmak üzere;

$X=(x_1, x_2, \ldots . x_n)$

$C=(c_1, c_2, \ldots . c_n)$

$$A=\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,n-m} \\ a_{2,1} & a_{2,2} & a_{2,n-m} \\ & & \\ a_{m,1} & a_{m,2} & a_{m,n-m1} \end{bmatrix} \qquad b=\begin{bmatrix} b_1 \\ b_2 \\ \\ b_m \end{bmatrix}$$

Olduğundan kısıt denklemlerini;(AX=b)

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,n} \\ & & \\ a_{m,1} & a_{m,2} & a_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ b_n \end{bmatrix}$$ şeklinde gösterebiliriz.

I_{mxm} birim matrisi, değişkenlerde uygun düzenlemeler yapıldıktan ve gerekiyorsa yapay değişkenler kullanıldıktan sonra, değişkenlerin sol tarafının daima en doğru durumu almasını garanti eder, yani Simpleks çözümü için başlangıç tablosunun oluşmasını sağlar.

Bundan sonra X_B'yi X vektörünün m elemanlı alt kümesi olarak tanımlanır. Bu durumda (AI)'nın X_B elemanlarına karşılık gelen vektörlerden oluşan B_m matrisi

olarak kabul edilerek, X'in geriye kalan elemanlarına sıfır atanır ve (AI)X=b $\longrightarrow$ B X_B=b indirgemesi yapılır.

B bir temel ise, $X_B = B^{-1}$b çözümü elde edilir. (B^{-1}, B'nin tersi anlamına gelmektedir.) Maks Z = CX, (AI)X=b, X≥ 0 olup X vektörünün X_I ve X_{II} olarak bölündüğünde başlangıç temel çözümü olan B=I'ya karşılık gelir. C vektörü de X_I ve X_{II}'ye karşılık gelecek şekilde ikiye bölünür. Böylece problem aşağıdaki hali alır.

$$\begin{bmatrix} 1 & -C_I & -C_{II} \\ 0 & A & I \end{bmatrix} \begin{bmatrix} z \\ X_I \\ X_{II} \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

X_B: Temel vektör

$C_B : X_B$ ile ilişkili amaç fonksiyonu katsayıları olsun.

Temel dışı değişkenler sıfır olduğundan B X_B=b Z=C_B X_B'ye indirgenir.

$$\begin{bmatrix} Z \\ X_B \end{bmatrix} = \begin{bmatrix} 1 & -C_B \\ 0 & B \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix} = \begin{bmatrix} C_B B^{-1} b \\ B^{-1} b \end{bmatrix}$$ işlemler yapılınca

$$\begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 1 & -C_I & -C_{II} \\ 0 & A & I \end{bmatrix} \begin{bmatrix} z \\ X_I \\ X_{II} \end{bmatrix} = \begin{bmatrix} 1 & C_B B^{-1} \\ 0 & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix}$$

bu işlem yapıldığında Simpleks tablosu aşağıdaki şekilde elde edilir.

Temel değişkenler	X_I	X_{II}	Çözüm
Z	$C_b B^{-1} A - C_I$	$C_b B^{-1} A - C_{II}$	$C_b B^{-1} b$
X_B	B^{-1}A	B^{-1}	$B^{-1} b$

Şekil 3 Simpleks algoritmasının çözüm matrisi

Özetle, yukarıda yer alan çizelgede B matrisinin bilinmesi uygun çözümü bulmak için yeterli olacaktır.

- **Büyük M yöntemi**

Büyük M Yöntemi'nde M'e verilen değerin çok büyük olması hatalara sebep olduğundan, sorunu aşmak için doğrusal programlama problemi iki aşamada çözülebilir. Bu yöntem aşağıdaki açıklanan yolla uygulanmaktadır.

Eğer bir doğrusal programlama probleminde ≥ veya = kısıtları varsa, Simpleks metot kullanılarak bir başlangıç temel uygun çözümü bulunamazsa, bu durumda yapay başlangıç çözümü uygulanır bunun için büyük M veya iki aşamalı Simpleks metot adı verilen iki yöntem kullanılmaktadır.

Büyük M yöntemi; standart olmayan ve uygun başlangıç çözümü olmayan durumlarda kullanılır. Bu yöntemde, kısıtlar ≥ durumunda ise bu kısıtları standart hale getirmek için negatif katsayılı gölge değişken eklenir, ancak gölge değişkenin negatif katsayılı olmasından dolayı birim matris oluşturulamayacağından, bu matrisi oluşturmak için ayrıca yapay değişken (Ri) eklemek gerekir. Böylece birim matris oluşur. Yapay değişkenler probleme sonradan girdiklerinden, sıfır değerini alarak çözüm dışı kalmalarını sağlamak için amaç fonksiyonunda bu değişkenlere $M \to \infty$ olacak şekilde bir ceza katsayısı verilir. Bu nedenle bu yöntemin adı bazı kaynaklarda penaltı metodu veya ceza metodu olarak da geçer. Verilen bu katsayı amaç maksimizasyon iken -M, amaç minimizasyon iken + M olur. Bundan sonra problemin çözümünde Simpleks metot aynen uygulanır

Simpleks tablosu ise Şekil 4'de ki gibi oluşturulur.

Temel değişken	x_i	s_i	R_i	Çözüm vektörü (b_i)
s_i	A	I		B
Z_j			$\pm M$	
$Z_j - C_j$			$\pm M$	

Şekil 4 Simpleks tablosu

- **İki aşamalı yöntem**

Büyük M Yöntemi'nde M'e verilen değerin çok büyük olması hatalara sebep olduğundan, sorunu aşmak için doğrusal programlama problemi iki aşamada çözülebilir. Bu yöntem aşağıdaki açıklanan yolla uygulanmaktadır.

I. aşama: Başlangıç çözüm için kısıtlara gerekli yapay değişkenler eklenir ve yapay değişkenlerin toplamını minimize edecek yeni formda bir amaç denklemi yazılır. Amaç fonksiyonu denkleminin minimum değeri sıfır ise problemin uygun çözümü vardır.Bu durumda II.aşamaya geçilir.Amaç denkleminin değeri pozitif ise problemin çözümü yoktur.

II. Aşama: I. aşamada elde edilen optimum temel çözüm kullanılarak problemin çözümü hesaplanır.

- **Dualite**

Bir DP probleminde amaç maliyet minimizasyonu veya kâr maksimizasyonu iken, dualitede amaç kaynakların en verimli şekilde planlanmasıdır. Örneğin; bir işletmenin kaynaklarını (hammadde, işgücü, üretim araçları) satın almak isteyen girişimci için gerekli olan model, işletmenin üretim faaliyeti için geliştirilen modelin dualidir.Yani işletmenin kısıtları dual modelde işletmenin imkânları haline dönüşür.

Bir çok programlama işleminde dual; optimizasyonun anlamına (maksimizasyon veya minimizasyon), kısıtların tipine ($\leq$, $\geq$ veya =) ve değişkenlerin işaretine (negatif olmama veya sınırlandırılmama) bağlı olarak primalin çeşitli durumları için tanımlanmıştır. Başlangıç simpleks tablosunu oluşturmak için daima standart hal kullanılır ve primal modelin optimum çözümü dual modelin de çözümüdür. Dolayısıyla standart primalden duali tanımlamak suretiyle, dual çözüm simpleks metot hesaplamalarıyla gerçekleştirilir. Dual problemin kısıtları ve değişkenleri simetrik olarak primal problemlerden aşağıdaki gibi oluşturulabilir:

- Primal modeldeki her değişkene karşılık dual modelde bir kısıt vardır ve j'inci dual kısıt j'inci primal değişkene karşılık gelir.
- Primal modeldeki her kısıta karşılık dual modelde bir değişken vardır ve i'ninci dual değişken i'ninci primal kısıta karşılık gelir.
- Dual modelin amaç fonksiyonu katsayıları primal modeldeki kısıtların sağ taraf değerleridir.
- Dual modelin kısıt katsayıları primal modelin kısıt katsayılarının transpozudur.

Bu aşamada primal-dual ilişkisindeki diğer husulara değinmekte fayda vardır: Matris katsayıları veya optimizasyon modelinin eşitlikleri ve eşitsizlikleri bağlantılı bir ilişki içinde bulunmaktadır. Matematik olarak, problem 90º döndürülmüş matris ile karakterize edilebileceği ve çözüme kavuşturulabileceği şekilde ifade edilebilir. Böyle bir matriste maksimizasyon problemi, minimizasyon problemi (veya tersi) olarak çözülebilir.

		Primal Model Maks (Min) Z					
		$x_1 \geq 0$	$x_2 \geq 0$	$X_3 \geq 0$	...	$x_4 \geq 0$	STD
Dual model Min (Maks) W	$y_1 \geq 0$	a_{11}	a_{12}	a_{13}	...	a_{1n}	$\leq b_1$
	$y_2 \geq 0$	a_{21}	a_{22}	a_{23}		a_{2n}	$\leq b_2$
	$y_3 \geq 0$	a_{31}	a_{32}	a_{33}	...	a_{3n}	$\leq b_3$
	.	.	.	.		.	.
	.	.	.	.		.	.
	.	.	.	.		.	.
	$y_1 \geq 0$	a_{m1}	a_{m2}	a_{m3}	...	a_{mn}	$\leq b_m$
	STD	$\geq c_1$	$\geq c_2$	$\geq c_3$	...	$\geq c_n$	

Şekil 5 Prime-Dual İlişkisi

Primal problemin dual probleme dönüştürülmesiyle ortaya şu ilişkiler çıkar.

- Primal modeldeki amaç fonksiyonu maksimum ise duali minimum, primal modeldeki amaç fonksiyonu minimum ise dual modeldeki amaç fonksiyonu maksimumdur.
- Maksimizasyon primal problemindeki kısıtlayıcıların yönü "≤" ise minimizasyon dual probleminde kısıtlayıcıların yönü "≥"tir. Minimizasyon primal probleminde kısıtlayıcıların yönü "≥" ise maksimizasyon dual probleminde kısıtlayıcıların yönü "≤"tir.
- Primal modeldeki eşitsizliklerin sağ tarafında yer alan b_i sabitleri, dual modelde amaç fonksiyonunun katsayıları olarak yer alır. Dual modeldeki kısıtlayıcıların sağındaki b_i sabitleri de primal modeldeki amaç fonksiyonunun katsayılarıdır.
- Kısıtlayıcı denklemlerin katsayıları aynı kalırken sadece dönüşüme uğrarlar. Diğer bir ifadeyle primal problemin kısıtlayıcılarının solundaki satır katsayıları dual problemin kısıtlayıcılarının sütun katsayıları olur.

$$\text{Primal problemin kısıtlayıcı katsayıları} \begin{bmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \end{bmatrix}$$

$$\text{Dual problemin kısıtlayıcı katsayıları} \begin{bmatrix} a_{11}a_{21} \\ a_{12}a_{22} \\ a_{13}a_{23} \end{bmatrix} \text{ şeklindedir.}$$

Buradan hareketle;

- Dual değişkenlerinin sayısı primal problemin kısıtlayıcı denklem sayısına eşittir. Dual problemin kısıtlayıcı sayısı ise primal problemin değişken sayısına eşittir.
- Her iki problemde de yer almakta olan değişkenler negatif değildir (0 veya pozitiftir).
- Primal problemin optimum çözümü varsa dual problemin de optimum çözümü vardır. Herhangi birisinde optimum çözüm yoksa diğerinde de yoktur.
- Primal modelde optimum çözümü veren tablodaki gölge değişkenlerin $Z_j–C_j$ satırlarındaki mutlak değerleri dual modelde optimum çözümü veren tablodaki çözüm vektörünün değerini verir.
- Primal problemin optimum çözüm tablosundaki çözüm vektöründeki değerler, dual problemin optimum çözüm tablosunda $Z_j–C_j$ satırındaki gölge değişkenlerin değerini verir.

Standart şekildeki primal problem dual şekle dönüştürüldüğünde ortaya yeni ilişkiler çıkmaktadır. Bunlar sırasıyla:

- Primal problem maksimizasyon amaçlı kısıtlayıcı denklemi eşitlik halinde ise dual problem minimum amaçlı olacağı gibi, kısıtlayıcı denklemlerin yönü ($\geq$) olacaktır.
- Primal problem minimizasyon amaçlı ve kısıtlayıcı denklemleri eşitlik halinde ise dual problem maksimizasyon amaçlı olacağı gibi kısıtlayıcı denklemlerin yönü ($\leq$) olacaktır.
- Primal problemlerin değişkenleri sınırlandırılmış işarette ise bunların karşılığı dual kısıtlayıcıları eşitlik halinde olacaktır.

- Primal problemin kısıtlayıcıları eşitlikte ise buna karşılık olan dual değişkenler sınırlandırılmamış işarette olacaktır.
- Simetriden dolayı dual problemin duali primal olacaktır.

- **Dual simpleks yöntem**

Dual simpleks metodun genel düşüncesini, ilk iterasyonun optimum fakat uygun olmayan bir noktadan başlanarak, iterasyon ilerledikçe optimalliğini kaybetmeden uygunluğa doğru hareket etmesi olarak tanımlanabilir.

Birçok durumda amaç fonksiyonu satırındaki tüm katsayıları negatif olmayan (dual uygunluk) ve en az bir sağ taraf değeri negatif olan (primal uygunsuzluk) problemleri çözmek daha kolay olmaktadır. Bu teknik her adımda dual uygunluğu sağlaması nedeniyle dual simpleks metodu olarak adlandırılır.Bu yöntemin algoritmik adımları aşağıdaki gibidir.

- **1.Adım:** Başlangıç simpleks tablosunu oluşturulur. Eğer tüm sağ taraf sabitlerinin değerleri negatif değilse simpleks metod uygulanır, aksi halde 2. adıma geçilir.
- **2. Adım:** En küçük değerli negatif sağ taraf sabiti belirlenir ve bu satıra denk gelen değişken çözümden çıkarılır.
- **3. Adım:** $Z_j - C_j$ satırındaki değerler çözüm dışına çıkacak değişken satırındaki negatif ($a_{ij}<0$) değerlere bölünür ($a_{ij}<0$ koşuluna uyan değer yoksa optimum çözümde yoktur), bu oran içinden en küçük olan değişken çözüme girer.
- **4. Adım:** Çözümden çıkan değişken ile çözüme giren değişkenin kesiştiği noktadaki eleman pivot elemandır. Yeni tablo normal simpleks metotla oluşturulur.
- **5. Adım:** Eğer tüm sağ taraf sabitleri negatif değilse klasik simpleks metot uygulanır, aksi durumda 2. adım'a dönülür.

1.4 Duyarlılık Analizi

Yöneylem Araştırmasında parametrelerin değişiminin, genel çözümü ve kararları nasıl etkileyeceğinin araştırılmasına duyarlılık analizi denir. Bir başka değişle duyarlılık analizi araştırmacılar için optimum çözümdeki parametrelerin değişmesi

durumunda doğrusal programlamanın çözümünün nasıl değiştiğini görme fırsatı veren önemli bir araçtır.Bakır ve Altunkaynak ise duyarlılık analizini, “amaç fonksiyonu katsayıları, kısıtlayıcı kaynak miktarları ve teknolojik parametrelerdeki değişmelere karşı optimum çözümün göstereceği tepkidir” şeklinde tanımlamışlardır.

Binay ve diğerleri , duyarlılık analizi yapılmasının başta zaman ve emek olmak üzere birçok kaynağın gereksiz bir şekilde kullanılmasını engelleyeceğini belirterek duyarlılık analizinin önemine vurgu yapmışlardır. Duyarlılık analizinin hedefi, mevcut çözüm değişmeden modelin katsayılarında değişiklik olup olmadığını tespit etmek ve eğer değişiklik söz konusu ise yeni bir optimuma etkili bir şekilde nasıl ulaşılacağını hesaplamaktır. Duyarlıklık analizi, doğrusal problemin mevcut optimum çözümüne ulaşıldıktan sonra uygulanmaktadır.

Taha ’ya göre modeldeki değişimlerin sonuçları dört başlık altında toplanabilir:

- Temel çözüm değişmeden kalır.
- Mevcut çözüm uygun olmayan hale gelir.
- Mevcut çözüm optimum olmayan hale gelir.
- Mevcut çözüm hem uygun olmayan hem de optimum olmayan hale gelir.

Birinci durumda çözüm değişmediği için herhangi bir yöntem kullanmaya gerek duyulmaz. İkinci durumda, uygunluğu sağlamak için dual simpleks metot uygulanır. Üçüncü durumda yeni optimuma ulaşmak için simpleks metot kullanılır, dördüncü durumda ise yeni çözüme ulaşmak için primal ve dual yöntemlerin ikisi de kullanılır. Aslında dördüncü durum, ikinci ve üçüncü durumların birleşimidir.

- **Uygunluğu etkileyen değişiklikler**

Optimum çözümün uygunluğunu, kısıtların sağ tarafının değişmesi ya da modele yeni bir kısıdın eklenmesi etkiler. Dolayısıyla uygunluğu sağlamak için dual simpleks metot kullanılmalıdır (Taha, 2000). Yeni bir kısıt eklenmesi, çözümün optimumluğunu değiştirebilir. Bu durumda yeni kısıtlayıcı için optimum çözüm sağlanıp sağlanmadığının kontrol edilmesi ilk işlem olmalıdır. Eğer bu koşul sağlanıyorsa çözüm değişmeden optimum kalır ve yeni kısıtlayıcının eklenmesi fazlalık gösterir. Eklenen yeni kısıtlayıcı eldeki optimum çözümü sağlamıyorsa, gerekli gölge ve

yapay değişkenler eklenmiş olarak değiştirilmiş optimum tabloda görülür. Ulaşılan yeni çözüm tablosu uygun bir çözüm değilse, dual simpleks metot uygulanarak optimum uygun çözüme ulaşmaya çalışılır.

- **Optimumluğu etkileyen değişiklikler**

  Temel çözümün optimumluğu sadece amaç fonksiyonu katsayılarının optimumluk koşullarından uzaklaşması halinde sona erecektir. Bu durum temelde olmayan değişkenlerin yeniden hesaplanmasını gerektirecektir. Temel değişkenler ise satır elemanları sıfır olacağı için dikkate alınmaz. Taha hesaplama işlemleri yapıldıktan sonra aşağıdaki durumların oluşabileceğini belirtmektedir:

  - Optimumluk koşulu sağlanmışsa mevcut çözüm aynı kalır, fakat amaç fonksiyonu yeni bir optimum değer alır.
  - Optimumluk koşulu sağlanmamışsa, optimumluğu sağlayacak simpleks metot uygulanır.

1.5 Doğrusal Programlama Çözücüleri

LINGO: Lingo, doğrusal, doğrusal olmayan (dışbükey ve dışbükey olmayan / küresel), ikinci dereceden, ikinci dereceden kısıtlı, ikinci dereceden konik, yarı belirli, stokastik ve tamsayılı optimizasyon modellerini daha hızlı, daha kolay ve daha verimli çözmek için tasarlanmış kapsamlı bir araçtır. Lingo, doğrusal, doğrusal olmayan ve tamsayılı problemleri hızla okunabilir bir formda formüle etmeye olanak sağlamaktadır. Doğrudan veritabanından veya elektronik tablolardan bilgi alan modeller oluşturmaya olanak sağlar. Modeller Lingo içinde oluşturulabilir, çözülebilir veya doğrudan yazılan bir uygulamadan çağrılabilir (https://www.lindo.com/index.php/products/lingo-and-optimization-modeling).

MATLAB: Matlab Optimization Toolbox, doğrusal programlama, karışık tamsayılı doğrusal programlama, ikinci dereceden programlama, doğrusal olmayan programlama, kısıtlı doğrusal en küçük kareler, doğrusal olmayan en küçük kareler ve doğrusal olmayan denklemler için çözücüleri içermektedir. Sürekli ve kesikli problemlere en uygun çözümleri bulmak, trade-off analizi yapmak ve algoritma ve uygulamalara optimizasyon metodlarını dahil etmek için Matlab çözücüleri kullanılabilir. Toolbox ile parametre

tahmini, bileşen seçimi ve parametre ayarını içeren optimizasyon görevleri tasarlanabilir. Portföy optimizasyonu, kaynak atama ve üretim planlama, çizelgeleme gibi uygulamalarda en uygun çözümü bulmak için kullanılabilir (https://www.mathworks.com/products/matlab.html).

XPRESS: Xpress Optimizasyon paketi - sadece Xpress olarak da adlandırılır – karışık tamsayılı doğrusal problemleri çözmek için tasarlanmış ticari tescilli bir yazılımdır. Xpress, bilgisayar platformlarında çok yaygın kullanılmaktadır ve ayrıca birkaç programlama dili için çağrılabilir bir kütüphane API'si (Application Programming Interface) ve bağımsız bir komut satırı arayüzü de dahil olmak üzere çeşitli arayüzler sunmaktadır (https://www.fico.com/en/products/fico-xpress-optimization).

CPLEX: Genellikle sadece CPLEX olarak adlandırılan IBM ILOG CPLEX Optimization Studio, büyük ölçekli karışık tamsayılı doğrusal problemleri çözmek için tasarlanmış ticari bir çözücüdür. CPLEX, IBM tarafından geliştirilmiştir. Yazılım ayrıca çeşitli arayüzlere sahiptir, böylece çözücü ile farklı programlama dilleri veya programlar ile bağlantı kurmak mümkündür. Bununla birlikte programın tek başına da çalıştırılabilmesi mümkündür (https://www.ibm.com/tr-tr/analytics/cplex-optimizer).

GUROBI: GUROBI optimizer, karışık tamsayılı matematiksel optimizasyon problemlerinin yanısıra doğrusal olmayan problemler için de kullanılan modern bir çözücüdür. GUROBI optimizer C dilinde yazılmıştır, tüm bilgisayar platformlarında geçerlidir ve çeşitli programlama dillerinden erişilebilirdir. Standart bağımsız modelleme sistemleri, problemleri tanımlamak ve modellemek için kullanılabilir (http://www.gurobi.com/).

LP_SOLVE: lp_solve, doğrusal (karışık-tamsayılı) programları çözmek için kullanılan ve tamsayılı programları Revize Simpleks ve Dal–Sınır metodu ile çözen ücretsiz ve açık kaynak kodlu programlama çözücüsüdür. Temel olarak lp_solve, karışık tamsayılı programlama problemlerini çözmek için neredeyse her programlama dilinden çağırılabilen API adı verilen bir dizi kütüphanedir (http://lpsolve.sourceforge.net/5.5/).

GLPK: GLPK (GNU Doğrusal Programlama Kiti) paketi, büyük ölçekli doğrusal programlama, karışık tamsayılı programlama ve ilişkili diğer problemlerin çözümü için tasarlanmıştır. GLPK paketi; Primal ve Dual Simpleks yöntemleri, Primal ve Dual İç Nokta yöntemi, Dal–Kesim yöntemi, GNU MathProg için çevirmen, API bileşenlerini içeren ücretsiz açık kaynak kodlu programlama çözücüsüdür (https://www.gnu.org/software/glpk/glpk.html).

CLP: Coin - OR projesi yöneylem araştırması topluluğuna açık kaynak kodlu yazılım oluşturmayı amaçlamaktadır. Coin - OR projesine bağlı olan projelerden bazıları: karışık tamsayılı doğrusal programlama problemlerini çözmek için kullanılan Symphony, doğrusal programlama tabanlı Dal–Kesim kütüphanesi olan CBC, doğrusal optimizasyon problemlerini çözmek kullanılan CLP kütüphanesi şeklinde özetlenebilir (https://www.coin-or.org/).

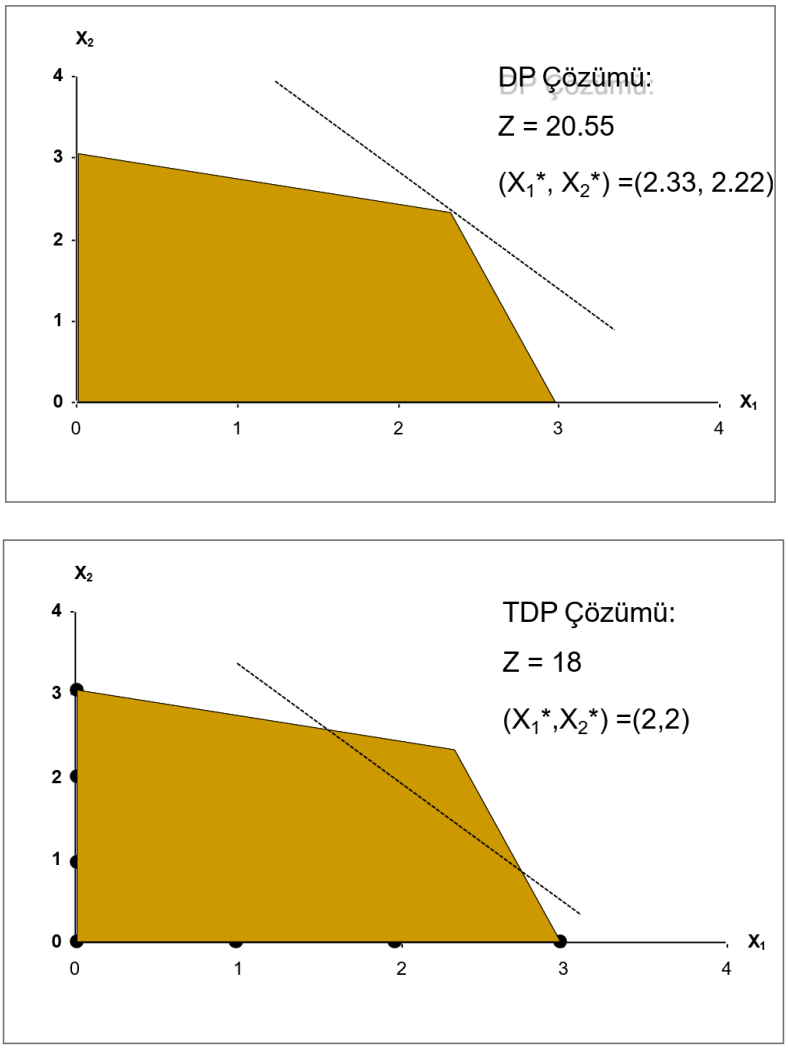
1.6 Doğrusal Programlama Çözücüleri için Kullanılan Platformlar

- RStudio, doğrudan kod yürütmeyi destekleyen bir konsol, sözdizimi vurgulama düzenleyicisi ve çizim, geçmiş, hata ayıklama ve çalışma alanı yönetimi araçları ile R için entegre bir geliştirme ortamıdır (https://rstudio.com/). RStudio ortamında GLPK, LP_SOLVE ve CLP çözücülerini kullanmak için ROI (R Optimization Infrastructure) kütüphanesi mevcuttur.
- MPL (Matematiksel Programlama Dili), model geliştiricinin karmaşık optimizasyon modellerini açık, özlü ve verimli bir şekilde formüle etmesini sağlayan gelişmiş bir modelleme sistemidir. MPL'de geliştirilen modeller, bugün piyasada bulunan çok sayıda ticari optimize ediciden herhangi biri ile çözülebilir. (http://www.maximalsoftware.com/mpl/).
- MPL OptiMax kütüphanesi, optimizasyon modellerini son kullanıcı uygulamalarına entegre etmek için özel olarak tasarlanmış, nesne yönelimli bir bileşen kütüphanesidir. OptiMax tasarımı Microsoft Activex / Automation bileşen yazılım teknolojilerine dayanmaktadır. OptiMax, MPL modellerini Excel / Access için VBA, Visual Basic, Visual C++, Delphi, Java ve Web için standart kodlama dilleri gibi çeşitli farklı programlama platformlarına sorunsuz bir şekilde entegre etmek için kullanılabilmektedir.

İKİNCİ BÖLÜM
2.TAMSAYILI DOĞRUSAL PROGRAMLAMA

Doğrusal programlama sonuçları çoğunlukla tamsayı olmayan pozitif değerlerdir. Ancak gerçek hayat problemleri, insan, makine, hayvan gibi bölünemeyen unsurları içermektedir, bu nedenle tamsayılı sonuçlara ihtiyaç duyulmasından dolayı tamsayılı doğrusal programlama modelleri geliştirilmiştir. Değişkenlerin yalnızca bir kısmının tamsayı değerlerine sahip olması durumunda bu model, karışık tamsayılı programlama problemi olarak adlandırılır. Eğer değişkenlerin tamamı tamsayılardan oluşuyor ise, saf tamsayılı programlama problemi olarak adlandırılır. Yalnızca 0 ve 1 ikili değişkenlerini içeren tamsayılı problemlere ise ikili (veya 0-1 tamsayı) programlama problemi adı verilir. Tamsayılı doğrusal programlamanın en çok kullanıldığı alanlar; iş-işlik (atölye)-makine planlama problemleri, **gezgin satıcı problemleri**, tesis, yerleşim problemleri, sermaye bütçesi planlaması, sabit maliyet problemleri vb.dir.

Bir tamsayılı doğrusal programlama probleminin çözüm kümesi tamsayılar kümesinin bir alt kümesi olmalıdır yani sonuçlar tamsayılı değerler olmak zorundadır.



Şekil 6 DP-TDP çözüm farkı

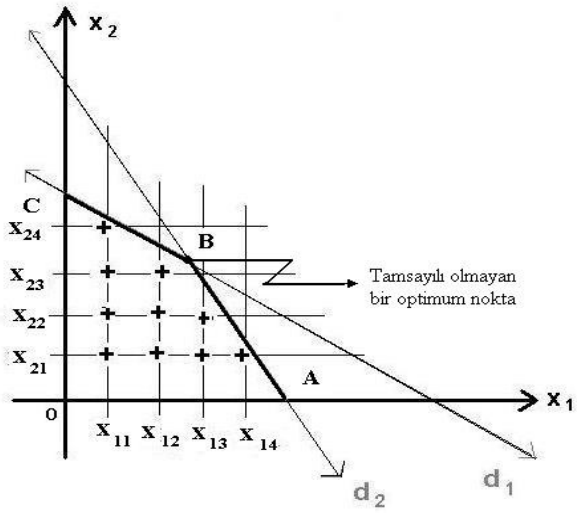
Genel olarak tamsayılı doğrusal programlama modeli aşağıdaki şekilde ifade edilebilir:

Amaç fonksiyonu : Maks (Min) $Z = \sum_{j=1}^{n} c_j x_j$ (j = 1,2,…,n)

Kısıtlar : $\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_j \leq (=;\geq)\ b_i$ $(i = 1,2,...,m)$ $(j = 1,2,...,n)$

$x_j \geq 0$ ve x tamsayı

Yukarıda da ifade edildiği gibi Doğrusal Programlama ile Tamsayılı Doğrusal Programlama modeli arasındaki tek farklılık, karar değişkenlerinin tamsayı olma koşuludur.

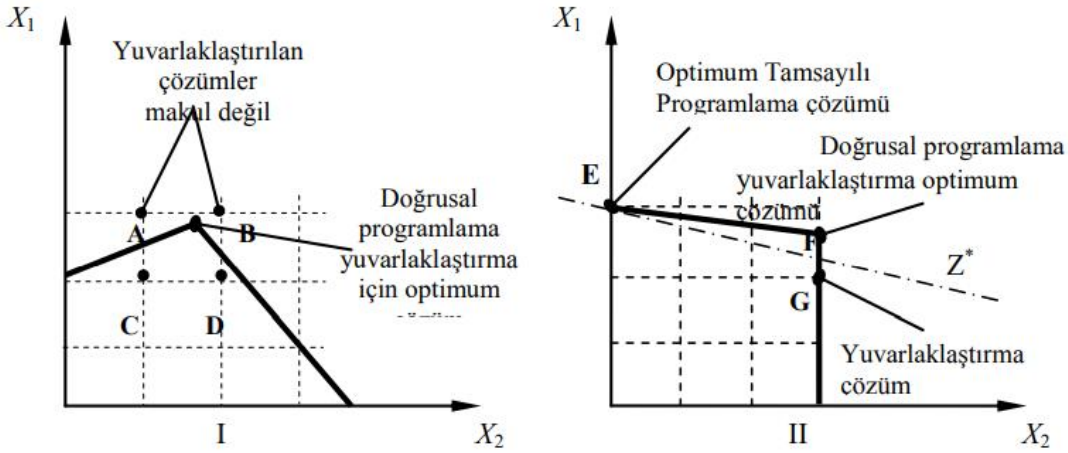


Şekil 7 Tamsayılı doğrusal programlamanın grafik ile gösterimi

Şekil 7.'de görüleceği üzere bu kısıtlar altındaki tamsayılı doğrusal program için uygun çözüm alanı OABC'dir. Bu alan içerisinde bulunan B noktası tamsayılı olmayan optimum bir noktadır. Bu nedenle B noktası çözüm dışı bırakılır. Yeni optimum nokta uygun çözüm alanı içinde bulunan ve "+" ile gösterilen tamsayılı değerler içinde aranır ve bu noktalar dışındaki değerler çözüm dışı kabul edilir (x_{ij}, tamsayılı karar değişkenlerini temsil etmektedir).

Tamsayılı doğrusal programlamadaki metotların açıklanmasından önce, doğrusal programlama yuvarlaklaştırmasına değinmekte fayda vardır. Kesirli değerlerin yuvarlaklaştırılması, tamsayılı doğrusal programlama problemlerinin en basit çözüm yoludur. Bu yaklaşım, doğrusal programlama problemlerinin iki yöntemle tamsayılı hale getirilmesini içerir; bunlardan birincisi, tüm kesirli kısımların çıkarılmasıdır. İkincisi ise daha büyük ya da daha küçük tamsayılı değerden hangisine yuvarlaklaştırılması gerektiğinin araştırılmasıdır. Ancak yuvarlaklaştırmanın bir takım

mahsurları da olabilir ve çözümler makul olmayabilir. Örneğin; Şekil 8'de birinci grafikte A ve B noktaları optimum noktaya en yakın yuvarlaklaştırılan sonuçlar olmasına rağmen uygun çözüm alanı dışında yer almaktadırlar. D noktası ise optimuma daha uzak bir nokta olmasına karşın uygun çözüm alanı içindeki optimum çözümü sağlayabilmektedir. Diğer bir durum ise ikinci grafikte gösterildiği gibi uygun çözüm alanı içinde ve optimum çözüm noktasına en yakın yuvarlaklaştırılmış çözüm noktası G iken, F noktasından daha uzak olmasına rağmen maksimizasyon probleminin çözümünde E noktası optimum nokta olabilir.Bu iki husus yuvarlaklaştırmanın dezavantajları olarak değerlendirilebilir.



Şekil 8 Yuvarlaklaştırmanın grafiksel gösterimi

Öztürk (2002)'e göre "Doğrusal programlama modelinin temel varsayımlarından birisi tüm değişkenlerin sürekli olması ve karar değişkenlerinin değerinin tamsayılı veya kesirli olmasıdır. Bazı işletme problemlerinde tamsayı değerli olmayan çözüm değişkenlerinin ekonomik anlamı yoktur. Girdi ve çıktıların bölünmezlik sorunu karar değişkenlerinin tamsayı değerli olmasını gerektirir."

Tamsayılı doğrusal programlamada, tamsayılı değişkenlerin sayısı, hesaplamaları etkileyen önemli bir faktördür. Değişken sayısının fazlalaştığı durumlarda, örneğin; değişken sayısının 100 olduğu 0-1 TDP probleminde 2^{100} tane alternatif söz konusudur ve bunu hızlı bir bilgisayarın çözmesi asırları bulabilir.Dolayısıyla, problemdeki tamsayılı değişken sayısını azaltmak hesaplama açısından avantajdır. Bütün bunların yanında, günümüzdeki bilgisayar programlarının TDP problemlerinin çözülmesi için yapılan hesaplamalara önemli katkıları vardır.

2.1 Tamsayılı Doğrusal Programlama Modelleri

Tamsayılı doğrusal programlama için karar değişkenlerinin tamamının tamsayı olmadığı durumlar da söz konusudur. Bu yüzden üç çeşit tamsayılı doğrusal programlama modeli vardır;

- Tüm Tamsayılı Programlama Modeli,
- Karma Tamsayılı Programlama Modeli,
- Sıfır – Bir Tamsayılı Programlama Modeli.ge

- **Tüm tamsayılı programlama modeli**

Bu model çözüm sonunda tüm değişkenlerin tamsayılı değerler alması gereken tamsayılı doğrusal programlama modelleridir.Tüm tamsayılı programlama modelini oluşturmak için probleme aşağıdaki gibi bir kısıt eklenir.

$$x_j \geq 0 \text{ ve } x_j : \text{tamsayı}$$

Bu kısıt ile model, optimum çözümü ararken bütün karar değişkelerinin tamsayı olmasını sağlayan çözümü hedefleyecektir.

İster günlük yaşamımız içinde karşılaştığımız problemler olsun, ister büyük sanayi kuruluşlarının çözüm aradığı problemler olsun, çoğunlukla tamsayılarla ifade edilebilecek türden çözümler gerektirmektedir. Örneğin; buzdolabı üretiminde kullanılacak cıvata, düğme, raf, kapak vb. miktarları, bir çiftlikteki tavuk sayısı ve üretilen yumurta miktarı, bir torna veya freze tezgâhında işlenen parça miktarı, bu verilerin tamamı tamsayılarla ifade edilmek zorunda kalınan verilerdir. Bu verileri içeren bir model hazırlanması gerekiyorsa bu tüm tamsayılı programlama modeli olmalıdır.

Tüm tamsayılı programlama modellerinde tamsayı olması gereken değerler x ile ifade edilen karar değişkenleridir. Z ile gösterilen amaç fonksiyonu katsayılarının dolayısıyla amaç fonksiyonun değerinin tam sayılı olması zorunluluğu yoktur.

- **Karma tamsayılı programlama modeli**

Tamsayılı doğrusal programlama probleminde değişkenlerin tümü optimum çözümde tamsayılı değerler almaktadır. Bununla birlikte bazı problemlerde değişkenlerin alt kümesi tamsayılı olmayabilir. Sadece bazı değişkenlerin tamsayı değer aldığı tamsayılı doğrusal programlama problemlerine karma tamsayılı programlama problemleri adı verilir. Bir başka ifade ile bir karma tamsayılı programlama problemi modeli mutlaka aşağıdaki kısıtları içermelidir.

$$j > i \quad olmak\ üzere$$

$$x_j \geq 0 \quad \text{ve} \quad x_i \ \text{tamsayı}$$

Bu kısıtlar ile model, optimum çözümü ararken değişkenlerden bir kısmının (x_i 'lerin) tamsayı olması şartını sağlayan çözümü hedefleyecektir. Böylelikle x_i için tamsayılı bir sonuç ortaya çıkarken, diğer x'ler için kesirli veya tamsayılı bir sonuç elde edilecektir.

- **Sıfır - bir (0–1) tamsayılı programlama modeli**

Birçok durumda bir doğrusal programlamanın değişkenleri, evet/hayır kararları ya da mantıksal ilişkiler şeklinde ifade edilir. Bu değişkenler doğal olarak sıfır ya da bir değerini alırlar ve ikili (binary) değişkenler olarak adlandırılır. İçinde yalnızca ikili değişkenler barındıran tamsayılı doğrusal programlar, ikili tamsayılı programlar olarak ifade edilir.

Değişkenlerin değerlerinin yalnızca 0 ve 1 ile sınırlandırıldığı modellerde toplam değişken sayısının az olduğu durumlarda çözüm kümelerinin tek tek hesaplanması yoluyla optimum çözüm kümesi kolaylıkla bulunabilmektedir. Fakat 0-1 tamsayılı doğrusal programlama modelinin muhtemel çözümlerinin 2^n adet olduğu göz önüne alındığında uygun çözümü bulmak için tüm seçeneklerin birer birer incelenmesinin ardından optimum çözüme ulaşmak uzun zaman alacaktır.Bu nedenle, 0-1 tamsayılı programlama problemlerinin çözümü için bazı özel yöntemlere ihtiyaç vardır. Söz konusu özel çözüm yöntemlerinin ilki Balas tarafından 1965 yılında geliştirilmiştir. Daha sonraki yıllarda Glover ve Geoffrion'un kısmi sayımlama ile özel algoritmalar geliştirdikleri bilinmektedir. Bugün 0-1 tamsayılı doğrusal programlamanın temelini Balas algoritması oluşturmaktadır.Balas algoritması, optimum ama uygun çözüm alanı dışındaki bir noktanın başlangıç

çözümü kabul edilmesiyle başlar ve bütün kısıtlar 0 (sıfir)'dan küçük olacak şekilde düzenlenir. Ardından gölge değişkenler eklenerek eşitlik haline getirilir. Daha sonra problem değişkenleri dallara ayrılarak 0 veya 1 değerleri verilerek optimum çözüme ulaşılmaya çalışılır.

Yöntemin uygulanmasında, x_j 'ler 0 veya 1 değerlerini aldıkları için amaç fonksiyonunda 1 değerini alanların katsayıları arasında toplama-çıkarma işlemi yapılır, 0 değerini alanlar ise çözüm dışı kaldıkları için hiçbir işlem görmez. Çözümde yalnız toplama ve çıkartma işlemleri yapılmasından dolayı bu yaklaşıma "Toplamlı Algoritma" veya "Balas'ın Algoritması" denilmektedir.

0-1 tamsayılı doğrusal programlama modeli problemi aşağıdaki gereksinmeleri karşılayacak bir biçimde ortaya konmalıdır;

- Amaç fonksiyonu, bütün katsayıları negatif olmama koşuluna uyan bir optimizasyondur.
- Kısıtlarının tümü (≤) tipinde olmalıdır. Gerekirse sağ taraflar negatif olabilir. Daha sonra bu kısıtlar, sol taraflarında sürekli gölge değişkenleri kullanmak yoluyla eşitlik haline getirilir (Kara, 1986).

Bu tezin uygulama kısmında ele alınan problemin yapısı 0-1 TDP'ye uygun yapıda olmadığından, Balas algoritması da denen 0-1 TDP modeline çözüm yöntemleri kısmında ayrıca değinilmeyecektir.

2.2 Tamsayılı Doğrusal Programlamada Çözüm Yöntemleri

Sanayinin gelişmesi ve endüstriyel faaliyetlerin ihtiyaçlarından dolayı Doğrusal Programlama üzerine yapılmış olan çalışmalarda tamsayılı değişkenleri içeren problemlerin çözüm ihtiyacı artmıştır. Bu çaba bilim adamlarını Tamsayılı Doğrusal Programlama problemlerinin çözüm algoritmalarına yönlendirmiştir.

Doğrusal programlama problemlerinin tamsayılı çözümlerinin elde edilmesini sağlayan hesaplama yöntemi ilk kez Gomory tarafından 1958 yılında ortaya konmuştur. Gomory'nin geliştirdiği bu hesaplama yöntemine Tamsayılı Algoritma veya Kesme Düzlemi Yöntemi adı verilmiştir. Bu yöntemde sonlu sayıda işlemden sonra optimum bir tamsayılı çözüm ortaya çıkmaktadır. Diğer taraftan bu yöntemin optimum çözüme ulaşmada simpleks metottan daha fazla işlemi gerektirdiği görülmüştür.

Gomory'nin geliştirmiş olduğu algoritmanın altında yatan teori normal simpleks metotta yapıldığı gibi bir kısıtı bir sonraki uç noktaya taşımak yerine, kısıtı (ya da kesmeyi) sisteme ekleyip uygun alanı küçültülmek ve aynı zamanda yeni bir alternatif uç nokta yaratmaktır.

Bilindiği üzere simpleks metot, çözüm uzayının uç noktasında olan optimum değeri bulmayı amaçlayan bir yöntemdir. Sonsuz sayıdaki çözüm seçeneğinden sonlu sayıdaki çözüm alternatiflerine yönelerek tek bir çözüme ulaşmak yöntemin gücünü göstermektedir. Tamsayılı doğrusal programlamada ise, sonlu sayıdaki noktalardan yola çıkılarak çözüme başlanır ve çözümü sağlayan tamsayılı noktalar arasından en uygun olanı seçilmeye çalışılır. (Özkan, 1998).

Tamsayılı doğrusal programlama algoritmaları, doğrusal programlamaların başarılı sonuçlar veren hesaplama yöntemlerinden yararlanarak geliştirilmiştir. Tamsayılı doğrusal programlama problemleri birbirinden çok farklı algoritmalarla çözülebilmektedir. McCarl ve Spreen (1997), bazı algoritmaların bir takım problemlerde daha iyi sonuç vermesi sebebiyle algoritma seçimini bir sanat olarak değerlendirmektedirler. Bu algoritmalardaki stratejiler üç adım içermektedir.

1. **Adım:** Herhangi bir 0-1 tamsayılı y değişkenini $0 \leq y \leq 1$ sürekli aralığında değerler alacak şekilde değiştirip, bütün tamsayılı değişkenlerle ilgili tamsayı olma kısıtlarını da kaldırarak TDP çözüm uzayı gevşetilir. Böylelikle normal doğrusal programlama haline gelmiş olur.

2. **Adım:** Doğrusal programlama problemini çözerek sürekli optimum belirlenir.

3. **Adım:** Sürekli optimumdan başlanıp, tekrarlı bir şekilde özel kısıtlar eklenerek çözüm uzayında düzeltmeler yapılır. Böylelikle, tamsayılı gereksinimleri de karşılayacak bir optimum uç noktaya ulaşılmaktadır.

Üçüncü adımda ifade edilen özel kısıtları oluşturabilmek için iki genel yöntem geliştirilmiştir. Bu yöntemler;

- Kesme Düzlemi Yöntemi
- Dal-Sınır (DS) Yöntemi

İki yöntemden hiçbiri TDP problemlerini çözmede sürekli daha iyi sonuç vermeyecektir. Bununla birlikte, deneyimler dal-sınır yönteminin kesme düzlemi yöntemine göre çok daha başarılı olduğunu göstermektedir.

İlk algoritma olan kesme düzlemi algoritması, Dantzif, Fulkerson ve Johnson, ile Gomory tarafından geliştirilmiştir.Optimum çözümü veren algoritmayı Gomory geliştirmiştir. Gomory'nin yaklaşımı, konu üzerindeki ilk genellemeleri yapmış olması nedeniyle, kaynaklarda "Gomory'nin Kesme Düzlemi Tekniği" olarak da yer almaktadır. Ardından Land ve Doig dal-sınır algoritmasını tanıtmışlardır. Bu iki temel algoritmanın haricinde de yeni geliştirilen farklı türde algoritmalar bulunmaktadır. 20 yıllık tecrübe ve binlerce akademik çalışmaya karşılık ne yazık ki mevcut hiçbir algoritma tamsayılı doğrusal programlama problemlerinin tümüne tatmin edici bir çözüm getirememektedir. Bununla birlikte bazı algoritma türleri kimi problem türlerinin çözümünde iyi sonuç vermektedir.

- **Kesme düzlemi algoritması**

Bu algoritmadaki ana fikir, uygun çözüm alanının dışbükey kümesini değiştirerek optimum noktanın tamsayılı değer almasını sağlamaktır.Bunun için kümeyi tamsayılı değerleri içerde bırakacak şekilde kesecek yeni kısıtlar oluşturulur. Böylece yeni elde edilen uç nokta tamsayılı değer alır. Eklenen kısıtlar (kesmeler) çözüm kümesini daraltacak şekilde en az bir uygun veya uygun olmayan tamsayı noktasını kesmelidir.

Bazaraa ve diğerlerine (1989) göre ise tamsayılı uygun çözüm kümesinin sınırlarını (uç noktalarını) hesaplamak imkânsız değil ama çok zordur. Bu yüzden kesme düzlemi metodunun hedefi tamsayılı optimum çözümün çevresinde iterasyon yapmaktır. Bu iterasyonlarla uygun çözüm alanı içindeki tamsayılı noktalar hariç olmak üzere uygun çözüm alanını bölen ek kısıtlar türetilmiş olunur. Bu yapının amacı tamsayılı optimum noktaları uygun çözüm alanın uç noktaları haline getirmektir. Bu başarıldığı taktirde optimum tamsayılı nokta DP gevşetmesiyle bulunabilir.

Genellikle bu algoritmada birkaç kesmeyle tamsayılı uç nokta elde edilir. Burada oluşturulan kesme sayısı problemin boyutlarından bağımsızdır. Dikkat edilecek husus eklenen kesmeler ile hiçbir tamsayılı uygun noktanın uygun çözüm kümesi

dışında bırakılmaması gerektiğidir. Yani TDP çözümlerinde elde edilen sonuç uygun çözüm kümesinin alt kümesi, uygun çözüm kümesi de tamsayılar kümesinin alt kümesi olmalıdır.

Kesme Düzlemi Algoritması sürekli bir doğrusal programlama probleminin optimum çözümle başlar. Yöntemde kesme denklemi adı verilen özel kısıtlar ardı ardına oluşturularak çözüm uzayının düzenlenmesine gidilir.Eklenen kısıtlar sonucu oluşan uygun çözüm alanı, başlangıç uygun çözüm alanının alt kümesi olmalı ve daha önce çözüme dâhil olan tüm tamsayılı sonuçları kapsamalıdır.

Gomory'nin Kesme Düzlemi yönteminde, başlangıç noktası olarak bir doğrusal programlama probleminin simpleks metotla bulunan optimum sürekli çözümü ele alınır. Eğer bu çözüm tamsayılı bir çözüm değilse, doğrusal programlama problemine ek bir doğrusal kısıtlama dâhil edilir. Bu ek kısıtlama Gomory'nin geliştirdiği "Kesme Düzlemi" kuralına göre elde edilir. Kesme Düzlemi kuralında simpleks metot ile elde edilen optimum çözüm değerlerinden en büyük kesir değerli karar değişkeni seçilir. Sonra da bu değişkenin satırında bulunan değişkenlerin katsayıları tamsayılı ve kesirli olarak yazılır ve tamsayılı değişkenler denklemin sağ tarafında toplanır. Sağ tarafta yer alan tamsayılı değişkenler atılır ve sadece kesirli eleman bırakılır. Tamsayılı değişkenler atıldığına göre eşitlik halindeki değişken eşitsizliğe dönüşecek ve sol taraftaki kesirli kısım sağ taraftaki elemanın değerinden büyük veya eşit olacaktır. Böylece ek kısıtlayıcı elde edilmiştir ve bu ek kısıtlayıcı denklemi standart hale getirmek için yeni bir değişken eklenir. Daha sonra bu ek kısıtlayıcıya tamsayılı olmayan optimum çözüm tablosunda yer verilerek simpleks çözüm işlemine geçilir.

Kesirli kısımdaki değişkenlerin değerlerinden negatif olanları varsa, bu sayının eşleniği bulunarak kendi yerine ikame edilir. İki sayı arasındaki fark bir tamsayı ise bu iki sayıya eşlenik denir. X eşlenik Y, simge olarak X ≡ Y şeklinde gösterilir. Kesme düzlemi matematiksel ifadelerle aşağıdaki şekilde gösterilebilir.

Temel Değişken	$x_1 \dots$	$x_i \dots$	x_m	$w_1 \dots$	$w_i \dots$	w_n	**Çözüm**
Z	**0…**	**0…**	**0…**	$\mathbf{c_1 \dots}$	$\mathbf{c_j \dots}$	$\mathbf{c_n \dots}$	$\mathbf{b_0}$
X_1	1…	0…	0…	$a_{1\,1}\dots$	$a_{1\,i}\dots$	$a_{1\,n}\dots$	b_1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
x_i	0…	1…	0…	$a_{i\,1}\dots$	$a_{ij}\dots$	$a_{in}\dots$	b_i
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
x_m	0…	0…	1…	$a_{m1} \dots$	$a_{m2} \dots$	$a_{mn} \dots$	b_m

Şekil 9 Simpleks tablosu

w_i : i. temel olmayan değişken

x_i : i. temel değişken

b_i : Sağ taraf sabitleri

s_i : i. gölge değişken

f_i : Sağ taraf sabitlerinin kesirli kısmı

f_{ij} : Çözüm simpleks tablosunda temel olmayan değişkenlere denk gelen değerlerin kesirli kısımları

c_j : Amaç fonksiyonu katsayıları

a_{ij} : Çözüm tablosunda temel olmayan değişkenlere karşılık gelen değerler olmak üzere, temel değişken satırını aşağıdaki gibi ifade edebiliriz;

$$x_i = b_i - \sum_{j=1}^{n} a_{ij} w_j$$

b_i ve a_i değerlerinin tamsayılı kısımları ile kesirli kısımlarını toplam şeklinde ifade edelim;

$$b_i = [b_i] + f_i$$

$$a_{ij} = [a_{ij}] + f_{ij}$$

$f_i - \sum_{j=1}^{n} f_{ij} w_j = x_i - [b_i] + \sum_{j=1}^{n} a_{ij} w_j$ bütün w_i, x_i değişkenlerinin tamsayılı olması için sağ taraf sabitleri de tamsayılı olmalıdır. $f_{ij} \geq 0$ ve $w_i \geq 0$ olduğundan $\sum_{j=1}^{n} f_{ij} w_j \geq 0$ olacaktır. Sonuç olarak,

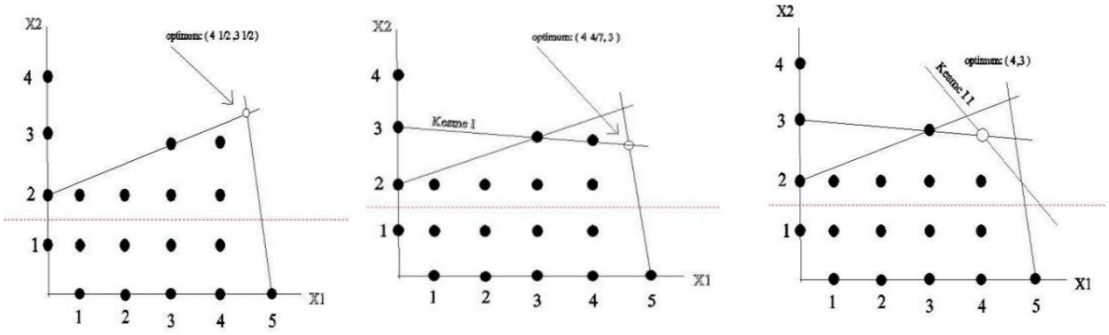
$f_i - \sum_{j=1}^{n} f_{ij} w_j \leq f_i \leq 1$ yazılabilir. $f_i - \sum_{j=1}^{n} f_{ij} w_j$ ifadesi tamsayılı olması gerektiğinden bu ifadeye s_i gibi bir gölge değişken eklenerek 0'a eşitlenir. Böylece bir kesme denklemi de elde edilmiş olunur. Bu kesme aşağıdaki denklemle gösterilir.

$$s_i = \sum_{j=1}^{n} f_{ij} w_j \quad - f_i$$

Kesme düzlemi algoritması adımları:

- **1. Adım:** Orijinal TDP probleminin tamsayı kısıtlarını göz ardı ederek problemin Doğrusal Programlama gevşetmesi simpleks metotla çözülür. Eğer Doğrusal Programlama gevşetmesinin çözümü tamsayı ise TDP probleminin çözümü elde edilerek durulur. Aksi halde 2. adıma geçilir.
- **2. Adım:** Son simpleks tablosunda $f_i>0$ olan i. satır için (birden fazla $f_i>0$ koşulunu sağlayan satır olduğunda en büyük f_i olan satır seçilir) kesme düzlemini tanımlayan bir kısıt eklenerek, simpleks metotla elde edilen tamsayılı olmayan bazı çözümler elimine edilmeye çalışılır. Bununla beraber yeni kesme kısıdı uygun tamsayılı çözümleri elimine etmeyecek biçimde dikkatle oluşturulmalıdır. Bu kısıt daha önce anlatılan matematiksel işlemler sonucu elde edilen kısıt denklemidir.
- **3. Adım:** Yeni TDP problemi, orijinal probleme 2. adımda türetilen kesme düzlemi kısıdını ekleyerek türetilir ve 2. adıma dönülür.Bu sürece devam edilerek kesme düzlemi kısıtlarıyla daraltılmış uygun çözüm alanında nihai olarak TDP probleminin çözümüne ulaşılır.

Kesme Düzlemi Yaklaşımı ile ilgili vurgulanması gereken birkaç nokta daha bulunmaktadır. Eklenen kesmeler, uygun çözüm alanındaki tamsayı noktalarının herhangi birini elimine ederken, bu kesmeler en az bir tane uygun ya da uygun olmayan tamsayı noktasından geçerek uygun çözüm alanını daraltmalıdır. Bunlar, herhangi bir kesme için temel gerekliliklerdir.



Şekil 10 Kesme düzlemi algoritmasının çözüm grafiği

Daha önce ifade edildiği gibi kesme sayısı problemin değişken veya kısıt sayısıyla ilişkili değildir, yani büyük çaplı bir problem birkaç kesmeyle çözülebilirken, küçük çaplı bir problem için onlarca kesmeye ihtiyaç duyulabilir. Çözüm algoritmasına pek çok kesme eklenmesine rağmen model kabul edilebilir bir çözüm sunamazsa başka çözüm yöntemleri denenir. Bu yöntemlerden biri dal-sınır algoritmasıdır.

- **Dal-sınır (DS) algoritması**

Geliştirilmiş olan çözüm metotlarından ikincisi dal-sınır algoritmasıdır. Dal-sınır algoritması Land ve Doig tarafından ortaya atılmış olup böl-fethet stratejisini izlemektir. Bu stratejide orijinal problem çok büyük olduğundan daha küçük alt problemlere bölünerek çözülür. Bölme işlemi tüm geçerli çözümlerin küçük alt kümelere dağıtılması, fethetme ise, alt kümedeki optimum çözümü belirleyip, eğer bu çözüm uygun ama optimum değilse bu alt kümeyi araştırma dışı bırakır

Dal-sınır yöntemi, temelde tüm uygun çözüm seçeneklerini belirlemeye yönelik bir tekniktir. Ancak optimum çözüme götürmeyen bazı çözüm seçenekleri önceden elimine edilmektedir. Bu nedenle çözüm alanı gerekli değerlendirmelerin sayısı kadar küçük alt setlere (kümelere) bölünür. Bu alt setlere "dallandırma noktaları" adı verilir

Dal-sınır yaklaşımı en çok kullanılan genel amaçlı Tamsayılı Doğrusal Programlama çözüm algoritmasıdır. Dal-Sınır Algoritmasında, çözüm prosedürü amaç fonksiyonun değerleri sınırlarla karşılaştırarak gerçekleştirilir. Bunun için önce tamsayı olma koşulu göz ardı edilerek bir başlangıç çözüm elde edilir ve bu çözüm noktası eğer tamsayı değilse bu noktaya en yakın tamsayılı noktalar sınır kabul edilerek bu sınırların dışında uygun çözümler aranır, uygun olmayan alt kümeler elimine edilir.

Bu algoritma düğüm noktalarında uygun çözüm sunabilir, ancak bunlar optimum çözümler olmayabilirler. Düğüm noktalarında bütün tamsayılı uygun çözüm noktaları dikkate alınmalıdır. Bu çözüm noktaları arasından optimum değeri verecek çözüm aranır. Böyle bir çözümün iki değişkenli bir problem için kolayca yapılabileceği açıktır. Ancak çok değişkenli problemler için farklı algoritmalar yardımıyla tamsayılı uygun çözümler karşılaştırılır ve optimum sonuca ulaşılır.

Bu modelin en büyük dezavantajı, her tamsayılı çözüm çiftinin elde edildiği düğümlerde Doğrusal Programlama çözümünün yapılması gerekliliğidir. Diğer yandan dal-sınır algoritması optimum çözümlere genellikle kısa sürede yaklaşmasına rağmen, çözümün optimum olduğunu doğrulama yani onaylama süreci çok zaman alıcıdır.

Aşağıda Tamsayılı Doğrusal Programlamanın dal-sınır yöntemiyle çözümünün matematiksel gösterimi verilmiştir. Ayrıca daha önce ifade edildiği gibi şekildeki dallandırma noktaları yardımıyla çözüme ulaşıncaya kadar algoritma işletilir.

c: Amaç fonksiyonu katsayısı

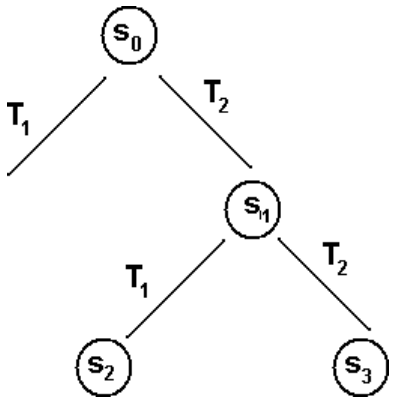
A: Kısıtlar için katsayılar matrisi

b: Sağ taraf sabiti

S_0: Başlangıç çözümü

S_k: k. düğümdeki uygun çözüm

T_i: i. dal



Şekil 11-Dal-Sınır Yönteminin Gösterimi

x_r^*: r. Düğümdeki bir optimum nokta

$$\text{Maks(Min)}\, Z = c \cdot x$$

$x \in S_0$ 'e bağlı olarak

$$S_0 = \{x | Ax = b,\ x \geq 0 \text{ tamsayı}\} \quad \text{iken.}$$

Dal-sınır yönteminin temel fikri ilk olarak problemin bir sürekli model olarak ele alınarak çözülmesidir. Modele karşılık gelen doğrusal programlamanın çözümü;

$$\text{Maks(Min)}\, Z = c \cdot x$$

$$x \in T_0 = \{x | Ax = b,\ x \geq 0\} \text{ 'e bağlı olarak,}$$

şeklindedir. x_r, optimum sürekli değeri x_r^* kesirli olan bir tamsayılı kısıtlanmış değişken olarak varsayılırsa, $\left[x_r^*\right] < x_r < \left[x_r^*\right] + 1$ aralığı her hangi bir uygun tamsayılı çözüm içermemektedir. Sonuç olarak x_r'nin uygun tamsayılı değeri aşağıdaki iki koşuldan birisini yerine getirmelidir. Bu koşullar:

$$x_r \leq \left[x_r^*\right] \quad \text{veya} \quad x_r \geq \left[x_r^*\right] + 1$$

şeklindedir. Bu iki koşul modele uygulandığında aşağıdaki iki küme ortaya çıkarır.

$$T_1 = \{x | Ax = b, x_r \leq \left[x_r^*\right], x \geq 0\}$$

$$T_2 = \{x | Ax = b, x_r \geq \left[x_r^*\right] + 1, x \geq 0\}$$

Kümelere tamsayılı kısıtları da ekleyince:

$$S_1 = \{x | Ax = b, x_r \leq \left[x_r^*\right],\ x \geq 0 \text{ tamsayı}\} \text{ ve}$$

$$S_2 = \{x | Ax = b, x_r \geq \left[x_r^*\right] + 1,\ x \geq 0 \text{ tamsayı}\}$$

haline gelir.

Problemin x^* optimum çözümü ya S_1'de ya da S_2'de yer almalıdır ve aynı zamanda alt problemlerden birisinin optimum çözümü olmalıdır:

$\text{Maks(Min)}\, Z = c \cdot x$ $\quad x \in S_1$ 'e bağlı olarak

$\text{Maks(Min)}\, Z = c \cdot x$ $\quad x \in S_2$ 'e bağlı olarak

Optimum çözüm kesirli değere sahip olduğu zaman, alt problemler bu tamsayılı kısıtı gevşetme süreci tekrarlanarak ve dallandırma yapılarak yeniden çözülebilir.

Dallandırma süreci bir alt probleme karşılık gelen ağacın her bir *k* düğümü ile köklü bir ağaç meydana getirir:

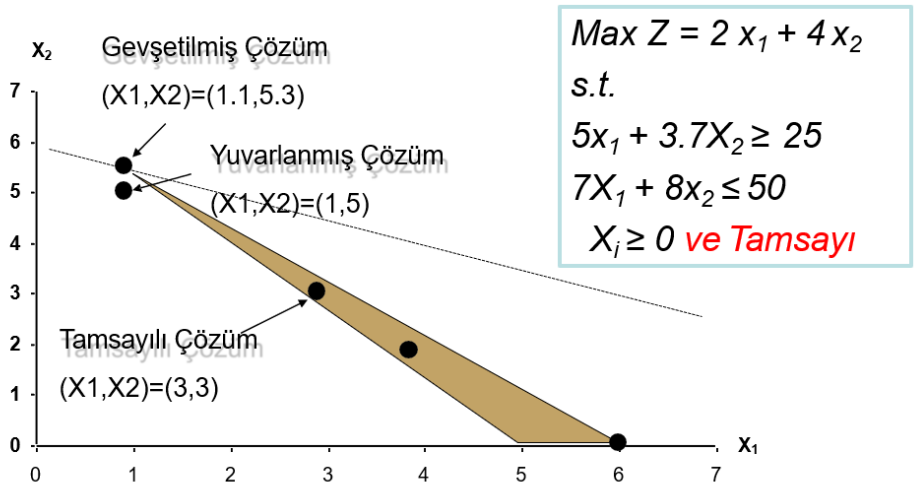
$$\text{Maks(Min)}\ Z = c \cdot x \qquad x \in S_k \text{'e bağlı olarak.}$$

Eğer yukarıdaki amaç fonksiyonuna karşılık gelen Doğrusal Programlamanın optimum çözümü tamsayılı kısıtlara göre uygun çözümse, işlem kaydedilir ve amaç fonksiyonun değeri optimum için alt-sınır olarak oluşturulur (Lawler ve Wood, 1966).

Tamsayılı doğrusal karar modelinde dal-sınır tekniğin uygulaması aşağıdaki algoritmaya göre yapılır.

- **1. Adım:** (Başlangıç) Verilen model değişkenlerin tamsayılı olmasına bakmaksızın çözülür. Sınırsız çözüm ya da uygun çözüm alanının boş küme olması durumunda durulur. Sonuç tamsayı ise yine durulur. Bunların dışındaki durumlarda amaç fonksiyonunun alt ve üst sınırları belirlenerek 2. adıma gidilir.
- **2. Adım:** (Dallandırma): İşlem dışı bir düğümde tamsayı değer alması istenen bir değişkene göre dallandırma yapılarak, alt bölümler elde edilir. Bunlardan biri çözülür. Bulunan optimum çözümde amaç fonksiyonun değeri önceki sınırdan küçükse bu düğüm işlem dışı bırakılır. Amaç fonksiyonunun değeri önceki sınırdan küçük değilse, değişkenler tamsayılı ise 3. adıma değilse 4.adıma geçilir.
- **3. Adım:** (Ayırma): Önceki alt sınır yerine amaç fonksiyonun yeni değeri konur. Bu düğüm işlem dışı bırakılır. Yeni alt sınır üst sınıra eşitse son adıma, değilse 4. adıma geçilir.
- **4. Adım:** Çözümü araştırılmamış yani işlem dışı olmamış alt bölüm var ise 4. adıma dönülür. Tüm düğümler işlem dışı ise 5. adıma geçilir.
- **5. Adım:** Son alt sınıra karşı gelen çözüm optimum çözümdür. Eğer, alt sınır $-\infty$ ise modelin tamsayı çözümü yoktur .

Dal-Sınır	Kesme Düzlemi
• Dal-Sınır algoritmasının altında yatan temel fikir, uygun çözüm alanı içerisindeki bütün tamsayılı çözüm çiftlerini incelemektir (Özkan, 1998). • Optimum sonucu kullanıcıya sunması açısından Kesme Düzleminden daha üstündür. • Dal-Sınır algoritmasının en büyük dezavantajı her tamsayılı çözüm çiftinin elde edildiği düğümlerde Doğrusal Programlama yapılması gerektiğidir. Büyük çaplı bir problemde çok zaman harcanarak her bir düğüm için ayrı işlem yapılır.	• Kesme Düzlemi algoritmasının altında yatan temel fikir ise, optimum doğrusal programlama çözümüne yakın olan optimum tamsayı noktasını bulmaktır. • Çözüme Dal-Sınır yöntemine kıyasla daha kısa sürede ulaşılır. • Kesme Düzlemi algoritmasının, yuvarlaklaştırma hatalarına karşı hassas oluşu bir dezavantajdır. Bazen optimum sonuca ulaşılamayabilir. • Yöntemin başarısı problemin şekline bağlıdır. Optimum çözüme ulaşmak için, bazen sadece kısıtların sırasını değiştirmek, gerekli kesme düzlemi sayısını önemli ölçüde etkileyebilmektedir



Şekil 12 Çözüm Karşılaştırması

2.3 Tamsayılı Programlamanın Kullanımı Örnekleri

Günümüzde işletme ve üretim süreçlerin karmaşıklığının artmasıyla birlikte karar verme ve yönetim süreçlerinde sezgisel yaklaşımlar karmaşık ve birbiri ile bağlantılı sistemlere çözüm getirmekte yetersiz kalmaktadır. Bu nedenle, artık işletmelerdeki üretim ve yönetim süreçlerinde nicel yöntemlerin kullanımı giderek yaygınlaşmaktadır. Söz konusu süreçlerin yönetiminde karşılaşılan problemlerde incelenen sistemin belirli kısıtlar altında optimizasyonu hedeflendiğinde doğrusal programlama yöntemleri tercih edilmektedir. Problemde bölünemeyen kaynakların olması durumunda (insan, makine, bina, vb. gibi) tamsayılı doğrusal programlama yöntemlerine başvurulmaktadır.

Dolayısıyla, rekabetin iyice kızıştığı bir ortamda maliyet baskısı altında en verimli bir şekilde üretim yapan veya hizmet sunan firmaların doğrusal programlama araçlarını kullanmasına da sıkça rastlanmaktadır.

- **Sanayideki uygulamalar**

Doğrusal programlamanın kullanıldığı en önemli alanlardan birisi imalat sanayidir. Özellikle hem fiziki hem de beşeri kaynak kullanan işletmeler, kaynak optimizasyonu ve kâr maksimizasyonu sağlamak için bu yönteme başvurmaktadırlar. İmalat sanayinde karşılaşılan problemlere bakıldığında, zamanla çözüm aranan problemlerin daha karmaşık olduğunu görülmektedir. Bunun iki nedeni bulunmaktadır. Birincisi, imalat sanayinin giderek piyasa koşullarından daha fazla etkilenmesi ve hizmet sektörüne daha sıkı entegre olmasından dolayı, sektördeki problemlerin birçok kısıda sahip olmasıdır. İkincisi ise, bilgisayarların işlem gücündeki artışa ve doğrusal programlama modellerini çözen paket programlardaki gelişmelere paralel olarak daha karmaşık modellerin kısa zamanda çözülebilmesidir. Örneğin, Şahin (1994) tarafından yapılan bir çalışmada, ele alınan işletmenin kâr maksimizasyonu hedefine ulaşabilmesi için üretim konusu ürünlerin, hangi türünden ne kadar üretilmesi gerektiği tamsayılı programlama yardımıyla tespit edilmiştir. Araştırma MAKSAN transformatör fabrikasında gerçekleştirilmiştir. Söz konusu işletme çeşitli tiplerde dağıtım, güç ve özel amaçlı transformatörler üreten bir işletme olup 1981 yılından bu yana elektromekanik alanda faaliyet göstermektedir. İşletmede sargı, nüve, kazan ve montaj olmak üzere 4 ana bölüm mevcuttur. Transformatör imalatı 4. bölümde yapılmaktadır. İşletmenin gerçeklerine uygun bir matematiksel model kurulabilmesi için göz önünde tutulan varsayımlar şu şekilde sıralanmıştır:

- Program için bir yıllık dönem esas alınmıştır. Bu süre içerisinde günde üç vardiya yapılması kısıtlayıcı faktörlerin kapasitesini belirlemiştir.
- İşletmenin tek amacı kâr maksimizasyonudur.
- Bütün ürünler sipariş üzerine üretileceğinden işletme için ürettiğini satamama gibi bir problem yoktur.
- İşgücü ve hammadde temini her an mümkündür.
- İşletmenin girdi ve çıktıları arasında doğrusal bir ilişki olduğu kabul edilmiştir.
- Kâr maksimizasyonu amaç fonksiyonunda ürünler için katsayı olarak brüt kârlar esas alınmıştır.

- Üretim esasları sıfır veya sıfırdan büyük tamsayı değerleri olacaktır. İşletmenin pazar payının büyük bir kısmını dağıtım trafoları oluşturmaktadır.

Trafoların yıllık toplam üretimin %95,3'ünü oluşturmasından dolayı, bu dağıtım trafolarının üretimi "programa alınan faaliyetler" olarak seçilmesi uygun görülmüştür. İşletmenin üretim süreci içerisinde sargı ve nüve bölümlerinin kısıtlayıcı faktör olduğu tespit edilmiştir. Kâr maksimizasyonu doğrultusunda dikkate alınan varsayım ve kısıtlar ile amaç ve kısıt fonksiyonlar yazılarak doğrusal programlama modeli oluşturulmuştur. Modelin çözümü için LINDO paket programı kullanılmıştır. Sonuç olarak, işletmenin 50 kVA transformatörlerden yılda 6867 adet, 1000 kVA transformatörlerden yılda 1299 adet üreterek bir ürün karması yapması ve bunu bağlı olarak 435.195.000.000 TL (435.195.00 TL günümüz için) kâr maksimizasyonu elde edeceği ortaya çıkmıştır. Bu sonucun optimal bir çözüm olduğu vurgulanmıştır.

Doğrusal programlama yönteminin uygulama alanı sadece imalat sanayi ile sınırlı değildir. İnşaat sektörü gibi yüksek miktarlardaki insan gücü, makine, malzeme, para ve bilginin koordinasyonunu gerektiren sektörlerde de bu yöntem kullanılmaktadır. Bahsedilen unsurların etkin bir biçimde yönetimi inşaat projelerinin zamanında bitirilmesini sağlayacağı gibi, projelerin beklenen maliyetler dâhilinde tamamlanmasını da mümkün kılacaktır. İnşaat projelerinin yönetimini kolaylaştırmak için geliştirilen yöntemler arasında CPM (Critical Path Method – Kritik Yol Yöntemi) sıklıkla tercih edilmektedir. Fakat karayolları, boru hatları ve tünel inşası gibi büyük ve birçok benzer görevin farklı mekânlarda tekrarını içeren projelerde CPM yönteminin yetersiz kaldığı ifade edilmekte ve tamsayılı doğrusal çizelgeleme yönteminin kullanımı önerilmektedir Bu yöntemin önerilmesinin üç temel nedeni bulunmaktadır:

- Kaynakların fiziksel sınırlarına bağlı kalmak
- Kaynak kullanımındaki günlük dalgalanmalardan kaçınmak
- Kaynak akışının sürekliliğini sağlamak

İnşaat projelerinin planlanmasında tamsayılı doğrusal çizelgeleme yönteminin tercih edilmesinin nedeni, kullanılan kaynakların kesirli olarak ifade edilmesinin sektörel dinamikler açısından uygun olmadığıdır. Dolayısıyla, Mattila ve Abraham (1998)'ın çalışmasında kaynak planlaması ve çizelgeleme probleminin matematiksel modelinin oluşturulmuş ve problemin tamsayılı doğrusal programlama yöntemi ile çözümü gerçekleştirilmiştir. Yukarıda da açıklandığı üzere, sınırlı kaynakların maksimum verimle kullanımı için modelin amaç fonksiyonu, tercih edilen günlük
kaynak kullanım oranlarındaki sapmaların karesinin minimize edilmesi şeklinde ifade edilmektedir. Buna göre kısıtlar da aşağıdaki şekilde tanımlanmıştır:

- Günlük kaynak kullanımı artı-eksi sapmalar dâhil tercih edilen oranda gerçekleştirilmelidir.
- Doğrusal aktiviteler için her bir gün için sabit bir gerçekleşme oranı belirlenmelidir.
- Blok olarak gerçekleştirilecek aktiviteler sadece bir pozisyon işgal etmelidir.
- Zaman-uzay diyagramında herhangi bir çakışma olmamalıdır.
- Kontrol çalışmaları değişkenlik göstermemelidir.

Amaç fonksiyonlarının ve kısıtların tanımlanmasının ardından, modelin gerçek veriler ile test edilmesi için Mattila ve Abraham (1998) tarafından Amerika Birleşik Devletleri'ndeki Kuzey Michigan eyaletinde yapılan 41 numaralı yol inşaatı projesinin verileri temel alınmıştır. Bahsedilen çalışma LINDO paket programının 5.3 versiyonu kullanılarak modellenmiştir ve çözümde dal-sınır yöntemi kullanılmıştır. Sonuçlara bakıldığında, LINDO tarafından oluşturulan çözümün CPM yöntemine kıyasla ortalama kaynak kullanımındaki sapmayı 94'ten 73'e indirdiği görülmektedir. Çözüm tarafından sunulan günlük kullanım oranlarına bakıldığında, CPM modelinde 17 birim kaynak kullanılarak gerçekleştirilen bir görevin tamsayılı doğrusal programlama yöntemi kullanılarak 13 birim kaynak ile gerçekleştirilebileceği işaret edilmektedir. Ayrıca toplam kaynak kullanımına bakıldığında da tamsayılı doğrusal programlama yönteminin toplam kaynağı, 289 günden 288 güne indirdiği görülmektedir.

Bu çalışmanın güncel literatüre katkısı değerlendirildiğinde, inşaat sektöründeki kaynak kullanımı probleminin çözümünde tamsayılı doğrusal programlama yönteminin nasıl kullanılacağına dair bir yöntem sunması ve kaynak kullanımına ilişkin matematiksel modelin nasıl kurgulanacağı ön plana çıkarmasıdır.

Doğrusal programlama yönteminin şu ana kadar bahsedilen uygulamalardan çok farklı bir sektöre uygulanmasını Weintraub (2007) tarafından yayımlanan bir çalışmada rastlanmıştır. Bahse konu çalışma, son yıllarda doğal hayatın korunması, erozyonun önlenmesi, su kalitesinin korunması ve doğa güzelliklerine zarar verilmemesi gibi ihtiyaçların giderek artan bir öneme sahip olduğu tespitiyle başlamakta ve ormanlardan kereste elde edilmesinin planlanmasının zorluk derecesi de giderek artığını vurgulamaktadır. Bu nedenle söz konusu çalışmada ormancılık sektöründe tamsayılı doğrusal programlamanın nasıl gerçekleştirilebileceğine dair bir örnek sunmuştur.

Bu çalışmada temel amaç ormanlardan kereste elde edilmesinden sağlanan gelirin maksimize edilmesidir. Bu nedenle amaç fonksiyonu kesim yapılacak alan ile birim alandan elde edilecek kârın çarpımı olarak ifade edilmiştir. Her ne kadar orman kesiminden elde edilecek gelirim maksimize edilmesi sağlanacak olsa da, kurallara göre birbirine komşu iki alanın kesiminin yapılması yasak olduğundan modele bir kısıt eklenerek bu durumun önüne geçilmesi öngörülmektedir. Ayrıca talepten fazla kesim yapılmasının engellenmesi için de her bir alandan elde edilecek birim kereste miktarı ile kesim yapılacak alanın çarpımının talebi karşılaması kısıdı eklenmiştir. Araştırmacı günümüzde ormancılık sektöründe Monte-Kârlo benzetim yönteminin ve Tabu arama metodunun kullanıldığını belirtmekte ve kereste üretiminin modellenmesinde önerilen modelin kullanımı durumunda önemli faydalar sağlanacağını belirtmektedir (Weintraub, 2007).

Ülkemizdeki lokomotif sektörlerden olan ve uzak doğu kaynaklı ucuz tekstil ürünleri ile rekabet etmeye çalışan tekstil sektörünün yatırım önceliklerinin belirlenmesi için de doğrusal programlama yöntemlerinden yararlanılmaktadır. Doğanlı (2006), yapmış olduğu bir çalışma ile tamsayılı programlama yöntemlerinin Türkiye'de tekstil sektörüne uygulanmasıyla elde edilecek faydaları tespit etmeyi amaçlamıştır. Uygulama

için Denizli MEBATEKS Tekstil Sanayi ve Ticaret A.Ş. fabrikası seçilmiştir. Bu fabrika bornoz ve havlu üretimi yapmakta ve ihracatını gerçekleştirmektedir. İşletme, büyüme ve teknolojiyi takip etme ve bu sayede pazar payını artırmayı planlamaktadır. İşletme iki dönem için 7 projeden (1-Klima sisteminin oluşturulması, 2-Kazan dairesinin yenilenmesi, 3-Boyahane binası yapılması, 4-Depo için yeni bina yapılması, 5-Yurt dışı temsilciliklerinin oluşturulması, 6-Dokuma tezgâhlarının yenilenmesi, 7-Ara mamuller için gerekli olan makinelerin [düğme, nakış, etiket vb.] alınması) en uygun olanları faaliyete geçirmeyi düşünmektedir. İşletmenin yatırımları için tek kısıtı her iki dönem içinde sermaye harcamalarının miktarlarıdır. İşletme 1. dönem için 500.000 YTL, 2. dönem için ise 200.000 YTL ayırabilmektedir. İşletmenin elindeki parayı en optimal şekilde kullanabileceği projeler tamsayılı programlama tekniği ile belirlenmiştir. Problemin bilgisayar ortamında çözümünde LINDO paket programı kullanılmıştır.

Uygulama sonucunda işletmenin 7 projeden 5'ini kabul etmesi, 2'sini reddetmesi gerektiği ortaya çıkmıştır. Diğer bir ifade ile işletme klima sistemi oluşturacak, kazan dairesini yenileyecek, depo için yeni binalar inşa edecek, dokuma tezgâhlarını yenileyecek, ara mamuller alacak, fakat boyahane için yeni bina inşası ve yurt dışı temsilcilikleri oluşturma projelerini reddedecektir. Böylece tamsayılı programlama ile en optimal sonuca ulaşılabileceği gerçekleştirilen bu uygulama ile gösterilmiştir.

Bu bölümde anlatılan çalışmalara bakıldığında imalat sanayi, inşaat, petro-kimya, ormancılık ve tekstil gibi birbirinden oldukça farklı sektörlerdeki problemlere çözüm getirmede doğrusal programlama yöntemlerinden yararlanıldığı görülmektedir. Dolayısıyla, bu tez çalışmasına konu teşkil eden vana ceketleri üretimi yapan bir firmanın probleminin çözümünde yukarıda bahsedilen çalışmalarda kullanılan yaklaşımların ve yöntemlerin yön gösterici olabileceği düşünülmektedir.

- **Hizmet sektörü**

Hizmet sektörü sahip olduğu dinamikler nedeniyle sanayiye göre önemli farklılıklar içermektedir. Sektörde iş gücü kullanımının büyük bir kısmının insan faktörüne bağlı olmasından ötürü oluşan problemlerin niteliği sanayi kuruluşları ile kıyaslandığında farklı yaklaşımlar gerektirmektedir. Bu sektördeki uygulamalar

incelendiğinde güvenlik, finans, sağlık, eğitim ve ulaştırma gibi alanlarda çok farklı uygulamalarla karşılaşılmaktadır.

Yapılan bir çalışmada, ülkemizde nüfusun üçte birinin yaşadığı altı büyük ilde (Adana, Ankara, Erzurum, İstanbul, İzmir ve Samsun) minimum maliyetli optimum beslenme problemi doğrusal programlama tekniği kullanılarak araştırılmıştır (Alparslan, 1996). Çalışmada et ve et ürünleri, balık ve deniz ürünleri, kümes hayvanları, sakatatlar, süt ve süt ürünleri, yumurta, kuru baklagil ve kuru yemişler, koyu sarı ve yeşil sebzeler, patates, turunçgil ve meyveler, tahıl ve ürünleri, ekmek ve türevleri, diğer fırın ürünleri, hayvansal ve bitkisel yağlar, şeker ve tatlılar olmak üzere 129 çeşit besin maddesi dikkate alınmıştır. Besinlerin yanı sıra illerde yaşayan nüfus da aile tipi ve yaş gruplarına göre sınıflandırılmıştır. İncelenen aile tipleri yoksul, ekonomik, orta ve zengin şeklinde ele alınmıştır. Yaş grupları ise 4-6, 7-10, 15-18, 19-22, 23-50 (erkek), 50+ (erkek), 23-50 (kadın), 50+ (kadın) şeklinde belirlenmiştir. Sağlıklı beslenme için yenebilen gıdalardan alınması gereken 13 enerji ve beslenme elemanı gerektiği düşünülerek, söz konusu besin maddelerinin analizleri de yapılmıştır. Yiyecek maddelerinin fiyatları için DİE kayıtları ve pazar araştırması sonuçları kullanılmıştır. Her il için ayrı ayrı amaç fonksiyonu yazma yerine amaç fonksiyondaki değişkenlerin katsayıları illerin alfabetik sıralamasına göre yazılarak toplu halde gösterilmiştir. Fiyatlar 100 gram cinsinden tespit edilmiştir. Araştırmada yaş gruplarına göre günlük alınması gereken minimum besin elemanları, dört tip aile için yaş gruplarına göre günlük alınması gereken yiyecek maddeleri, her bir aile tipi için yaş gruplarına göre günlük alternatif beslenme maliyetleri verilmek suretiyle bütçe kısıtları oluşturulmuştur.

Problemlerin çözüm aşamasında LINDO programı kullanılırken, amaç fonksiyon değerleri her il için ayrı ayrı verilmiştir. Başlangıçta 129x61 boyutlu Simpleks tablo çözümü elde edilmiştir. Duyarlılık analizi yapılmış ve değişkenlerin gölge fiyatları da alınmıştır. Daha sonrasında bütçe kısıdı da eklenerek çözüm gerçekleştirilmiştir. Sonuç olarak, 1994-1997 yılları arasında altı ilde dört tip ailedeki farklı yaş gruplarının her biri için ayrı ayrı minimum maliyetli optimum beslenmenin nasıl olacağı ortaya konmuştur. Ayrıca araştırmacı, modelin sadece çalışmanın yapılacağı yıllar ile sınırlı kalmayacağını, gerektiğinde modelde kullanılan maliyetlerin güncel maliyetlerle değiştirilerek modelin her zaman kullanılabileceğini belirtmektedir

Hizmet sektöründe doğrusal programlama yöntemlerinin kullanıldığı bir diğer alan ise eğitimdir. Örneğin, 1998'de Ohio Üniversitesi Ticaret Koleji'nde, öğretim üyelerini ve derslerin, sınıflara ve belirlenen ders saatlerine çizelgelenmesi amacıyla tamsayılı programlama kullanılmıştır.O zamana kadar COB (Colleauge of Business) her akademik dönem için çizelgelemeyi elle yapmaktadır. COB' un her dönem 65 - 75 öğretim üyesi, 110-130 dersi, 16 sınıfı bulunmaktadır. COB derslerin verileceği zaman aralıklarını önceden belirlemiştir. Dolayısıyla dersler belirlenen saatlere atanacaktır.

Mevcut süreçte dekanlık, bina içinde her bölüm için gerekli özel sınıfları ve iki veya daha fazla bölümün ortak kullanabileceği sınıfları belirtmektedir. Daha sonra bölüm başkanları önce özel sınıflara ilgili dersler atanır ardından hangi dersin verileceğini, derslere atanacak hocaları eldeki sınıf durumunu ve öğretim üyelerinin tercihlerini değerlendirerek bölüm derslerini, kalan sınıflara atamaktadır. Bölüm başkanları atama yaptıktan sonra boş sınıf ve zaman kalırsa diğer bölümleri bilgilendirmektedir. Eğer bina içi derslik sayısı yetersiz kalırsa diğer okulların derslikleri kullanılmaktadır.

Bu yaklaşım uygulanırken binadaki sınıflar optimum kullanılmamaktadır. Derslik yetersizliğinde, fakülte üyeleri dersleri bina dışındaki dersliklerde vermektedirler. Ancak buradaki derslikler, verilecek dersler için uygun sevide değildir. Öğretim üyelerinin sayısı arttıkça sınıf atama için yeni bir yaklaşım zorunlu hale gelmiştir. Dolayısıyla, öğretim üyelerin tercihleri çerçevesinde, öğretim üyesi, ders, sınıf, zaman aralığı atamasının en iyi bir şekilde yapılması için bir model oluşturulması planlanmıştır. Modeldeki amaç fonksiyonu sınıfların optimum kullanımı ve öğretim görevlilerinin tercihlerinin azami oranda karşılanması dikkate alınarak oluşturulmuştur. Ayrıca, modeldeki aşağıda bahsedilen kısıtları içermektedir:

- Bir öğretim görevlisi, gerekli olduğu kadar derse atanmalı ve bir ders saati içerisinde birden fazla derse atanmamalı,
- Bir sınıfta aynı anda birden fazla ders verilmemeli,
- Mümkün olduğunca çok ders, dersliklere ve zaman aralığına atanmalıdır.

Üyelerin ve yönetimin istediği tarzda bir çizelgeleme hazırlamak için aşağıdaki başlıklara dikkat edilmelidir.

- Maksimum Ders Günleri: Bazı öğretim üyeleri ders vereceği günlerin sadece haftanın belirli iki günü olmasını isteyebilir. Bu kısıt öğretim üyesinin ders vermek istediği gün sayısı kadar atama yapmaya zorlar. Eğer böyle bir kısıt olmazsa bir öğretim üyesine haftanın dört günü ders atanabilir.

- Ardışık Gelen Ders Saatleri: Bu kısıtla; modelin, öğretim üyelerine aralıksız ders ataması engellenmektedir.

- Zaman Aralık Grupları: Örneğin bir öğretim üyesi sadece sabahları veya öğleden sonraları ya da sadece Pazartesi, Çarşamba veya Salı, Perşembe günü ders vermek isteyebilir.

- Her Derse Çok Öğretim Görevlisi: Yeni eğitim sistemlerinde, dersler farklı disiplinler altında ve birkaç eğitmenle işlenmektedir. Örneğin Ohio Üniversitesi'nin eğitim programında dört eğitim görevlisinin verdiği iki küme dersi* bulunmaktadır. Bu dersler haftada 16 saat olarak planlanmıştır. Manuel atamada; küme dersi veren eğitmenlerin başka derslere de atanmış olmasından dolayı genelde aynı saate ancak bir eğitmen atanabilmektedir.

- Günlük Çizelgelenen Minimum Ders Sayısı: Bu kısıt; sınıf kullanımının haftanın günlerine eşit olarak dağılımını gerçekleştiren bir çizelge hazırlanmasını sağlamaktadır.

- Bölüm Seviyelendirme: Bir kolej bütün bölümleri için çizelgeleme yaptığında ve kendi derslerini atayacağı yeteri kadar dersliği kalmadığında, bölümlerin derslerinin oransız bir şekilde atanmasından kaçınmalıdır.

- Atama Öncesi: Literatürde, araştırmacılar atama öncesi terimini kullanmaktadırlar. Otomatik bir çözüm öncesi öncül bir atama yapılabilmektedir. COB atama öncesini küme dersleri için kullanmaktadır.

Yukarıda tanımlanan bu model Pentium tipi 1600 MHz bir bilgisayarla iki ile beş dakika arasında çözülmüştür. Problemin çözülmesinde CPLEX paket programı kullanılmıştır. Modelin ne kadar büyük ve karmaşık olduğu aşağıdaki parametrelerden de anlaşılabilmektedir:

- 0-1 Değişkenleri: 1900-2400
- Sürekli Değişkenler: 10-150
- Kısıtlar: 1600-1900

Modelin COB'de uygulanmasının ardından bina içindeki sınıf kullanımının arttığını belirtilmektedirler. Ayrıca öğretim üyelerinin sayısının artmasına rağmen COB binası dışında fakülte üyeleri daha az sayıda ders vermeye başlamışlardır. Çizelgeleme süresi %50 oranında azalmıştır. Bölüm başkanları ders dağılımında hocaların tepkisinden kurtulmuş ve öğretim üyelerini atamanın kişisel olmadığını konusunda ikna

edebilmişlerdir. Mesela, bir dönem az ders atanan öğretim üyelerine diğer dönem öncelik verilmiş ve fazla ders atanmıştır.

Doğrusal programlana yöntemi bilindik uygulamalar dışında yeni alanlara da uygulanabilmektedir. Örneğin, bilim adamları protein sentezi sırasında da doğrusal programlama yöntemleri kullanarak kararlı protein zincirlerinin tasarımını gerçekleştirebilmektedir. Kingsford ve diğerleri (2004) protein tasarımı sırasında karşılaşılan ve karmaşık ve büyük boyutlu bir problem olan zincir-yanına konumlama (SCP: side-chain positioning) hesaplama yöntemine hızlı ve etkin bir çözüm getirmek için tamsayılı doğrusal programlama modelinden yararlanmışlardır.

SCP yöntemi, sabit bir molekül omurgası üzerine, dönerek yerleşen molekül (aminoasit) zincirlerinin tüm sistemin enerjisini minimize edilecek şekilde yerleştirilmesini öngörmektedir. Bu bağlamda, amaç fonksiyonu molekül omurgası ve moleküller arasındaki enerjinin minimize edilmesine yönelik olarak oluşturulmuştur. Problemin kısıtları da her bir boşluğa bir molekül bağlanmasını sağlamaktadır. Geliştirilen modelin denenmesi aşamasında, seçilen 25 protein üzerinde çalışmalar yapılmış ve model CPLEX paket yazılımı kullanılarak çözülmüştür. Bu çalışmanın çıktısın en büyük etkisinin SCP yönteminin kullanımında doğrusal programlamanın kullanımının mümkün olduğunu kanıtlaması olduğu belirtilmiştir.

Taşımacılık faaliyetlerinde doğrusal programlama ile işlerin verimliliğinin artırılabilmektedir. Söz gelişi ülkemizde Gül ve Elevli (2006) tarafından yapılan bir çalışmada, bir çimento fabrikasının torba çimento nakliye işinde bir dış kaynaklı firmadan yararlanılması durumunda elde edilecek olan maliyet avantajını belirlenmesi çalışılmıştır. Bu amaçla, dış kaynak firmasının sahip olması gereken kamyon filosunun büyüklüğünün tespit edilmesinde tamsayılı doğrusal programlamadan faydalanmışlardır. Çalışmada çimento fabrikasının 8 bayisinin 2004 yılı Mayıs ayına ait aylık talepleri, bayilerin çimento fabrikasına olan uzaklıkları, farklı kapasitedeki üç adet kamyonun birim maliyetleri dikkate alınmıştır. Kurulacak tamsayılı programlama modeli ile fabrikadan 8 farklı bayi deposuna bayiinin ihtiyacını karşılayacak çimento nakliyesinin kaç seferde hangi tip kamyonla yapılması gerektiğinin saptanması amaçlanmıştır.

Araştırıcılar ortaya koydukları tamsayılı doğrusal programlama modelini bilgisayar ortamında çözmek için TORA ve WinQSB bilgisayar programlarından yararlanmışlardır. Sonuç olarak, söz konusu bu çalışmadan çimento fabrikasının nakliye işlerini dış kaynaklı bir firma ile gerçekleştirmesinin nakliye maliyetlerinden %35 oranında tasarruf elde edileceği ortaya çıkmıştır.

Taşımacılık sektöründe doğrusal programlama yönteminin kullanıldığı diğer bir uygulama da Ergülen ve Kazan (2007) tarafından gerçekleştirilen çalışmadır. Bahsedilen

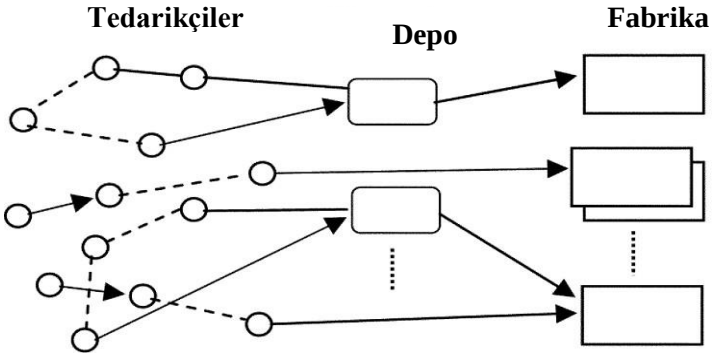
çalışmada dağıtım ağına sahip bir firmanın taşımacılık maliyetlerinin minimize edilmesi için bulanık mantık ve tamsayılı doğrusal programlama yöntemlerinin uygulanması planlanmaktadır. Bahse konu firmanın 3 farklı tipteki 25 araçlık araç filosu ile 11 farklı ile dağıtım yaptığı belirtilmiştir. Bu bilgiler ışığında model, amaç fonksiyonu ve kısıtlar şu şekilde oluşturulmuştur.

Amaç fonksiyonu araçların farklı bölgelere yapacağı seferlerden doğan taşıma maliyetleri ile araç kiralama maliyetlerini minimize etmeye çalışmaktadır. Bu fonksiyonu çözebilmek için sistemde iki farklı kısıt tanımlanmıştır. İlk kısıtta her bir aracın toplam sefer süresinin araçların en fazla süre ile seferde bulunabilecekleri toplam sefer süresinden az olması gerektiğin tanımlanmıştır. İkinci kısıtta ise, taşınan yük miktarının merkezin talep ettiği yük miktarını karşılaması sağlanmıştır. Bahsedilen problem yukarıdaki model temel alınarak bulanık mantık kullanımıyla ve tamsayılı doğrusal programlama yöntemiyle hesaplanmıştır. Hesaplama sonucunda elde edilen optimum plan firmanın mevcut dağıtım planı ile kıyaslandığında firmanın toplam dağıtım maliyetinde %11,51 oranında düşüş olduğu gözlemlenmiştir. Ayrıca optimum planda firmanın dağıtım araçlarını yaptığı toplam sefer sayısı da 168'den 117'ye indirilmiş ve sefer sayılarında da %30 oranında azalma sağlanmıştır.

Ergülen ve diğerleri (2005) de, taşımacılık sektörüne yönelik çalışmışlar ve tamsayılı doğrusal programlama modelini firma yöneticilerinin kendi işletmelerine kolaylıkla tatbik edecekleri şekilde basamaklar halinde açıklayarak genel kapsamlı tamsayılı doğrusal programlama modelini gıda sektöründe faaliyet gösteren ve 24 adet distribütörü olan bir firmaya uygulamışlardır. Bahse konu çalışmada kurulan model sefer maliyetlerinin ve araç kiralama giderlerinin minimizasyonuna dayanmaktadır. Genellikle bu problem firmanın daha önceki tecrübeleri ışığında çözümlenmeye çalışılmıştır fakat bu çalışmada adı geçen probleme bilimsel bir çözüm getirilmesi hedeflenmiştir.

Modeldeki amaç fonksiyonu araçların sefer maliyetleri ve sefer sayıları ile araçların kiralama maliyetleri dikkate alınarak oluşturulmuştur. Modelin kısıtlarına bakıldığında ise, iki temel kısıtın tanımlandığı görülmüştür. İlk kısıt araçların birim zamanda yapacakları sefer sayısı dikkate alınarak oluşturulmuş, ikinci kısıt ise dağıtımı yapılacak malların araçların tonaj değerleri ve dağıtım bölgelerindeki taleplere göre belirlenmiştir. Bu doğrultuda oluşturulan modelin çözümlenmesinde WINQSB paket programı kullanılmıştır. Elde edilen sonuçlara göre modelde oluşturulan optimum çözüm planına ait dağıtım maliyetlerine ulaşılmıştır. Firmanın dağıtım maliyeti tespit edilirken 24 distribütörün siparişlerine göre 13 tonluk klimalı araçlarla yapılan sefer sayılarına göre de firmanın dağıtım planına ait dağıtım maliyeti çıkarılmıştır. Sonuç olarak, tamsayılı doğrusal programlama modeli ile yıllık seviyede karşılaştırıldığında, dağıtımın yaklaşık %6 tasarruf sağladığı tespit edilmiştir.

Taşımacılık sektörü ile ilgili Onoyama ve diğerleri (2008) tarafından yapılan Japonya'daki bir uygulama da birden fazla işletmeye girdi sağlayan tedarikçilerin ham madde ve ara mamullerinin taşıma maliyetlerinin azaltılması amacını taşımaktadır. Şekil 14'den de görülebileceği üzere, fabrikaların ihtiyaç duyduğu mallar hem depolardan hem de doğrudan tedarikçilerden elde edilebilmektedir. Böyle bir durumda her iki farklı kaynaktan elde edilen malların taşıma maliyetlerinin minimize edilmesi bu çalışmanın temel amacını oluşturmaktadır.



Şekil 13 Tedarik Ağının Gösterimi

Yukarıda bahsedilen çalışmanın amaç fonksiyonu tedarikçilerden depolara ve fabrikalara giden malların taşıma maliyeti ile depolanan malların depolama maliyetlerinin minimize edilmesini içermektedir. Kısıtların içerikleri de aşağıda özetlenmektedir:

- Her bir tedarikçinin depolara ve fabrikalara göndereceği mallar tedarikçinin mal sağlama kapasitesini aşmamalıdır.
- Depolardan ve tedarikçilerden sağlanan mallar fabrikanın ihtiyaçlarını karşılamalıdır.
- Her bir parçadan sağlanan miktar deponun topla kapasitesini aşmamalıdır.
- Depodan sağlanan her bir parçanın toplam miktarı toplam taşıma kapasitesini aşmamalıdır.
- Tedarikçilerden doğrudan fabrikaya aktarılan mallar araçların alt ve üst kargo kapasitesi dâhilinde olmalıdır.

Tedarikçilerden fabrikaya mal aktarımı olması durumunda son kısıt modele dâhil edileceğinden dolayı, söz konusu kısıtın 0-1 değişkeni kullanılarak ifade edilmesi uygun bulunmuştur. Bu nedenle, bahsedilen problemin karma tamsayılı doğrusal programlama yöntemi ile çözülmesine karar verilmiştir. Bunun ardından modelin 4 tedarikçi, 2 depo, 2 fabrika, 5 farklı tedarikçi fabrika rotasından oluştuğu farz edilerek, çözüm aşamasına geçilmiştir. Modelin sağladığı çözümün daha önce yapılan yaklaşık hesaplamalara göre %12 oranında maliyet tasarrufu sağladığı görülmektedir.

Doğrusal programlama yöntemleri ile işletmelerdeki iş gücü planlaması da yapılmaktadır. Örneğin, Çevik (2006) tarafından yayınlanan bir çalışmada, Tokat il merkezinde faaliyet gösteren bir işletmede minimum maliyeti sağlayacak işgücü planlaması tamsayılı doğrusal programlama yardımıyla yapılmıştır. Lokanta olan işletme aynı zamanda müşterilerine sunduğu tatlıların imalatını da kendisi üstlenmektedir. İşletme 35 çalışanı olan bir aile işletmesi durumundadır. Söz konusu işletmede mesaiye pek fazla özen gösterilmemesi bir takım problemleri beraberinde getirmekte ve ayrıca iyi bir işgücü planlamasının olmamasından dolayı mevcut personel bazen yetersiz bazen de atıl durumda kalmaktadır. Bahsedilen bu sorunları gidermek için işlemeye minimum maliyeti sağlayacak ve bununla birlikte hizmeti aksatmayacak bir personel düzenlemesi gerekmektedir. Araştırıcı işletme yöneticisinden mevcut çalışma saatlerini, bu saatler arasında istenilen vardiya şeklini ve özelliklerini, çalışanların işletmedeki görevlerini ve her bir personelin ayrı ayrı aylık ve günlük ücretlerini elde ederek model geliştirmiştir. Model için en uygun vardiya sisteminin tam zamanlı çalışan komi, garson ve bulaşık yıkama elemanları için 06-14, 12-18, 14-22, yemek ustaları için 06-14, 10-18 ve 14-22 şeklinde olmasını uygun görmüştür. Yarı zamanlı çalışan personel için ise 10-14, 14-18 ve 18-22 şeklinde olmasını planlamıştır. Verilere göre oluşturulan tamsayılı doğrusal programlama modeli bilgisayarda WINQSB 1.00 paket programı ile çözümlenmiştir.

Elde edilen bulgulardan, işletme personel maliyetini minimuma indirmek ve işleri aksatmamak için 10-14 ve 18-22 vardiyalarında yarı zamanlı 2'er eleman, 06-14 vardiyasında 5 komi, 5 garson, 1 yemek ustası, 2 bulaşık yıkama elemanı, 1 baklava ustası, 9 baklava elemanının çalıştırılması gerektiği anlaşılmıştır. 12-20 vardiyasında 7 komi, 5 garson, 2 bulaşık yıkama elemanı, 10-18 vardiyasında ise 1 yemek ustası çalıştırılmalıdır. 14-22 vardiyasında ise 2 komi, 4 garson, 1 yemek ustası ve 2 bulaşıkçı çalıştırılmalıdır. Bu tablo içerisinde işletmenin günlük minimum maliyetinin 726,72 YTL olacağı hesaplanmıştır. Böylece Tokat il merkezinde bulunan işletmenin personel giderlerin minimize edilerek, personelinin daha verimli bir şekilde kullanması sağlanmıştır (Çevik, 2006).

2.4 Tamsayılı Programlama Yazılımları

- **Baron (Branch-And-Reduce Optimization Navigator)**

BARON, global optimalite için konveks olmayan optimizasyon problemlerinin çözümünde kullanılan bir hesaplama sistemidir. Saf sürekli, saf tamsayılı, karma tamsayılı doğrusal olmayan problemler bu yazılımla çözülebilmektedir.

BARON genel varsayımlar altında global optimaliteyi garanti etmektedir. Ancak gonyometrik fonksiyonları içeren veya dış bir fonksiyonla ilintili kısıtları olan problemleri çözememektedir. BARON matematiksel program içerisindeki bütün doğrusal olmayan değişkenler ve ifadelerin alt ve üst limitlerle sınırlandırılmış olmasına gerek duymaktadır. Bu şekilde bütün değişkenler sınırlandırıldığında 2000 değişken ve kısıtlı modelleri çözebilmektedir (BARON Solver Information, 2007).

- **Conopt**

CONOPT; ARKI danışmanlık firması tarafından geliştirilmiş, AIMMS sistemi içerisinde yer alan, büyük ölçekli doğrusal olmayan problemleri etkili şekilde çözen bir programdır. AIMMS, CONOPT tarafından, doğrusal olmayan programlama model tiplerini daha etkili çözmek için kullanılan ikinci dereceden türevleri de çözebilmektedir. CONOPT için en iyi alternatif SNOPT'dur ve bu iki yazılım birbirini tamamlayan yazılımlardır. Eğer her iki yazılım da problem çözmede başarısız olursa, model ya çok zordur ya da hatalı modellenmiştir. Ayrıca CONOPT 3 serisi QCP yeteneği de sunmaktadır. CONOPT, yeni ilaveleri ile beraber GRG metoduna dayalı bir uygun yol çözücüsüdür. CONOPT, birçok model tipi için etkili ve güvenilir bir şekilde tasarlanmıştır. GRG metodu büyük dereceli doğrusal olmayan modellerin güvenli ve hızlı bir şekilde çözülmesine yardımcı olmaktadır ve CONOPT, uygunluğun zor sağlandığı modeller ve büyük dereceli doğrusal olmayan modeller için tercih edilmektedir. CONOPT'u; çok metotlu yapısı ile birleşen en uygun metodu seçen mantık yapısı diğer doğrusal olmayan programlama çözücülerinden üstün hale getirmiştir.

CONOPT' un bütün bileşenleri büyük ölçekli modeller için dizayn edilmiştir. 10.00 kısıtın üzerindeki modeller bu program yardımı ile çözülebilmektedir. Hatta bir milyon kısıtın üzerindeki bazı özelleşmiş modeller dahi CONOPT yardımı ile çözülebilmektedir. Bu sınırlayıcı faktörleri tanımlamak zordur. Bu faktörler; kısıt veya değişken sayısının süper temel değişkenlerle kombinasyonu, optimum nokta çevresi bağımsızlığının derecesinin bir ölçüsüdür. 500 süper değişkenli bir model yavaş çözülebilmektedir. CONOPT 3 serisi yeni 2. dereceden türev tabanlı metotları ile bu modelleri daha basit hale getirmektedir (CONOPT Solver Information, 2007).

- **Snopt (Sparse Nonlineer Optimization)**

SNOPT; Philip Gill, Walter Murray ve Michael Saunders tarafından geliştirilmiş doğrusal olmayan programlama çözücüsüdür. SNOPT özellikle (konkav olmak şartıyla) fonksiyonları ve eğimleri hesaplaması zor olan doğrusal olmayan problemlerin çözümünde etkilidir (SNOPT Solver Information, 2007).

- **AOA (AIMMS Outer Approximation)**

AOA, karma tamsayılı doğrusal olmayan programlama problemlerini, bir dışsal tahmin kullanarak çözmektedir. Dışsal tahmin, doğrusal olmayan tamsayılı programlama problemlerini çözmede en bilinen yaklaşımdır. Algoritmanın temelini, karma tamsayılı programlama problemlerinin çözücüsü ile doğrusal olmayan problemlerin çözücüsü arası etkileşim oluşturmaktadır.

AOA dışsal tahmin modülünü kullanmaktadır. Bu modül AIMMS'de yazılmış standart bir dışsal tahmin algoritmasını içermektedir. Kullanıcı, aşağıdaki amaçları gerçekleştirmek için kişisel algoritmik basamakları özelleştirebilmektedir.

  - Daha iyi performans elde edebilmek
  - Daha iyi bir sonuç elde edebilmek
  - Algoritma tarafından bulunan bir çok tamsayılı sonucu depolayabilmek
  - Uygun sonuç elde etme ihtimalini artırmak için algoritma süresince değişik çözücüleri kullanabilmek
  - Bir problemin alt modellerini değiştirmek (AOA Solver Information, 2007).

- **Xpress**

Xpress, Dash optimizasyon tarafından sunulan, bir matematiksel modelleme/optimizasyon yazılımıdır. Bir tamsayılı programlama Ayrıca XPRESS, karma tamsayılı ve karma tamsayılı kuadratik programlama problemlerinin çözümünde de kullanılabilmektedir.

Xpress, üç optimizasyon algoritmasının kullanımına izin verir.

  - LP problemleri çözen Primal ve dual metotları içeren simpleks,
  - Karma tamsayılı ve karma tamsayılı kuadratik programlama problemlerini çözmek için dal sınır algoritması ve doğrusal ve kuadratik programlama için iç nokta metodu **(**interior point method**)** (XPRESS Solver Information, 2007)

Xpress-Mosel, Xpress-Optimizer ve Xpress kütüphaneleri olmak üzere üç temel bileşeni vardır:

Mosel; model oluşturma ortamı, Optimizer ise oluşturulan doğrusal programlama (DP), karışık tamsayılı programlama (KTP) modellerini, yapısındaki güçlü optimizasyon algoritmaları ile çözen modüldür. Xpress-MP Kütüphaneleri ise

daha özel uygulamalarda, Mosel ve Optimizer bileşenlerine C/C++, Java ve Visual Basic uygulamalarından giriş sağlanması için kullanılabilmektedir.

- **Xpress mosel**

Mosel, Xpress-MP nin temel bileşenlerinden biridir. ODBC kütüphane modülüyle Mosel, geniş bir veritabanı ailesiyle veri alış verişi yapabilmektedir. Genel olarak Mosel, bir text dosyasından veya veritabanından veriyi alır, Optimizer'da en uygun çözümü bulur ve sonuçları tekrar ODBC tabanlı bir ürüne gönderir.

- **Xpress MP**

Xpress-MP nin çekirdeğini, doğrusal programlama (DP), kuadratik programlama (KP) ve karışık tamsayılı programlama (KTP) problemlerinin çözüm metotları konusunda yıllarca yapılmış araştırmaların ürünü olan Xpress-Optimizer oluşturmaktadır.

Gerek Dash Optimizasyon uzmanlarıyla ve gerek dünya genelindeki araştırma gruplarıyla olan yakın ilişkileriyle, Optimizer sürekli bir gelişim içersindedir. Xpress-Optimizer kendini geniş bir problem sahasında ispatlamış bir üründür.

Çoğu problemin çözümü için en uygun stratejiler tanıtılmış olmasına rağmen çok yönlü kontrolörleri vasıtasıyla Optimizer'ın performansına müdahalelerde bulunulabilir. Özellikle karışık tamsayılı programlama problemlerinin çözümünde her zaman optimum sonuca ulaşmak tercih edilir bir şey olmayabilir. Böyle durumlarda ihtiyaca göre problemi belirli bir seviyeye kadar çözmek zaman ve maliyet açısından daha uygundur. Optimizer'ın yapısında bu esneklik tanımlanmıştır.

- **Xpress-MP kütüphaneleri**

Daha özel uygulamalarda, Mosel ve Optimizer bileşenlerine C/C++, Java ve Visual Basic uygulamalarından giriş sağlanması için Xpress-MP Kütüphaneleri kullanılabilmektedir. Xpress-MP Kütüphanelerinin sağladığı asıl avantaj, kendi uygulamalarınıza Xpress-MP'nin fonksiyonelliğinin katılabilmesidir (Xpress-MP elease 13, 2007).

- **Knitro**

KNITRO; Ziena Optimizasyon firması tarafından 2001 yılında geliştirilmiştir. Tamamlayıcı kısıtlar, eşitlik ve eşitsizlik kısıtları (konveks ve konkav) ve sınır kısıtları içeren doğrusal olmayan programlama problemlerinin çözücüsüdür. Doğrusal olmayan programlama problemlerinde, çok farklı durumlar ortaya çıkmaktadır. KNITRO geniş çaplı doğrusal olmayan programlama uygulamalarına hitap edebilmek için üç değişik algoritma sunmaktadır (KNITRO Solver Information, 2007).

KNITRO; opsiyonel özelliği ile büyük bir esnekliğe sahiptir; ayrıca C, C++, Fortran, Java, AMPL, AIMMS, Mathematica, Microsoft Excel, GAMS, LabVIEW ve MATLAB gibi programlama dilleri ve programlama ara yüzleri ile çalışabilir (KNITRO, 2007).

- **Gams (The General Algebraic Modeling System)**

GAMS; özellikle doğrusal, doğrusal olmayan ve karma tamsayılı optimizasyon problemlerinin çözmek için tasarlanmıştır. Sistem özellikle tam doğru bir model kurmak için birçok değişiklik gerektiren kompleks ve büyük çaplı problemlerin çözümünde kullanışlıdır. Kullanıcı bu sistemle; formülasyonu, çözücüyü, hatta doğrusal durumu doğrusal olmayan duruma kolaylıkla çevirebilmektedir.

GAMS; duyarlılık analizini kolaylaştırmaktadır. Kullanıcı; farklı değerler için problemi kolaylıkla çözebilmekte ve her durum için üretilen sonucu listeleyebilmektedir. Model eş zamanlı olarak geliştirilebilmekte ve çıktı alınabilmektedir. Çünkü GAMS, kullanıcıya eşitliklerin ve sembollerin açıklandığı bir metin sunmaktadır (The GAMS System, 2007).

- **Path**

Karma tamamlayıcı programlama (Mixed Complementarity Programming - MCP) problemlerinin çözümünde kullanılan Newton tabanlı bir çözücüdür. PATH genel olarak ekonomistler tarafından genel denge problemlerinin çözümünde kullanılmaktadır (PATH Solver Information, 2007).

- **Minos**

Stanford Üniversitesi tarafından geliştirilen doğrusal olmayan programlama problemlerinin çözümünde kullanılan bir çözücüdür. Özellikle doğrusal olmayan amaç fonksiyonlu ve seyrek (sparse) doğrusal kısıtlı problemlerin (kuadratik programlama gibi) çözümünde etkilidir. Ayrıca MINOS, konkav olmak şartıyla büyük sayıda doğrusal olmayan kısıtı işleme tabi tutabilmektedir. CONOPT ve SNOPT, MINOS a göre daha etkilidir ancak amaç fonksiyonu ve eğimlerini hesaplamak daha kolaysa MINOS daha hızlı ve verimlidir (MINOS Solver Information, 2007).

- **Tora**

TORA Programı Operations Research (Yöneylem Araştırması) isimli kitabın yazarı olan Hamdy Taha (2000) tarafından geliştirilen bir optimizasyon yazılımıdır. Oldukça basit ve kullanışlı bir arayüze sahip olan bu yazılım doğrusal programlama, tamsayılı doğrusal programlama, kuyruk modelleri, ağ modelleri, stok modelleri, ve taşımacılık problemleri gibi alanlardaki modelleri çözebilme yeteneğine sahiptir.

- **WinQSB**

WinQSB paket programı birçok yöneylem araştırması problemini çözebilcek araçlara sahip kapsamlı bir paket programdır. Bu paket program, doğrusal ve tamsayılı programlama yöntemine ek olarak, doğrusal olmayan programlama, dinamik programlama, hedef programlama, ağ modelleri, iş çizelgelemesi, karar analizi gibi yöneylem araştırması alanındaki birçok probleme çözüm getiren araçlar içermektedir. WinQSB, tamsayılı doğrusal programlamada dal-sınır yöntemini en iyi uygulayan programlardan biridir ve karmaşık problemlerin çözümüne kısa sürede ulaşmaktadır. Ayrıca WinQSB Microsoft Excel benzeri bir arayüz ile veri girişi imkanı sağlayan oldukça kullanışlı bir arayüze sahiptir.

2.5 Tamsayılı Programlama Yazılımlarının Bugünü ve Geleceği

Tamsayılı doğrusal programlama problemlerinin çözümü, hızlı gelişen bir alandır. Güvenli ve hızlı optimizasyon yazılım sistemleri, finanstan sağlığa, savunmadan telekomünikasyona birçok organizasyonun, stratejik, taktik ve operasyonel kararlarının modellenmesi ve optimizasyonunda etkili bir rol oynamaktadır.

Günümüzde tamsayılı programlamanın yayılması hızla devam etmektedir. Örneğin fiber-optik telekomünikasyon ağı tasarımı, ışın tedavisi, genetik, para/mal değişimi gibi alanlar tamsayılı programlamanın yeni kullanım alanlarıdır. Tamsayılı programlamanın bu gibi farklı problem tiplerinde de kullanılması, geniş çaplı problemleri tanımlama ve bu problemleri çözme şeklinde yeni teknolojilerin gelişmesine neden olmuştur.

Kısıtların otomatik olarak sınıflandırılması ve bu sınıflandırmanın süreç öncesi etkin kullanımı, kesme jenerasyonları, genel amaçlı tamsayılı doğrusal programlama kullanıcılarının gücünü hatırı sayılır bir şekilde artırmıştır. Gelecekte tamsayılı doğrusal programlama çözücü programlar tarafından daha çeşitli problem yapılarının çözümünü beklemektedir.

Dal ve sınır algoritmasının davranışı; algoritmanın çok önemli bileşenlerini kontrol eden parametre setleri tarafından değiştirilebilir. Bu değişikliğin yapılması, uygun problem için uygun parametreye karar verilmesi açısından oldukça çaba isteyen bir iştir. Dahası çözüm sürecinde ilk başta uygun olan, sonraki basamaklarda uygun olmayabilir. Tamsayılı programlama sistemlerinin, analize dayalı parametre setlerinin dinamik olarak ayarlanması ihtiyacı vardır.

Bugünün Tamsayılı Programlama sistemleri; dal-sınır ağacı formülasyonuna dinamik olarak kısıt eklenmesine izin vermektedir. Gelecekte Tamsayılı Doğrusal Programlama sistemlerinin, ağaçtaki formülasyona değişken eklenmesine de izin vereceğini istemekte ve algoritmalara daha kolay uygulanmasını beklemektedir.

Tamsayılı doğrusal programlama teknolojisi, gerçek hayat optimizasyon problemlerini çözmede pratik bir araç haline geldikçe, belirsiz bilgileri ve verileri etkili bir şekilde yönetecek yaklaşımlara olan talepte artma olmuştur. Araştırmacılar, bu ihtiyaca cevap verebilmek için çalışmaktadırlar. Bu çabalar, büyük ölçekli problemlerin etkili bir şekilde çözülebilmesi açısından Tamsayılı Doğrusal Programlama alanında ilerlemelere sebep olmuştur. Gelecekte Tamsayılı Doğrusal Programlama yazılımlarının özel olarak tasarlanmış, belirsiz amaç ve teknoloji katsayılarını yönetecek özellikleri artırılabilir

ÜÇÜNCÜ BÖLÜM

3. KULLANDIĞIMIZ KÜTÜPHANEDEKİ ILP FONKSİYONLARI VE İŞLEVLERİ

Projemizde kullanmak için Python programlama dilinin sunduğu **MIP (Mixed-Integer Linear Programming)** ve **pywraplp** kütüphanelerini kullandık. Bu kütüphaneleri ve içerdiği fonksiyonları inceleyecek olursak:

1-) MIP

Python MIP, Karma Tamsayı Doğrusal Programlama problemlerinin modellenmesi ve çözümü için kullanılan Python araçlarının bir koleksiyonudur. MIP sözdizimi Pulp'tan ilham alınmıştır. Tıpkı CyLP gibi MIP de kesim üretimi (cut generation), tembel kısıtlamalar (lazy constraints), MIPstart ve çözüm havuzları (solution pools) gibi gelişmiş çözücü özelliklerine erişim sağlar.

MIP kütüphanesinin sağladığı bazı fonksiyonlar şunlardır:

- minimize (objective)

  Verilen doğrusal ifadeyi minimize etmek için kullanılan amaç fonksiyonunu hazırlamak için kullanılması gereken fonksiyondur.

  Parametre olarak amaç fonksiyonunu alır.

  Geri dönüş tipi olarak *mip.LinExpr (doğrusal ifade)* verir.

- maximize (objective)

  Verilen doğrusal ifadeyi maksimize etmek için kullanılan amaç fonksiyonunu hazırlamak için kullanılması gereken fonksiyondur.

  Parametre olarak amaç fonksiyonunu alır.

  Geri dönüş tipi olarak *mip.LinExpr (doğrusal ifade)* verir.

- xsum (terms)

  Bir toplamdan, doğrusal ifade yaratmak için kullanılması gereken fonksiyondur. Bir Python fonksiyonu olan sum() fonksiyonu da kullanılabilirken xsum(), doğrusal ifadenin hızlıca üretilebilmesi için optimize edilmiş bir versiyonudur.

  Parametre toplanacak terimler grubu (ideal olarak bir liste olur)

  Geri dönüş tipi olarak *mip.LinExpr (doğrusal ifade)* verir.

- compute_features (model)

  Bu fonksiyon, karma tamsayılı doğrusal programlama için örnek özellikler hesaplar. Bu özellikler; satır, sütun sayıları, matris yoğunluğu gibi örnek karakteristiklerdir. Bu özellikler makine öğrenmesi algoritmalarında parametre ayarları önermelerinde kullanılabilir. Hesaplanan özelliklerin isimlerini kontrol etmek için *features()* fonksiyonu kullanılır.

Parametre özelliklerin çıkarılacağı karma tamsayılı doğrusal programlama modeli olur.

Geri dönüş tipi float tipindeki özelliklerin bir listesi olur. (List [float])

- features ()

Bu fonksiyon, hesaplanabilir problem özellikleri isimlerinden oluşan bir liste döndürür.

Parametre almaz.

Geri dönüş tipi str tipindeki isimlerin bir listesi olur. (List [str])

2-) pywraplp

Kütüphanenin sağladığı bazı fonksiyonlar aşağıdaki gibidir:

- ExportModelAsLpFormat (*args)

Güncel modeli (değişken, kısıtlama, amaç), sözde "CPLEX LP dosya formatı" olarak kodlanmış bir string tipinde döndüren fonksiyondur.

LP dosya formatı kolayca okunabilir durumdadır.

Çalıştırma sırasında bir hata olması durumunda "false" döndürür. İsimlerin geçerliliği otomatik olarak kontrol edilir. Eğer bir değişken veya kısıtlama ismi geçerli değilse veya var olmayan bir isimse yeni bir geçerli isim otomatik olarak oluşturulur.

- ExportModelAsMpsFormat (*args)

Güncel modeli (değişken, kısıtlama, amaç), "free" MPS formatı kullanılarak MPS dosya formatında kodlanmış bir string olarak döndüren bir fonksiyondur.

Çalıştırma sırasında bir hata olması durumunda "false" döndürür. Maksimize amaçlı modeller bir hatayı tetikler. Çünkü, MPS yalnızca minimizasyon problemleri için kodlanabilir.

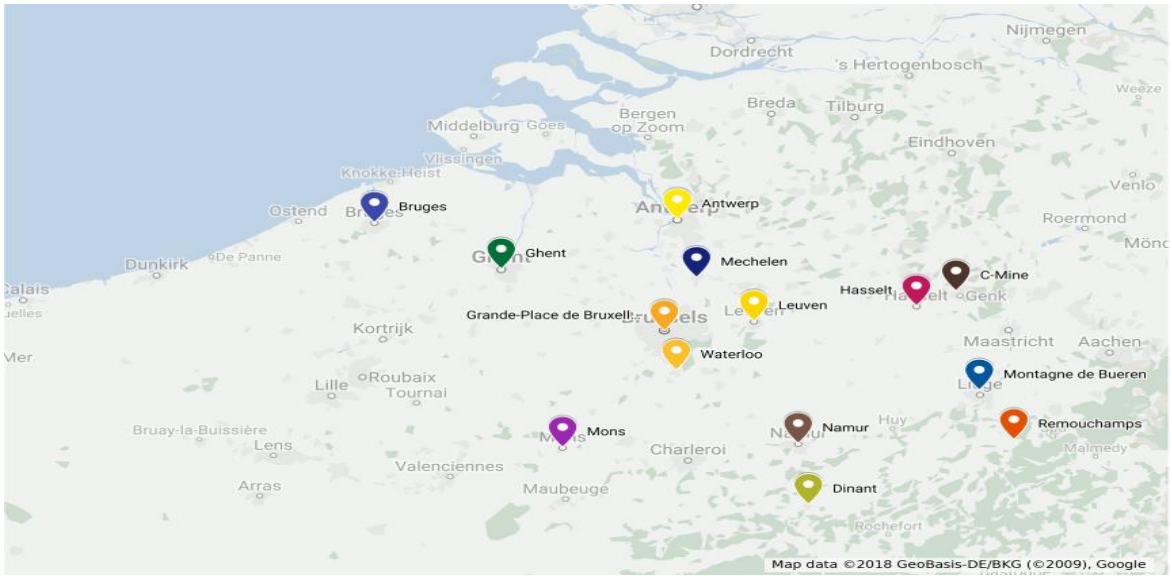
İsimlerin geçerliliği otomatik olarak kontrol edilir. Eğer bir değişken veya kısıtlama ismi geçerli değilse veya var olmayan bir isimse yeni bir geçerli isim otomatik olarak oluşturulur.

DÖRDÜNCÜ BÖLÜM

4. ILP İLE MAX/MİN PROBLEMİ ÇÖZÜMÜ

- **Gezgin Satıcı Problemi (The Traveling Salesman Problem)**

Tam sayılı doğrusal programlamanın en çok kullanıldığı alanlardan biri olan Gezgin Satıcı Problemi (TSP), 50'li yıllardaki ilk hesaplama çalışmalarından beri üzerinde en çok çalışılan kombinasyonel optimizasyon problemlerinden biridir. Bu problemi anlaşılabilmesi için Belçika'da biraz zaman geçirileceği ve aşağıdaki haritada belirtilen bazı önemli turistik yerlerin ziyaret edileceği düşünülebilir:



Tüm konumları ziyaret etmek için mümkün olan en kısa yolun bulunması istiyoruz. Formülize olarak;

Minimize:

$$\sum_{i \in I, j \in I} c_{i,j} . x_{i,j}$$

Subject to:

$$\sum_{j \in V \setminus \{i\}} x_{i,j} = 1 \ \forall i \in V$$

$$\sum_{i \in V \setminus \{j\}} x_{i,j} = 1 \ \forall j \in V$$

$$y_i - (n+1) . x_{i,j} \geq y_j - n \ \forall i \in V \setminus \{0\}, j \in V \setminus \{0, i\}$$

$$x_{i,j} \in \{0, 1\} \ \forall i \in V, j \in V$$

$$y_i \geq 0 \ \forall i \in V$$

şeklinde ifade edilebilir.

Burada, kısıtlamaların ilk ikisi, her nokta için yalnızca bir kez varış ve ayrılış olmasını zorunlu kılar. Yalnızca bu kısıtlamaları dahil ederek bu problem için en uygun çözüm aşağıdaki gösterildiği gibi alt-turlardan oluşan bir çözümle sonuçlanabilir.

Bağlantılı rotaların üretimini zorlamak için ek değişkenler ($y_i \geq 0$), üretilen rotadaki her bir noktanın sırasal düzenini belirten modele dahil edilmiştir. Sıfır noktası başlangıç noktası olarak keyfi bir şekilde seçilir ve $x_{i,j}, y_i \quad y_j$ değişkenlerini bağlayan koşulsal kısıtlamalar, başlangıç düğümü hariç tüm düğümler için yaratılır. TSP için en uygun rotanın yaratılması, optimize edilmesi ve yazdırılması için gereken Python kodu aşağıda verilmiştir:

```python
from itertools import product
from sys import stdout as out
from mip import Model, xsum, minimize, BINARY

# names of places to visit
places = ['Antwerp', 'Bruges', 'C-Mine', 'Dinant', 'Ghent',
          'Grand-Place de Bruxelles', 'Hasselt', 'Leuven',
          'Mechelen', 'Mons', 'Montagne de Bueren', 'Namur',
          'Remouchamps', 'Waterloo']

# distances in an upper triangular matrix
dists = [[83, 81, 113, 52, 42, 73, 44, 23, 91, 105, 90, 124, 57],
         [161, 160, 39, 89, 151, 110, 90, 99, 177, 143, 193, 100],
         [90, 125, 82, 13, 57, 71, 123, 38, 72, 59, 82],
         [123, 77, 81, 71, 91, 72, 64, 24, 62, 63],
         [51, 114, 72, 54, 69, 139, 105, 155, 62],
         [70, 25, 22, 52, 90, 56, 105, 16],
         [45, 61, 111, 36, 61, 57, 70],
         [23, 71, 67, 48, 85, 29],
         [74, 89, 69, 107, 36],
         [117, 65, 125, 43],
         [54, 22, 84],
         [60, 44],
         [97],
         []]

# number of nodes and list of vertices
n, V = len(dists), set(range(len(dists)))

# distances matrix
c = [[0 if i == j
      else dists[i][j-i-1] if j > i
      else dists[j][i-j-1]
      for j in V] for i in V]
```

,

```
model = Model()

# binary variables indicating if arc (i,j) is used on the route or not
x = [[model.add_var(var_type=BINARY) for j in V] for i in V]

# continuous variable to prevent subtours: each city will have a
# different sequential id in the planned route except the first one
y = [model.add_var() for i in V]

# objective function: minimize the distance
model.objective = minimize(xsum(c[i][j]*x[i][j] for i in V for j in V))

# constraint : leave each city only once
for i in V:
    model += xsum(x[i][j] for j in V - {i}) == 1

# constraint : enter each city only once
for i in V:
    model += xsum(x[j][i] for j in V - {i}) == 1

# subtour elimination
for (i, j) in product(V - {0}, V - {0}):
    if i != j:
        model += y[i] - (n+1)*x[i][j] >= y[j]-n

# optimizing
model.optimize()

# checking if a solution was found
if model.num_solutions:
    out.write('route with total distance %g found: %s'
              % (model.objective_value, places[0]))
    nc = 0
    while True:
        nc = [i for i in V if x[nc][i].x >= 0.99][0]
        out.write(' -> %s' % places[nc])
        if nc == 0:
            break
    out.write('\n')
```

$x + 7y$	≤	17.5
x	≤	3.5
x	≥	0
y	≥	0
x, y integers		

Problem: Yukarıdaki kısıtlamalara göre x + 10y işlemini maksimize ediniz.

- ***Kısıtlamalar doğrusal olduğundan, bu çözümlerin sadece birer tamsayı olması gereken bir doğrusal optimizasyon problemidir***

- **Doğrusal Çözücü Paketinin Dahil Edilmesi**

Bir ILP probleminin çözülebilmesi için programa önce doğrusal çözücü paketi olan "pywraplp" kütüphanesi dahil edilmelidir.

```
from ortools.linear_solver import pywraplp
```

- **MIP Çözücünün Tanımlanması**

Kullanılmak istenen MIP çözücüsü tanımlanır. Aşağıda verilen kod parçasında SCIP (Solving Constraint Integer Programs) tanımlanmıştır:

```
# Create the mip solver with the SCIP backend.
solver = pywraplp.Solver.CreateSolver('SCIP')
```

- **MIP Çözücünün Çağırılması**

Aşağıdaki kod parçasında MIP çözücü çağırılır:

```
status = solver.Solve()
```

- **Değişkenlerin Tanımlanması**

Aşağıdaki kod parçasında değişkenlerin tanımlanması gösterilmiştir:

```
infinity = solver.infinity()
# x and y are integer non-negative variables.
x = solver.IntVar(0.0, infinity, 'x')
y = solver.IntVar(0.0, infinity, 'y')

print('Number of variables =', solver.NumVariables())
```

Program, negatif olmayan tam sayı değerleri alan x ve y değişkenlerini yaratmak için MakeItVar metodunu kullanır.

- **Kısıtlamaların Tanımlanması**

Aşağıdaki kod parçasında problemin kısıtlamaları belirlenmiştir:

```
# x + 7 * y <= 17.5.
solver.Add(x + 7 * y <= 17.5)

# x <= 3.5.
solver.Add(x <= 3.5)

print('Number of constraints =', solver.NumConstraints())
```

- **Amaç Fonksiyonunun Tanımlanması**

Aşağıda problemin amaç fonksiyonu tanımlanmıştır.

```
# Maximize x + 10 * y.
solver.Maximize(x + 10 * y)
```

- **Sonucun Gösterilmesi**

Aşağıdaki kod parçasında problemin sonucu gösterilmiştir:

```
if status == pywraplp.Solver.OPTIMAL:
    print('Solution:')
    print('Objective value =', solver.Objective().Value())
    print('x =', x.solution_value())
    print('y =', y.solution_value())
else:
    print('The problem does not have an optimal solution.')
```

```python
from ortools.linear_solver import pywraplp


def main():
    # Create the mip solver with the SCIP backend.
    solver = pywraplp.Solver.CreateSolver('SCIP')

    infinity = solver.infinity()
    # x and y are integer non-negative variables.
    x = solver.IntVar(0.0, infinity, 'x')
    y = solver.IntVar(0.0, infinity, 'y')

    print('Number of variables =', solver.NumVariables())

    # x + 7 * y <= 17.5.
    solver.Add(x + 7 * y <= 17.5)

    # x <= 3.5.
    solver.Add(x <= 3.5)

    print('Number of constraints =', solver.NumConstraints())
```

```python
 # Maximize x + 10 * y.
    solver.Maximize(x + 10 * y)

    status = solver.Solve()

    if status == pywraplp.Solver.OPTIMAL:
        print('Solution:')
        print('Objective value =', solver.Objective().Value())
        print('x =', x.solution_value())
        print('y =', y.solution_value())
    else:
        print('The problem does not have an optimal solution.')

    print('\nAdvanced usage:')
    print('Problem solved in %f milliseconds' % solver.wall_time())
    print('Problem solved in %d iterations' % solver.iterations())
    print('Problem solved in %d branch-and-bound nodes' %
solver.nodes())


if __name__ == '__main__':
    main()
```

Programın Çıktısı Şu Şekildedir:

```
Number of variables = 2
Number of constraints = 2
Solution:
Objective value = 23
x = 3
y = 2
```

Amaç fonksiyonunun optimal değeri: x=3, y=2 noktasında oluşan 23'tür.

KAYNAKÇA

- **Taha, Hamdy, "Yöneylem Araştırmaları", Literatür**
- **Öztürk, Ahmet, "Yöneylem Araştırması", Ekin**
- **Doç.Dr.Mehmet Hakan Satman Yöneylem Araştırmaları I Ders Notları**
- **https://buildmedia.readthedocs.org/media/pdf/python-mip/latest/python-mip.pdf**
- **https://www.isical.ac.in/~arijit/courses/autumn2016/ILP-Lecture-1.pdf**
- **Genova, Krasmira and Guliashki, Vassil, "Linear Integer Programming Methods and Approaches-A Survey", Cybernetics and Information Technologies · Ocak 2011**
- **https://google.github.io/or-tools/python/ortools/linear_solver/pywraplp.html**
- **https://python-mip.readthedocs.io/en/latest/quickstart.html**
- **Bakır, M. A., Altunkaynak, B., Tamsayılı Programlama Teorisi, Modeller ve Algoritmalar, Nobel Yayınevi, Ankara, 2003.**