ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR BİLİMLERİ
ANABİLİM DALI

DOĞRUSAL PROGRAMLAMA VE REVISED SIMPLEKS YÖNTEME DAYALI BİR BİLGİSAYAR PROGRAMI GELİŞTİRME UYGULAMASI

Pakize TOHUMCU

Yönetici:
Prof.Dr. Sibkat KAÇTIOĞLU

Yüksek Lisans Tezi

ÖZET

Doğrusal programlama, yaygın olarak kullanılan yöneylem araştırması optimizasyon tekniklerinden birisidir.

Çalışmada önce Doğrusal Programlama Modeli tanıtılmış, çözüm teknikleri üzerinde durulmuş ve bu modeller için söz konusu olabilen bazı özel durumlar incelenmiş, daha sonra da QBASIC programlama dili kullanılarak Revised Simpleks Algoritmaya dayalı bir bilgisayar programı (ATADP) geliştirilmiştir.

Geliştirilen programın kişisel bilgisayarlarda yaygın olarak kullanılan QSB ile mukayesesi programın QSB'ye göre önemli avantajlar sağladığını göstermiştir.

SUMMARY

Linear programing is one of the widely used operation research optimization techniques.

In this study , first linear programming model was introduced, solution techniques were reviewed and some special cases, which can be possible for these models, were examined, then a computer programme (ATADP) based on Revised Simpleks Algorithm was developed by using QBASIC programming language.

The comparison of the developed programme with the QSB programme ,which is widely used for similar purpose in PC, showed that the programme had some important advantages over QSB.

TEŞEKKÜR

Bu tezin hazırlanmasında hem çalışmalarından hem de bilgi ve tecrübelerinden büyük ölçüde istifade ettiğim sayın hocam Prof.Dr. Sibkat KAÇTIOĞLU'na teşekkür ederim. Ayrıca yüksek lisans çalışmalarım esnasında bilgilerinden yararlandığım ve her türlü desteği gördüğüm hocalarım Prof.Dr.Fatin SEZGİN ve Prof.Dr Yahya Kemal YOĞURTÇU'ya teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

1. GİRİŞ

Yöneylem araştırması ekonomilerin ve işletmelerin başta gelen amacı olan kaynakların en verimli şekilde kullanılmasını sağlamak üzere geliştirilmiş kantitatif karar verme tekniklerinden oluşur. Bilimsel yaklaşımla karar verme olarak da tanımlayabileceğimiz yöneylem araştırmasının amacı, yönetimlerin politika ve eylemlerinin bilimsel olarak belirlenmesini sağlamak, verilecek kararların tutarlılık ve uygulanabilirliğini artırmaktır.

Yöneylem araştırması doğrusal ve doğrusal olmayan programlama, markov zincirleri, kuyruk teorisi, stok kontrolü, simülasyon, dinamik programlama ve kuadratik programlama gibi çok sayıda yöntemi kapsar. Bunlardan doğrusal programlama bu bilim dalının geliştirdiği yaygın kullanım alanına sahip bir optimizasyon tekniğidir. Doğrusal programlama modellerinin çözümünde simpleks metod kullanılır. Tezin konusu olan Revised (Düzeltilmiş) Simpleks Metod ise simpleks metodun geliştirilmiş özel bir şeklidir. Doğrusal programlama problemlerinin bilgisayar desteği ile çözümünde tercih edilen bir yöntemdir.

Çalışmanın ikinci bölümünde Doğrusal Programlama modeli hakkında genel bilgiler verilmiş, bu modelin çözüm tekniklerinden olan Simpleks Metod ve özellikle Revised Simpleks Metod incelenmiştir. Üçüncü bölümde ise Revised Simpleks çözüm metodunu kullanarak, 100x100 boyuta kadar olan doğrusal programlama problemlerini çözmek üzere QBASIC'te geliştirilen ATADP programı incelenmiştir. Dördüncü bölümde ise sonuç olarak, hazırlanan program yaygın olarak kullanılan benzer amaçlı bir program ile karşılaştırılmıştır.

2. DOĞRUSAL PROGRAMLAMA

2.1. Giriş

Herhangi bir karar problemini yöneylem araştırması yoluyla çözmenin temeli model kurmadır. Gerçek hayattaki olayların soyut bir görünümünü tasvir anlamı taşıyan model kurma, fiziksel bilimlerde söz konusu olan deney yapmanın yönetim bilimindeki karşılığıdır (1).

Yöneylem araştırmalarında verilen bir problemin çözülebilmesi için;

- Problemin formüle edilmesi
- Modelin kurulması
- Modelin çözümü
- Çözümün test edilmesi
- Çözümün kontrol altına alınması
- Çözümün uygulanması gerekir.

Model kurma aşamasında dikkat edilmesi gereken hususlar ise şunlardır. Model;

-Sistemin ve karar organlarının amaçlarını en iyi şekilde temsil etmelidir.

- Gerçeği en uygun şekilde temsil etmelidir. Hiçbir gerçek hayat problemi tam olarak modellenemez. Ancak belirli sınırlar ve varsayımlarla modellenirler.

-Mevcut çözüm teknikleriyle çözülebilmelidir.

-Gerekli olan veriler bulunmalıdır.

2.2. Doğrusal Programlama Modeli

2.2.1. Tanım

Doğrusal programlama modeli , doğrusal eşitlik veya eşitsizlik sınırlayıcı şartları altında doğrusal bir amaç fonksiyonunu optimize(maksimize veya minimize) etmeye yarar. Genel bir doğrusal programlama modeli üç unsurdan oluşur. Bunlar optimize edilecek olan amaç fonksiyonu, kısıtlayıcı şartlar ve pozitiflik şartlarıdır (2).

2.2.2. Amaç Fonksiyonu, Kısıtlayıcılar ve Pozitiflik Şartı

Eldeki sınırlı kaynakların en iyi dağılımını belirleme tekniği olarak tanımlanabilen doğrusal programlama modelinde optimize edilecek olan büyüklük amaç fonksiyonudur. Her kuruluşun bir amacı vardır ve bu amacı açık bir şekilde tanımlamak mümkündür. Bir fabrikada amaç bir ürünü minimum maliyetle üretmek olduğu gibi, bir işletmedeki amaç maksimum kar sağlamak olabilir. Amaç fonksiyonu matematiksel olarak belirlenebilen tek bir fonksiyondur. Fonksiyon f, değişkenler x_1, x_2, ...x_n ve değişkenlere ait katsayılar da c_1, c_2, ...c_n ile gösterilirse amaç foksiyonu;

$$\text{(Maksimize veya minimize) } f = c_1 x_1 + c_2 x_2 + \ldots c_n x_n \qquad (2.1)$$

şeklinde ifade edilir.

Amaç, bu fonksiyonu maksimum veya minimum kılan x_1, x_2, ...x_n değerlerinin bulunmasıdır. x_1, x_2, ...x_n için kısıtlayıcı şart olmadığı takdirde bu f fonksiyonunun maksimumu pozitif sonsuzda minimumu ise negatif sonsuzda olacaktır. Halbuki hiçbir kaynak sınırsız değildir, kısıtlayıcı şartları vardır.

Bir doğrusal programlama modelinde kısıtlayıcı şartlar;

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \dots a_{1n}x_n\,(<,=,>)\ b_1\\
&a_{21}x_1 + a_{22}x_2 + \dots a_{2n}x_n\,(<,=,>)\ b_2\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&a_{m1}x_1 + a_{m2}x_2 + \dots a_{mn}x_n\,(<,=,>)\ b_m
\end{aligned}
\tag{2.2}
$$

şeklinde ifade edilir.

Doğrusal programlama konusunun temel kavramlarından birisi olan "kısıtlayıcılar" deyimi ile eldeki sınırlı kaynaklarla karar verme durumundaki kişi veya grupların çoğu zaman kontrolleri altında bulunan ekonomik değer veya güçlerin sınırları ifade edilmek istenmektedir.

Bir doğrusal programlama modelinde x_1, x_2, ...x_n değişkenleri gerçek hayatta hiçbir zaman negatif değer alamayacağından bu değişkenlerin hepsinin en azından sıfır veya sıfırdan büyük olma şartları vardır.

$$x_1,\ x_2,\ \dots x_n >= 0 \tag{2.3}$$

Bu şart doğrusal programlama problemlerinin pozitiflik şartı olarak bilinir.

2.2.3. Doğrusal Programlama Modelinin Dayandığı Varsayımlar

Bir matematiksel model belli varsayımlar altında geçerlidir. Problem bu varsayımları sağladığı sürece model tarafından en iyi şekilde temsil edilebilir ve modelin çözümü ile elde edilen sonuçlar anlamlı olabilir. Bu varsayımlar şu şekilde sıralanabilir.

2.2.3.a. Oranlılık(Doğrusallık)

Her bir karar değişkeninin alacağı değere göre amaç fonksiyonu ve kısıtlayıcılar belli ve doğrusal bir şekilde etkilenirse modelin bu varsayımı sağladığı kabul edilir.

2.2.3.b. Toplanabilirlik

Bu varsayımda modeldeki değişkenlerin birbirini etkilemediği kabul edilir. Amaç ve kısıtlayıcılar her değişkenden gelen katkıların toplanması ile oluşuyorsa toplanabilirlik varsayımının sağlandığı kabul edilir.

2.2.3.c. Bölünebilirlik

Modeldeki karar değişkenlerinin her türlü reel değeri alabilmesi halinde bölünebilirliğin sağlandığı kabul edilir. Gerçek hayatta mutlaka tam sayı ile ifade edilmesi gereken karar değişkenlerine sahip problemler bu varsayımı sağlayamadıklarından doğrusal programlama ile çözülemezler. Bu tür problemlerde tamsayılı programlama tekniklerinden faydalanılır.

2.2.3.d. Belirlilik

Modelin tüm katsayılarının ve sağ taraf değerlerinin bilinen değerler olması halinde belirlilik özelliği sağlanır. Gerçek hayatta bu değerler genellikle kesin ve belirli değildir. Bu durumlarda probleme duyarlılık analizi teknikleri ile çözüm getirilmeye çalışılır.

2.2.4. Bazı Özel Modeller

Burada doğrusal programlamanın iki özel hali olan tipik maksimizasyon ve tipik minimizasyon modelleri ele alınacaktır.

2.2.4.a. Tipik Maksimizasyon Modeli

Bir doğrusal programlama modelinde amaç maksimizasyon ve bütün kısıtlayıcılar küçük-eşit şeklinde ise bu tür modellere tipik maksimizasyon modeli denir. Tipik bir maksimizasyon modeli:

Maksimize

$f = c_1 x_1 + c_2 x_2 + \dots c_n x_n$

Kısıtlayıcı şartlar: (2.4)

$a_{11} x_1 + a_{12} x_2 + \dots a_{1n} x_n < b_1$

$a_{21} x_1 + a_{22} x_2 + \dots a_{2n} x_n < b_2$

....

$a_{m1} x_1 + a_{m2} x_2 + \dots a_{mn} x_n < b_n$

Pozitiflik şartı;

$x_1 , x_2 , \dots x_n > 0$

şeklinde ifade edilir.

2.2.4.b. Tipik Minimizasyon Modeli

Bir doğrusal programlama modelinde amaç minimizasyon ve bütün kısıtlayıcılar büyük-eşit şeklinde ise bu tür modellere tipik minimizasyon modeli denir. Tipik bir maksimizasyon modeli aşağıdaki şekilde ifade edilir.

Minimize

$f = c_1 x_1 + c_2 x_2 + \dots c_n x_n$

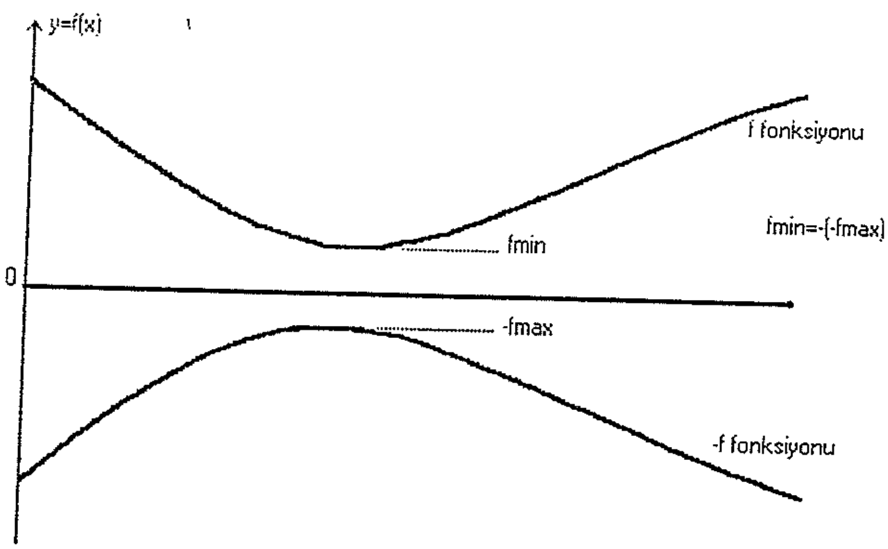
Kısıtlayıcı şartlar:

$$a_{11}\,x_1 + a_{12}\,x_2 + \ldots a_{1n}\,x_n > b_1$$
$$a_{21}\;x_1 + a_{22}\,x_2 + \ldots\; a_{2n}\,x_n > b_2 \qquad (2.5)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{m1}\,x_1 + a_{m2}\,x_2 + \ldots\; a_{mn} > b_m$$

ve pozitiflik şartı ;

$$x_1\,,\,x_2\,,\,\ldots x_n > 0$$

Verilen bu iki model doğrusal programlamada iki özel modeldir. Fakat kısıtlayıcılar her zaman aynı yönde olmazlar. Genel bir doğrusal programlama modelinde her türlü kısıtlayıcı bulunur. Ancak genel bir formülasyon için amaç fonksiyonu her zaman maksimizasyon alınır. Amaç minimizasyonsa, minimize amaç fonksiyonu (-1) ile çarpılarak maksimize hale çevrilir. Bir fonksiyonun minimum noktası bu fonksiyonun ters işaretlisinin maksimum noktasının ters işaretlisine eşit olduğundan bulunan maksimum amaç değeri (-1) ile çarpılarak amaç fonksiyonunun minimum değeri bulunur.



Şekil 2.1. Minimize amaç fonksiyonu ile maksimize amaç fonksiyonu arasındaki ilişki

2.2.5. Genel Bir Doğrusal Programlama Modeli (Standart Model)

Genel bir doğrusal programlama modeli şu şekilde ifade edilir.

maksimize

$f = c_1x_1 + c_2x_2 + ... + c_nx_n$

Kısıtlayıcı şartlar:

$a_{11}x_1 + a_{12}x_2 + ... a_{1n}x_n < b_1$

..................................k tane küçük eşit kısıtlayıcı

$a_{k1}x_1 + a_{k2}x_2 + ... a_{kn}x_n < b_k$

....

$a_{k+1,1}x_1 + a_{k+1,2}x_2 + ... a_{k+1,n}x_n = b_{k+1}$

.....................................l - k tane eşit kısıtlayıcı

$a_{l1}x_1 + a_{l2}x_2 + ... a_{ln}x_n = b_l$

....

$a_{l+1,1}x_1 + a_{l+1,2}x_2 + ... a_{l+1,n}x_n > b_{l+1}$

....m – l tane büyük eşit kısıtlayıcı

$a_{m1}x_1 + a_{m2}x_2 + ... a_{mn}x_n > b_m$

Pozitiflik şartı: (2.6)

$x_1, x_2, ... x_n > 0$

2.3. Doğrusal Programlama Modellerinin Çözümü

Doğrusal programlama modellerinin çözümünde çok çeşitli metodlar geliştirilmiştir. Aşağıda bu metodlarda kullanılan bazı kavramlar üzerinde durulacaktır.

2.3.1. Doğrusal Programlama Modelinde Kullanılan Bazı Kavramlar

2.3.1.a. Konveks Set

S bir nokta kümesini göstermek üzere bu küme içinde bulunan bütün P_1, P_2 nokta çiftlerini birleştiren doğru parçaları S kümesi içinde kalıyorsa bu küme konveks settir.

2.3.1.b. Uygun Çözüm

Bir doğrusal karar modelinin tüm sınırlayıcı ve pozitiflik şartlarını sağlayan her x vektörüne bir uygun çözüm, bu uygun çözümlerin oluşturduğu kümeye ise uygun çözüm alanı denir. Bu uygun çözümler kümesi bir konveks set oluşturur ve doğrusal karar modelinin uygun çözüm alanı dışbükey kümedir (3).

2.3.1.c. Temel Çözüm ve Temel Uygun Çözüm

Değişken sayısı n, kısıtlayıcı sayısı m (n>m) olan bir doğrusal program modelinde n-m adet değişken sıfır olarak alınırsa m bilinmeyenli m adet denklem elde edilmiş olacaktır. Bu denklem sisteminin katsayılar determinantı sıfırdan farklı olmak şartıyla elde edilecek çözüme Temel Çözüm denir. Temel çözümlerden pozitiflik şartını sağlayanlara ise Temel Uygun Çözüm (TUÇ) denir. n değişkenli m kısıtlayıcılı bir doğrusal modelde n-m adet bağımsız değişken ;

n!/(m!(n-m)!) farklı şekilde seçilebileceğinden bu kadar temel çözüm mevcuttur.

2.3.1.d. Optimal Çözüm

Doğrusal Programlama modelinde amaç fonksiyonunu optimize (maksimize veya minimize) eden çözüme optimum çözüm (OÇ) denir. Optimum çözüm TUÇ'lerden biridir (4).

2.3.2. Doğrusal Programlama Modellerinin Çözüm Metodları

Doğrusal programlama modellerinin çözümünde çeşitli metodlar geliştirilmiştir. Grafik çözüm metodu, simpleks çözüm metodu ve revised (düzeltilmiş) simpleks çözüm metodu bu metodlardandır. Simpleks Metod da kendi içinde Big M metodu ve İki Evreli Simpleks Metod olarak ikiye ayrılır. Burada her iki metoddan da bahsedilecektir.

2.3.2.a. Grafik Çözüm Metodu

Anlaşılması en kolay çözüm metodudur. Ancak değişken sayısı üçü aşan modellerde kullanılamaz. Genellikle iki değişkenli modellerin çözümünde kullanılan bir metoddur.

Grafik çözüm metodunda doğrusal modellerin pozitiflik şartından dolayı koordinat sisteminin sadece sağ üst kısmı kullanılır. Çözüm değerlerinin de bu bölgede yer alması gerekir. Kısıtlayıcıların ifade ettiği eşitlik veya eşitsizlikler koordinat sistemine yerleştirilir. Çözüm eşitsizliklerin ve pozitiflik şartının sınırladığı konveks alan içerisinde aranır. Uygun çözüm alanını gösteren bu alanın köşe noktaları temel uygun çözümlerdir. Bu köşe noktalarının herbirine karşılık gelen x_1 ve x_2 değerleri bulunarak amaç fonksiyonunda yerine konur. Amaca göre en büyük değeri veya en küçük değeri sağlayan köşe seçilir.

2.3.2.b. Simpleks Çözüm Metodu

Simpleks metod, bir başlangıç temel uygun çözümünden hareketle her adımda yeni bir temel uygun çözüme geçerek belli sayıdaki iterasyonlar sonunda optimal çözüme ulaşır.

n adet değişkene ve m adet kısıtlayıcıya sahip genel bir doğrusal programlama modelinde kısıtlayıcıların bir kısmı küçük-eşit, bir kısmı eşit ve bir kısmı da büyük eşit şeklindedir . Burada (2.6) sistemindeki genel doğrusal programlama modeli ele alınacaktır. Bu modeli simpleks metodla çözmek için önce eşitsizlikler eşitlik haline getirilir. Küçük-eşitlikleri eşitlik haline çevirirken y_i boş değişkenleri (slack variable) kullanılır. Bu değişkenlerin kısıtlayıcılardaki katsayısı bir, amaç fonksiyonundaki katsayıları ise sıfırdır. Bu sebeple amaç fonksiyonunda gösterilmezler. Boş değişkenler gerçek hayat problemlerinde kullanılmayan kapasiteleri gösterirler. Eşitlik şeklindeki kısıtlayıcıların da kanonik bir form oluşturması için z_i yapay değişkenleri (artificial variable) eklenir. (Doğrusal denklem sisteminin herbir denkleminde, değişkenlerden bir tanesinin +1 katsayı ile yalnızca o denklemde bulunup, diğer denklemlerde bulunmaması durumunda denklem sistemi kanonik formdadır denir.) Büyük eşit kısıtlayıcıları eşitlik haline getirmek için bu kısıtlayıcılardan y_i boş değişkenleri çıkarılır. Kanonik bir form oluşturması için z_i yapay değişkenleri eklenir.

Eşit ve büyük eşit kısıtlayıcılara eklenen z_i yapay değişkenlerinin bir an önce temelden çıkarılmaları gerekir. Bu yapay değişkenler temelden çıkarılamaz ise modelin çözümü yoktur. Bunu sağlamak üzere z_i değişkenlerinin toplamının minimize (veya toplamının ters işaretlisinin maksimize) edilmesi gerekir. Genel bir modelde amaç fonksiyonu maksimizasyon türünde olduğu için z_i ler toplamının ters işaretlisi aynı amaç fonksiyonuna eklenerek maksimize edilir. Bu aşamadan sonra z_i değişkenlerinin bir an önce temelden atılması için iki metod vardır. Bunlardan ilki big M metodu, ikincisi de iki evreli simpleks metoddur.

Big M metoduna göre z_i yapay değişkenlerin bir an önce temelden atılabilmesi için bu değişkenler amaç fonksiyonunda negatif değerli ve çok büyük katsayılı olmalıdırlar. M>0 olmak üzere;

$$\text{Maksimize } f = c_1 x_1 + c_2 x_2 + \ldots c_n x_n - Mz_1 - Mz_2 - \ldots Mz_{m-k} \quad (2.7)$$

olur. Amaç fonksiyonu bu şekilde oluşturulduktan sonra simpleks tablo düzenlenerek simpleks çözüm işlemlerine başlanır. Bu yaklaşıma yapay değişkenlere amaç fonksiyonunda verilen birim katkıların M olmasından dolayı Big M metodu denir.

z_i değişkenlerini bir an önce temelden atmak için kullanılan diğer bir metod ise iki evreli simpleks metoddur. Bu metod (2.7) de verilen amaç fonksiyonunu maksimize edilecek iki amaç fonksiyonu gibi düşünür.

$$\begin{aligned} f_c &= c_1 x_1 + c_2 x_2 + \ldots c_n x_n \\ f_m &= -Mz_1 - Mz_2 - \ldots Mz_{m-k} \end{aligned} \quad (2.8)$$

Amaç fonksiyonlarından biri değişkenlerin oluşturduğu amaç fonksiyonu, diğeri ise yapay değişkenlerin oluşturduğu amaç fonksiyonudur. Metodun ilk aşamasında yapay değişkenlerin oluşturduğu amaç fonksiyonu ele alınır ve tüm yapay değişkenler temelden atılmaya çalışılır. Bu başarılırsa metodun ikinci aşamasına geçilerek amaç fonksiyonu optimize edilmeye çalışılır.Bu çözüm şeklinde öncelikli amaç yapay değişkenleri bir an önce temelden atmak olduğu için artık Big M metodundaki M katsayılarına ihtiyaç duyulmaz. (Kaçtıoğlu,1984)

Bu durumda f_m fonksiyonu aşağıdaki gibi olur.

$$f_m = -z_1 - z_2 - \ldots z_{m-k} \qquad (2.9)$$

Amaç fonksiyonu bu şekilde düzenlenir ve kısıtlayıcılara gerekli olan yapay ve boş değişkenler eklenirse (2.6) sistemi aşağıdaki hale gelmiş olur.

maksimize

$$-c_1x_1 - c_2x_2 - \ldots - c_nx_n + f_c = 0$$

$$z_1 + z_2 + \ldots z_{m-k} + f_m = 0$$

Kısıtlayıcı şartlar:

$$a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n + y_1 = b_1$$

..................................k tane küçük eşit kısıtlayıcı

$$a_{k1}x_1 + a_{k2}x_2 + \ldots a_{kn}x_n + y_k = b_k$$

....

$$a_{k+1,1}x_1 + a_{k+1,2}x_2 + \ldots a_{k+1,n}x_n + z_1 = b_{k+1}$$

......................................l - k tane eşit kısıtlayıcı

$$a_{l1}x_1 + a_{l2}x_2 + \ldots a_{ln}x_n + z_{l-k} = b_l$$

....

$$a_{l+1,1}x_1 + a_{l+1,2}x_2 + \ldots a_{l+1,n}x_n - y_{k+1} - z_{l-k+1} = b_{l+1}$$

......................................m – l tane buyük eşit kısıtlayıcı

$$a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n - y_{k+m-l} + z_{m-k} = b_m$$

ve

Pozitiflik şartı: (2.10)

$$x_1, x_2, \ldots x_n > 0$$

Bu durumda sistemin temel değişkenleri f_c , f_m , y_1 , y_2 ,...y_k , z_1 , z_2,...z_{m-k} 'dır. x_1, x_2 ,...x_n, y_{k+1}, y_{k+2},...y_{m-l+k} ise temele girecek değişkenlerdir.

Elde edilen (2.10) sistemi satırları temel değişkenler, sütunları temele girecek değişkenler ve temel değişkenler olmak üzere simpleks tabloya yerleştirilir. f_m satırındaki z_i yapay değişken katsayılarından dolayı, temel değişkenlere ait sütunların bir birim matris oluşturmadığı görülür. (Tablo 2.1 Kuruluş Tablosu) Sistemi kanonik forma sokabilmek için tabloda yapay değişkenlere ait satırlar f_m satırından çıkarılır. Böylece f_m satırında ki yapay değişkenlere ait değerler sıfırlanmış olur. Bu şekilde elde edilen yeni tabloya da başlangıç tablosu denir. (Tablo 2.2 Başlangıç tablosu)

Tablo 2.1 Kuruluş Tablosu

Temel Değ.	Temel Değ.Değ	x_1	...	x_n	f_c	f_m	y_1	..	y_k	z_1	..	z_{m-k}
f_c	0	c_1	...	c_n	1	0	0	0	0	0	0	0
f_m	0	0	0	0	0	1	0	0	0	1	1	1
y_1	b_1	a_{11}	...	a_{1n}	0	0	1	0	0	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	..	
y_k	b_k	a_{k1}	...	a_{kn}	0	0	0	0	1	0	0	0
z_1	b_{k+1}	$a_{k+1,1}$	...	$a_{k+1,n}$	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
z_{m-k}	b_{m-k}	a_{m1}	.	a_{mn}	0	0	0	0	0	0	0	1

Bilindiği gibi büyük eşit kısıtlayıcıları eşitlik haline getirmek için kısıtlayıcılardan y_i boş değişkenleri çıkarılmakta idi. Simpleks metodda bu boş değişkenler temel olmayan değişkenler olarak ele alınır. Tabloda fazla yer kaplamaması için (-1) katsayılı olan bu değişkenler gösterilmemiştir.

Bundan sonraki tablolarda temel değişkenler yerine TD, temel değişken değerleri yerine ise TDD yazılacaktır.

Tablo 2.2 Başlangıç Tablosu

TD	TDD	x_1	...	x_n	f_c	f_m	y_1	...	y_k	z_1	...	z_{m-k}
f_c	0	c_1	...	c_n	1	0	0	0	0	0	0	0
f_m	0	$-a_{k+1,1}...-a_{m1}$	...	$-a_{k+1,n}...-a_{mn}$	0	1	0	0	0	0	0	0
y_1	b_1	a_{11}	...	a_{1n}	0	0	1	0	0	0	0	0
.	.	.	...	.	.	.	.	.	.	.	.	.
.	.	.	...	.	.	.	.	.	.	.	.	.
y_k	b_k	a_{k1}	...	a_{kn}	0	0	0	0	1	0	0	0
z_1	b_{k+1}	$a_{k+1,1}$	...	$a_{k+1,n}$	0	0	0	0	0	1	0	0
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
z_{m-k}	b_{m-k}	a_{m1}	.	a_{mn}	0	0	0	0	0	0	0	1

Görüldüğü gibi sistem artık kanonik formdadır. Amaç fonksiyonu da fc ve fm olarak ikiye ayrılmıştır. Önce fm satırı amaç satır olarak kabul edilip, optimize edilmeye çalışılır. fm optimize edildiğinde temelde yapay değişken varsa modelin çözümü yoktur. Tüm yapay değişkenler temelden atılmış ise fm'e ait satır ve yapay değişkenlere ait sütunlar silinerek ikinci evreye geçilir. İkinci evrede fc satırı amaç satır olarak ele alınıp optimize edilir. Optimal çözüm bulunduğunda;

$f_{max} = f_{c\ max}$ olur.

Simpleks metodda her bir iterasyonda yeni bir simpleks tablo oluşturulur. Temele girecek ve temelden ayrılacak değişkenler o iterasyona ait tablodan belirlenerek model optimize edilmeye çalışılır.

Amaç fonksiyonları optimize edilirken her bir adımda temel değişkenlerden birisi yerine temel olmayan bir değişken getirilir. Temele giren her bir değişken , fonksiyonu optimum sonuca biraz daha yaklaştırır.

Temele giren ve temeli terkeden değişkenlerin seçiminde şu kurallar uygulanır.
-Amaç satırındaki (ilk evrede f_c , ikinci evrede f_m satırı) negatif değerli katsayılardan mutlak değerce en büyük olan, temele girecek değişken olarak seçilir. Bu değişkene ait sütuna anahtar sütun denir. Böylece amaç fonksiyonu değerinde mümkün olan en büyük artış sağlanmış olacaktır. Amaç fonksiyonu satırında negatif değerli katsayı kalmadığında optimal çözüm bulunmuş olacaktır.
- Temeli terkedecek değişken seçiminde, sağ taraf değerleri (temel değişken değerleri) temele girecek değişken katsayılarına (yani anahtar sütun elemanlarına) sırasıyla bölünerek bir takım oran değerleri elde edilir. Negatif değerli olanlar bir tarafa bırakılırsa, elde edilen oran değerleri içinde en küçük olana karşılık gelen temel değişken, temelden ayrılacak değişken olarak seçilir. Bu değişkenin ait olduğu satıra anahtar satır, anahtar satır ile anahtar sütunun kesiştiği noktadaki katsayıya da anahtar sayı denir. Anahtar satır ve anahtar sütun belirlendikten sonra yapılacak iş, bir sonraki tablo değerlerini hesaplamaktır. Bir katsayının bir sonraki tablodaki değeri;

$A^*=A-(BC/D)$ formülü ile hesaplanır. Burada A^*, A katsayısının bir sonraki tablodaki değeri, D anahtar sayı, B ve C de A nın hizasındaki anahtar satır ve anahtar sütun elemanlarıdır.

Örnek 2.1.

Amaç fonksiyonu ;

Minimize $f = 2x_1 + x_2$

Kısıtlayıcılar;

$$2x_1 - x_2 < 15$$
$$2x_1 + 3x_2 = 50 \text{ ve}$$

Pozitiflik şartı;

x_1 , x_2 , x_3 > 0 olan doğrusal programlama problemini iki evreli simpleks metodla çözünüz.

- Amaç fonksiyonu minimizasyon türünde ise (-1) ile çarpılarak maksimizasyon türüne çevrilir. O halde amaç fonksiyonu;

Maksimize $f = -2x_1 - x_2$ olur.

- Kısıtlayıcılara yapay ve boş değişkenler eklenirse;

Maksimize $f = -2x_1 - x_2 - z_1 - z_2$

$$2x_1 - x_2 + y_1 = 15$$
$$2x_1 + 3x_2 + z_1 = 50$$
$$x_1 + x_2 - y_2 + z_2 = 10 \quad \text{olur.}$$

- Amaç fonksiyonu $f_c + f_m$ şeklinde ikiye ayrılarak ;

$f_c + 2x_1 + x_2 = 0$

$f_m + z_1 + z_2 = 0$ eşitlikleri oluşturulur.

- Temel değişkenler; TD, temel değişken değerleri; TDD, anahtar sütun değerlerinin temel değişken değerlerine bölümü; oran olmak üzere modelin katsayıları Tablo 2.3'te ki kuruluş tablosuna yerleştirilir.

Tablo 2.3 Örnek 2.1 'e ait kuruluş tablosu

TD	TDD	Oran	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
f_c	0		2	1	0	1	0	0	0	0
f_m	0		0	0	0	0	1	0	1	1
y_1	15		2	-1	0	0	0	1	0	0
z_1	50		2	3	0	0	0	0	1	0
z_2	10		1	1	-1	0	0	0	0	1

- Görüldüğü gibi katsayılar matrisi kanonik değildir. Kanonik hale getirmek için f_m satırından önce z_1, sonrada z_2 satırları çıkarılır. Bu işlemler yapıldıktan sonra başlangıç tablosuna (Tablo 2.4) geçilir.

Tablo 2.4 Örnek 2.1'e ait başlangıç tablosu

TD	TDD	Oran	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
f_c	0	-	2	1	0	1	0	0	0	0
f_m	-60	15	-3	-4	1	0	1	0	0	0
y_1	15	-	2	-1	0	0	0	1	0	0
z_1	50	50/3	2	3	0	0	0	0	1	0
z_2	10	10	1	1	-1	0	0	0	0	1

- Başlangıç tablosunda önce f_m amaç fonksiyonu ele alınarak optimize edilmeye çalışılır. Bunun için;

1. f_m satırındaki en negatif katsayılı elemanın bulunduğu sütun, anahtar sütun olarak seçilir. Örnek problemde x_2 sütunu anahtar sütundur.
2. TDD'ler x_2 sütun değerlerine bölünerek oran değerleri elde edilir. Bu oran değerlerinden en küçük oran değerinin ait olduğu satır(z_2) anahtar satır olarak belirlenir. Bir sonraki tabloda x_2 'nin z_2 'ye ait satırdan temele girdiği görülür.

Tablo 2.5 Örnek 2.1'e ait birinci iterasyon tablosu

TD	TDD	Oran	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
f_c	-10	-	1	0	1	1	0	0	0	0
f_m	-20	-	1	0	-3	0	1	0	0	0
y_1	25	-	3	0	-1	0	0	1	0	0
z_1	20	20/3	-1	0	3	0	0	0	1	0
x_2	10	-	1	1	-1	0	0	0	0	1

Tablo 2.5 için temele girecek değişken y_2 temelden ayrılacak değişken z_1 olarak belirlenir. Simpleks metod kuralları uygulanarak bir sonraki tablo değerleri hesaplanır.

Tablo 2.6 Örnek 2.2'ye ait ikinci iterasyon tablosu

TD	TDD	Oran	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
f_c	-50/3	-	4/3	0	0	1	0	0	0	0
f_m	0	-	0	0	0	0	1	0	0	0
y_1	95/3	-	8/3	0	0	0	0	1	0	0
y_2	22/3	-	-1/3	0	1	0	0	0	1	0
x_2	50/3	-	2/3	1	0	0	0	0	0	1

Tablo 2.6'dan görüldüğü gibi f_m satırında negatif katsayılı terim kalmadığından optimum çözüme ulaşılmıştır. f_m optimize edildikten sonra f_m satırı ve yapay değişkenlere ait sütunlar silinerek f_c amaç fonksiyonu ele alınır. f_c satırında da negatif katsayılı terim kalmadığından optimum çözüme ulaşılmıştır.

$$\min f_c = -(\text{Max } f_c) = -(-50/3) = 50/3 = 16.67$$

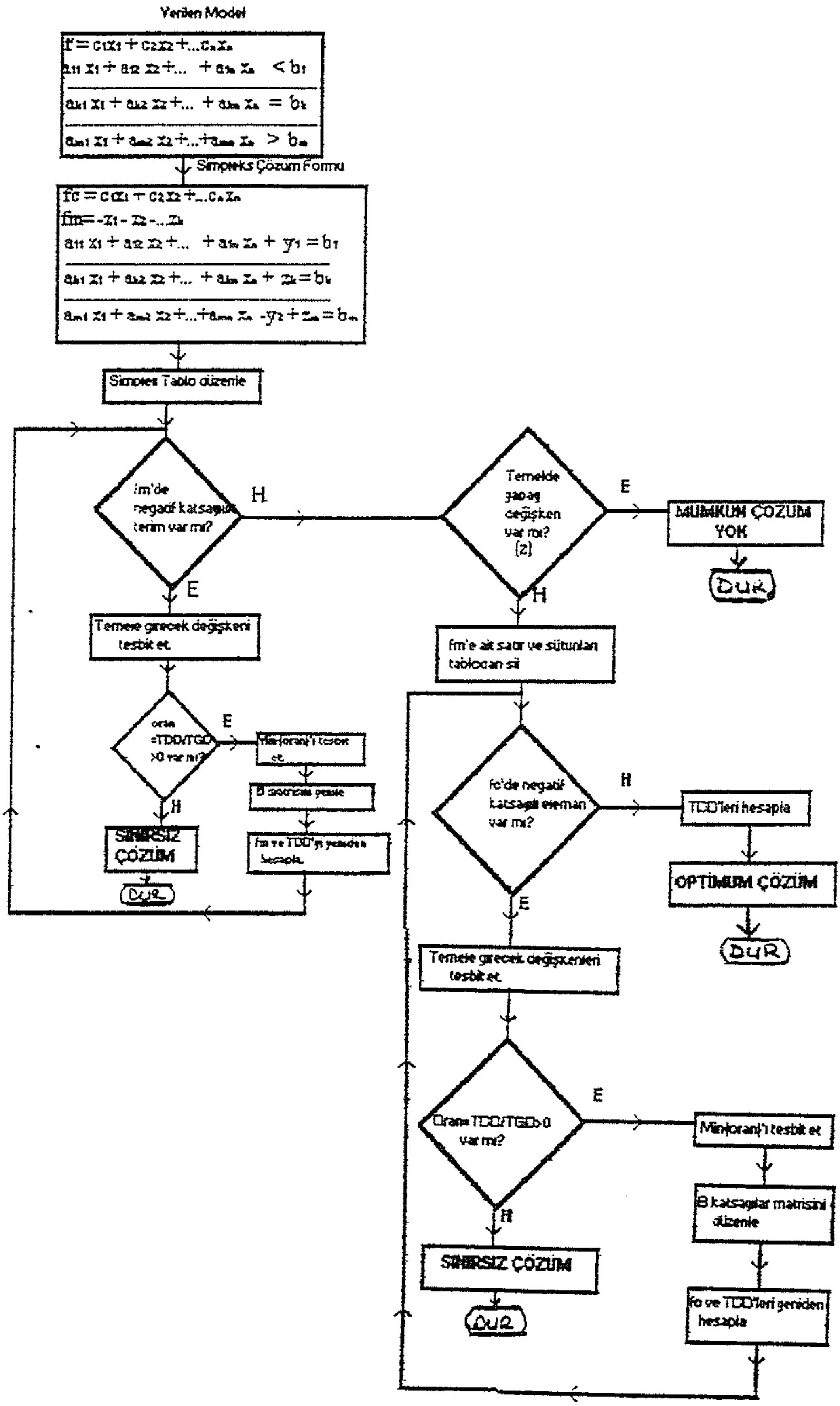
$$y_1 = 95/3 = 31.67$$

$$y_2 = 20/3 = 6.67$$

$x_2 = 50/3 = 16.67$ bulunur.

2.3.2.c. Revised Simpleks Metod

Bir doğrusal programlama problemini Simpleks metodla çözerken her iterasyonda (yani bir tablodan diğerine geçişte) tablonun gövdesi tümü ile işlem görür. Küçük boyutlu problemlerde bu husus fazla bir sakınca doğurmaz. Fakat doğrusal programlama problemleri genelde büyük boyutlu olduğundan ve bilgisayarda çözüm gerektirdiğinden bu metod oldukça büyük bellek gerektirir. Bu sebeple bilgisayarların gelişmesine paralel olarak yeni bir metod arayışına gidilmiş ve 1953 yılında G.B. Dantzig ve Orchard-Hays tarafından Revised Simpleks Metod (Düzeltilmiş Simpleks Metod) geliştirilmiştir. Bu metod aynı bilim adamları tarafından geliştirilen "Product Form of the Inverse" metodunun geliştirilmiş şeklidir (5). Revised Simpleks Metod genel olarak simpleks çözüm metodunu kullanır. Ancak simpleks metod gibi her iterasyonda yeni bir tablo oluşturmaz. Orjinal tablodan o iterasyonda işlem görecek bazı satır ve sütunları oluşturur. Bu suretle hem hafıza tasarrufu hem de zaman tasarrufu sağlar. Bugün çoğu bilgisayar programlarında Revised Simpleks Metodun tercih edilme sebebi de budur. Çözüm kurallarının uygulanabilmesi için amaç satırındaki, temele girecek değişken sütunundaki ve temel değişken değerleri sütunundaki değerlerin bilinmesi yeterlidir. Bu satır ve sütunlar temele girecek ve temelden ayrılacak değişkeni belirler.



Şekil 2.2 Revised Simpleks Metod Çözüm Algoritması

Simpleks metodda olduğu gibi modele gerekli yapay ve boş değişkenler eklenerek amaç fonksiyonu oluşturulur.

$$f - c_1 x_1 - c_2 x_2 - \ldots c_n x_n + z_1 + z_2 + \ldots z_n = 0 \qquad (2.11)$$

Amaç fonksiyonu f_c ve f_m şeklinde ikiye ayrılır. Böylece genel bir doğrusal programlama modeli (2.10) sistemindeki şekle gelmiş olur. Kuruluş tablosu bu sistemin katsayıları ile oluşturulur. f_m amaç fonksiyonundaki yapay değişkenler yok edilerek (2.12) elde edilir.

$$f_m - (a_{k+1,1} + a_{k+2,1} \ldots + a_{m1})x_1 - (a_{k+1,2} + a_{k+2,2} \ldots + a_{m2})x_2 \ldots (a_{k+1,n} + a_{k+2,n} + \ldots a_{mn})x_n = 0 \qquad (2.12)$$

Elde edilen yeni f_m satırı tabloya yerleştirilir. Oluşturulan bu tablo (Tablo 2.2) başlangıç tablosudur. Başlangıç tablosundaki temel değişken sütunlarının oluşturduğu matrise B^{-1} matrisi denir. Tablodan görüldüğü gibi ilk durumda B^{-1} matrisi birim matristir. Bu adımlardan sonra genel bir doğrusal programlama modelinin (2.6) revised simpleks metodla çözülmesi için şu işlemler yapılır: (Burada revised simpleks metod iki evreli simpleks metod kurallarını esas almaktadır.)

- f_m amaç satırı katsayıları B^{-1} matrisinin ikinci satırı (f_m satırı) ile başlangıç tablosundaki temele girecek değişken sütunUnlarının çarpımından elde edilir. Temele girecek değişken bu katsayılardan belirlenir. (Mutlak değerce en büyük negatif katsayılı terim temele girecek değişkeni belirler.) Temele girecek değişken kalmadığında f_m optimize edilmiş demektir. f_m optimize edildiğinde temelde yapay değişken yoksa B^{-1} matrisindeki f_m'e ait satır ve sütunlar silinerek f_c amaç fonksiyonunun optimizasyonuna başlanır. f_c satırında negatif katsayılı terim (temele girecek değişken) kalmayınca modelin optimum çözümü elde edilmiş olur.

- Temele girecek değişken sütunu , B^{-1} matrisi ile başlangıç tablosundaki o değişkene ait sütunun çarpılmasından elde edilir. Aynı şekilde temel değişken değerleri, B^{-1} matrisi ile başlangıç tablosundaki temel değişken değerleri sütununun çarpımından elde edilir. Temel değişken değerlerinin temele girecek değişken değerlerine bölünmesinden elde edilen en küçük pozitif oran değerinden temelden ayrılacak değişken belirlenir.

- Temele girecek değişken sütununda anahtar sayı bir, diğer değerler sıfır olacak şekilde B^{-1} matrisi üzerinde işlemler yapılır. Bu şekilde bir değişken temele girmiş olur. Aynı işlemler bir sonraki iterasyon için tekrar edilir.

Örnek 2.2.

Amaç fonksiyonu ;

Minimize $f = 2x_1 + x_2$

Kısıtlayıcılar;

$$x_1 + x_2 > 10$$

$$2x_1 - x_2 < 15$$

$$2x_1 + 3x_2 = 50 \text{ ve}$$

Pozitiflik şartı;

$x_1, x_2, x_3 > 0$ olan doğrusal programlama problemini revised simpleks metodla çözünüz.

- Amaç fonksiyonu minimizasyon türünde olduğundan (-1) ile çarpılarak maksimizasyon türüne çevrilir.

Maksimize $f = -2x_1 - x_2$ olur.

- Kısıtlayıcılara yapay ve boş değişkenler eklenerek model;

$$
\begin{aligned}
&\text{Maximize } f = -2x_1 - x_2 - z_1 - z_1 \\
&2x_1 - x_2 + y_1 = 15 \\
&2x_1 + 3x_2 + z_1 = 50 \\
&x_1 + x_2 - y_2 + z_2 = 10
\end{aligned}
$$

haline getirilir.

- Amaç fonksiyonu $f_c + f_m$ şeklinde ikiye ayrılarak ;

$$f_c + 2x_1 + x_2 = 0$$

$$f_m + z_1 + z_2 = 0$$

eşitlikleri oluşturulur. Tüm bu değerler tabloya yerleştirilir. Tablo 2.7'den görüleceği gibi ilk durumda B^{-1} matrisi kanonik değildir.

Tablo 2.7 Örnek 2.2'ye ait kuruluş tablosu

					B^{-1} matrisi				
TD	TDD	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
f_c	0	2	1	0	1	0	0	0	0
f_m	0	0	0	0	0	1	0	1	1
y_1	15	2	-1	0	0	0	1	0	0
z_1	50	2	3	0	0	0	0	1	0
z_2	10	1	1	-1	0	0	0	0	1

B^{-1} matrisinin kanonik hale gelmesi için z_1 ve z_2 satırları f_m satırından çıkarılır.

Tablo 2.8 Örnek 2.2'ye ait başlangıç tablosu

iterasyon	TD	TDD	x_1	x_2	y_2	f_c	f_m	y_1	z_1	z_2
	f_c	0	2	1	0	1	0	0	0	0
	f_m	-60	-3	-4	1	0	1	0	0	0
	y_1	15	2	-1	0	0	0	1	0	0
	z_1	50	2	3	0	0	0	0	1	0
	z_2	10	1	1	-1	0	0	0	0	1
1	f_m	-60	-3	-4	1					
2	f_m	40	4	4	-4					
3	f_m	200/3	8/3	4	0					

- Temele girecek değişken ve temelden ayrılacak değişken tesbit edilir. Başlangıç tablosundan görüleceği gibi temele girecek değişken x_2 temelden ayrılacak değişken z_2 olarak belirlenir.

Tablo 2.9 Örnek 2.2'ye ait birinci iterasyon tablosu

iterasyon	TD	B^{-1}					yeni TD	TDD	Oran
1	f_c	1	0	0	0	0	1 x_2	0	-
	f_m	0	1	0	0	0	-4	-60	15
	y_1	0	0	1	0	0	-1	15	-
	z_1	0	0	0	1	0	3	50	50/3
	z_2	0	0	0	0	1	1	10	10(ek)

- x_2 sütununun z_2 satırındaki(anahtar satır) değeri 1, diğer eleman değerleri 0 olacak şekilde matris işlemleri B^{-1} üzerinde uygulanır. Bu şekilde yeni temel değişken(x_2) sütunu kanonik hale getirilmiş ve amaç fonksiyonunda yerini almış olur.
- Bir sonraki iterasyonda aynı işlemleri tekrarlamak üzere B^{-1} matrisinin f_m satırı ile tablonun gövdesi (temele girecek değişken sütunları) çarpılır. Buradan f_m amaç fonksiyonu katsayıları bulunur. Bu işlemler f_m satırında negatif katsayılı terim kalmayıncaya kadar tekrar edilir. f_m optimize edildikten sonra f_m'e ait satır ve sütun B^{-1} matrisinden silinir. Artık amaç f_c fonksiyonunu optimize etmektir. Elde edilen yeni B^{-1} matrisinden yararlanılarak f_m için yapılan işlemler f_c satırında negatif katsayılı terim kalmayıncaya kadar tekrar edilir.

Tablo 2.8'de ikinci iterasyondaki f_m katsayılarından temele girecek değişken y_2 ve temelden ayrılacak değişken z_1 olarak tesbit edilir. y_2 sütununu kanonik hale getirmek için gerekli matris işlemleri yapılarak bir sonraki B^{-1} matrisi elde edilir.

Tablo 2.10 Örnek 2.2'ye ait ikinci iterasyon tablosu

İterasyon	TD	B^{-1}					yeni TD	TDD	Oran
2	f_c	1	0	0	0	-1	1 y_2	-10	-
	f_m	0	1	0	0	4	-4	40	-
	y_1	0	0	1	0	1	-1	25	-
	z_1	0	0	0	1	-3	3	20	20/3
	x_2	0	0	0	0	1	-1	10	-

Tablo 2.11 Örnek 2.2'ye B^{-1} final tablosu

iterasyon	TDD	B^{-1}				
3	f_c	1	0	0	-1/3	0
	f_m	0	1	0	4/3	0
	y_1	0	0	1	1/3	0
	y_2	0	0	0	1/3	-1
	x_2	0	0	0	1/3	0

Bu B^{-1} matrisinin f_m satırı ile başlangıç tablosunun gövdesi(temele girecek değişken sütunları) çarpılarak üçüncü iterasyon için f_m değerleri hesaplandığında, f_m satırında negatif katsayılı terim kalmadığı, dolayısıyla optimal sonuca ulaşıldığı görülür. (Tablo 2.8) B^{-1} matrsinden f_m 'e ait satır ve sütunlar silinerek yeni B^{-1} matrisi elde edilir. (Tablo 2.12)

Tablo 2.12 Örnek 2.2'ye ait B^{-1} sonuç tablosu

iterasyon	TDD	B^{-1}			
3	f_c	1	0	-1/3	0
	y_1	0	1	1/3	0
	y_2	0	0	1/3	-1
	x_2	0	0	1/3	0

f_c satırı ile tablonun gövdesi (Tablo 2.8'deki temele girecek değişken sütunları) çarpılarak f_c satırının değerleri hesaplanır (4/3, 0, 0). f_c satırında da negatif katsayılı eleman olmadığından optimal çözüme ulaşılmıştır denir.

B^{-1} matrisi ile temel değişken değerleri sütunu çarpılarak optimum değerler hesaplanır. Burdan;

f_c= -50/3= -16.67

y_1 = 95/3=31.67

y_2 = 20/3=6.67

x_2= 50/3 =16.67 bulunur.

f_c amaç fonksiyonu maksimize olduğundan (-1) ile çarpılarak minimum değeri bulunur. Yani f_c = 16.67 bulunur.

2.4. Simpleks Metod Uygulamalarında Karşılaşılan Özel Durumlar:

Simpleks Metod uygulamalarında karşılaşılan dört özel durum şunlardır:

- Dejenerasyon (Degenerecy)
- Alternatif optimum çözüm (Alternative optima)
- Sınırsız Çözüm (Unbounded Solution)
- Mümkün çözümün olmaması (No feasable Solution)

2.4.1. Dejenerasyon

Genel bir doğrusal programlama modelinde çözüm, optimum noktaya ulaşılıncaya kadar bir temel uygun çözüm noktasından daha optimum bir temel uygun çözüm noktasına doğru hareket edilerek bulunur. Bazı durumlarda ise optimum çözüm noktasına ulaşılmayabilir. Bu duruma dejenerasyon denilir.
Simpleks metodun iki önemli kuralı vardır. Bunlardan ilki anahtar sütunun yani temele girecek değişkenin belirlenmesi kuralıdır. Bu kural uygulanırken amaç satırında birden fazla mutlak değeri en büyük negatif terim varsa bunlardan herhangi biri seçilebilir ve bu durum bir dejenerasyona yol açmaz. İkinci kural

ise anahtar satırın yani temelden ayrılacak değişkenin belirlenmesi kuralıdır. Bu kural uygulanırken en küçük oran değeri birden fazla satırda ortaya çıkmışsa bunlardan herhangi birinin seçimi bir dejenerasyona yol açabilir. Bu dejenerasyon iki şekilde görülür.

- Simpleks tabloda iki veya daha fazla en küçük oran değeri sıfır olabilir. Bu iki veya daha fazla temel değişken değerinin sıfır olması demektir. Pratik olarak mümkün olmadığı halde, teorik örnekler verilebilir. Bu problemlerde temelden ayrılacak değişkenin rastgele seçimi çözümü kısır döngüye sokar. Problem belli sayıda iterasyondan sonra tekrar başa döner , optimum çözüme varılamaz. Pratikte hiçte karşılaşılmayan bu sorunların çözümü için Dantzig(6), Charnes(7), Wolfe(8) ve Dantzig, Orden, ve Wolfe(9) tarafından çeşitli teknikler geliştirilmiştir. Kısır döngü olarak tanımlanan bu duruma hiçbir uygulamalı problemde rastlanmadığı için lineer programlama ile uğraşan bir çok bilgisayarcı bu tekniklere programlarında yer vermemiştir. Buna rağmen bu tekniklerin önemi simpleks metodu kusursuz yapmalarıdır (10).

Burada Charnesin önerdiği "Perturbation Method" üzerinde durulacaktır.

Bu metoda göre değerleri sıfır olan temel değişken değerlerine sıfıra çok yakın bir sayı olan e ve çeşitli kuvvetleri aşağıdaki sıraya göre eklenir. Böylece temel değişken değerleri sıfırdan farklı bir değer almış olur.

$B1 + e$

$B2 + e^2$

....

$Bm + e^m$ olur. e çok küçük bir sayı olduğundan bu sayının kuvvetleri büyüdükçe değerleri küçülecektir. Yani $e > e^2 > ... > e^m$'dir. Yeni temel değişken değerlerinin anahtar sütuna karşılıklı olarak bölünmesi ile elde edilen sonuçlar incelenerek en küçük pozitif değeri veren satır anahtar satır olarak kabul edilir.

Örnek 2.3.

Amaç fonksiyonu;

Maksimize $f = 0.75x_1 - .20x_2 + 0.5x_3 - 6x_4$

Kısıtlayıcılar; $0.25x_1 - 8x_2 - x_3 + 9x_4 < 0$

$0.5x_1 - 12x_2 - 0.5x_3 + 3x_4 < 0$

$x_3 < 1$

ve pozitiflik şartı x_1 , x_2 , x_3 , $x_4 > 0$ şeklindeki bir maksimizasyon problemi ele alınacaktır.

Amaç fonksiyonundan temele girecek değişken olarak x_1 seçilir. x_1 için oran değerleri;

TD	TDD	x_1	oran
y_1	0	1/4	0
y_2	0	1/2	0
y_3	1	0	∞

olduğu görülür. Bu durumda herhangi bir kural uygulanmadan rastgele seçim yapılırsa problem kısır döngü'ye girer.

Buna karşı anahtar satırı seçmek için perturbation metodu uygulanırsa ;

TD	TDD	x_1	oran(TDD/x 1)
y_1	$0 + e$	1/4	$4e$
y_2	$0 + e^2$	1/2	$2e^2$
y_3	$0 + e^3$	0	∞

$2e^2 < 4e$ olduğundan en küçük oran değeri veren satır y_2 anahtar satır olarak belirlenir. Bu şekilde simpleks çözüm işlemlerine devam edilirse optimum çözüme ulaşılır.

- Temel değişken değerlerinin anahtar sütunu değerlerine karşılıklı olarak bölünmesi sonucunda en küçük pozitif değeri sağlayan sıfırdan farklı birden fazla satırın varlığı halinde, anahtar satırın rastgele seçilmesi de dejenerasyona sebep olur. Bu dejenerasyon iki şekilde görülür. Ya sonlu iterasyonlardan sonra problem optimum sonuca ulaşır veya kısır döngü hali ortaya çıkar. Sonlu iterasyondan maksat kısır döngü olmamasıdır. Yoksa bu durumda da problem gereksiz yere uzatılmış ve bir bozulma olmuştur. Bu dejenerasyonun önlenmesi için de çeşitli teknikler geliştirilmiştir.

Programa ilk alınan boş veya yapay değişkenlerden indisi en küçük olanın anahtar satır olarak belirlenmesi bu metodlardan birisidir. Diğer bir metod ise rastgele sayı tablosunun kullanılmasıdır. Genel olarak kullanılan metod ise eşit oranlı satırlarda birim matristen itibaren her eleman bulunduğu satırın anahtar sütunu üzerindeki elemana bölünerek çeşitli değerler elde edilir. Elde edilen değerler soldan sağa doğru gidilerek mukayese edilir. Eşit olmayan ilk mukayesede, eşit olmayan değerlerin hangisi en küçükse bu değere ait satır anahtar satır olarak kabul edilir.

Örnek 2.4.

Amaç fonksiyonu ;

Maksimize $f = 3x_1 + 9x_2$

Kısıtlayıcılar;

$$x_1 + 4x_2 < 8$$

$$x_1 + 2x_2 < 4$$

ve pozitiflik şartı;

$x_1, x_2 > 0$ olan bir maksimizasyon problemi ele alınacaktır.

Amaç fonksiyonunda en büyük katsayı x_2 'ye ait olduğu için temele girecek değişkenin x_2 olduğu kolayca görülecektir. Buna karşılık oran değerleri;

TD	TDD	x_2	oran
y_1	8	4	2
y_2	4	2	2

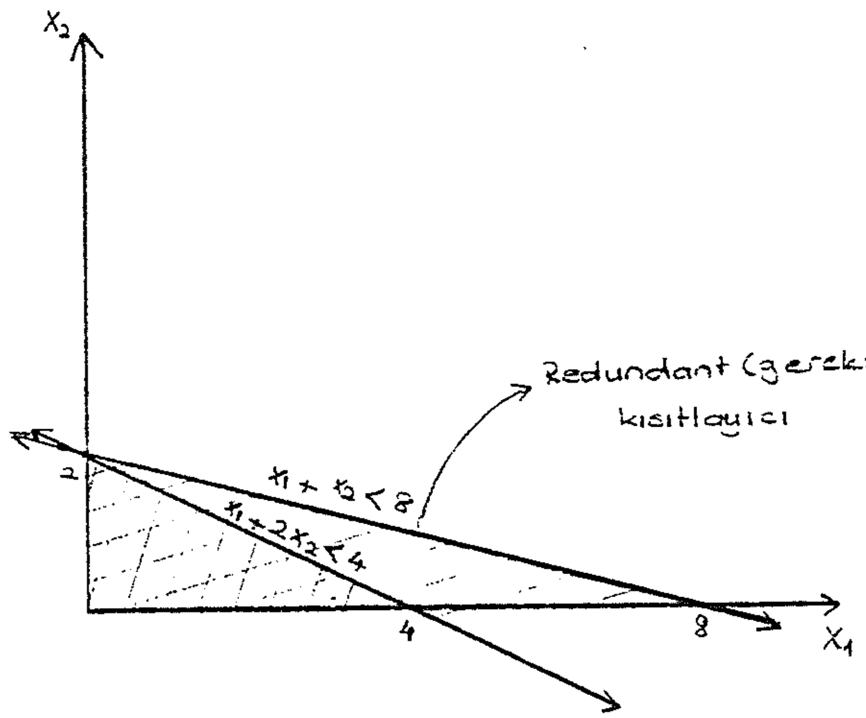
bulunur.

Birim matris ise ;

	y_1	y_2	oran
y_1	1	0	1/4
y_2	0	1	0/2

şeklindedir. Birim matrisin ilk elemanlarından elde edilen oranlar 1/4 ve 0 dır. Görüldüğü gibi eşitlik ilk anda bozulmuştur. En küçük değer y_2 ye ait değer olduğundan anahtar satır yani temelden ayrılacak değişken olarak y_2 tesbit edilir. Bu şekilde simpleks çözüme devam edilirse optimum sonuca ulaşılır.

Pratik olarak dejenerasyon doğrusal programlama probleminin en az bir gereksiz(redundant) kısıtlayıcı bulundurduğu durumlarda ortaya çıkar. Kısır döngü hali oluşturmasalar bile gereksiz kısıtlayıcılar problemin çözümünü uzatarak optimum çözüme daha geç ulaşılmasına neden olurlar. Bu sebeple bir doğrusal programlama modeli kurarken bu gibi gereksiz kısıtlayıcıların oluşturulmamasına dikkat edilmelidir. Yukarıdaki problem grafik üzerinde gösterilirse gereksiz kısıtlayıcı daha açık bir şekilde görülecektir. (Şekil 2.3)



Şekil 2.3. Dejenere doğrusal programlama modeli

2.4.2. Alternatif Optimum Çözüm

Amaç fonksiyonunun eğimi kısıtlayıcılardan herhangi birinin eğimi ile aynı olursa amaç fonksiyonu birden çok noktada optimal çözümü sağlayacaktır. Bu durum alternatif optimum çözüm olarak adlandırılır.

Örnek 2.5.

Amaç fonksiyonu;

$$f = 2\ x_1 + 4x_2$$

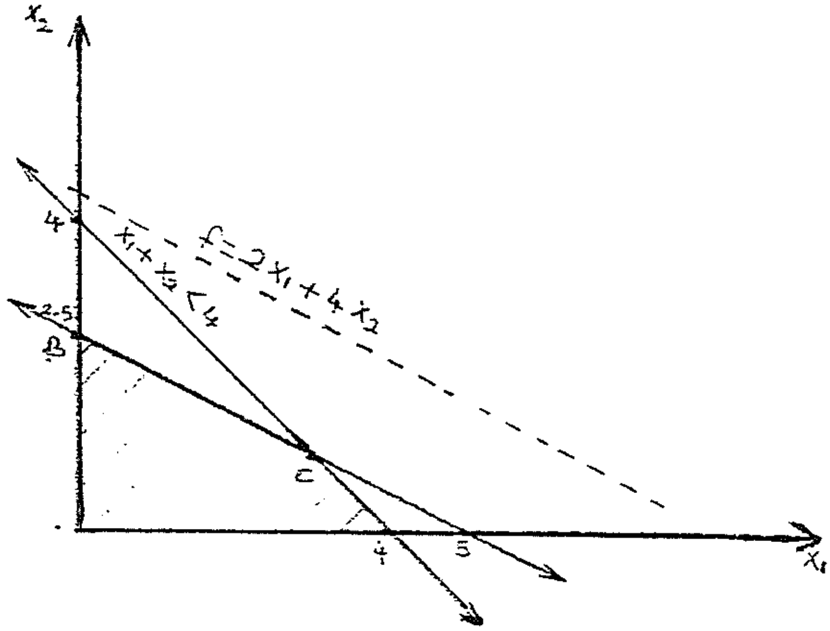
Kısıtlayıcılar;

(1) $x_1 + x_2 < 4$

(2) $x_1 + 2x_2 < 5$

ve pozitiflik şartı;

x_1 , x_2> 0 olan maksimizasyon problemi ele alınacaktır.



Şekil 2.4. Alternatif Optimum Çözüm

Model grafik üzerinde gösterilecek olursa amaç fonksiyonunun ikinci kısıtlayıcıya paralel olduğu görülür.(Şekil 2.4) BC doğru parçası boyunca her x_1 , x_2 değişkeni için amaç fonksiyonu değeri maksimum ve 10 dur. Ancak cebirsel olarak simpleks metod sadece köşe noktalarını çözüm olarak seçtiği için optimum çözüm noktaları B ve C kabul edilecektir.

Pratikte alternatif optimum çözüm, işletme yönetimine kendi kısıtlarına uyan birden fazla seçenek sunduğu için yararlıdır.

2.4.3. Sınırsız Çözüm

Bazı doğrusal programlama problemlerinde değişken sayısı kısıtlayıcıların durumuna bağlı olarak artabilir. Temele girecek değişken varken temelden ayrılacak bir değişken (pozitif katsayılı en küçük oran değeri) mevcut olmaz. Temelden ayrılacak bir değişken olmadığı için de yeni temel değişkenin temele hangi değerle gireceği tesbit edilemez. Bu durumda yeni temel değişkenin alacağı sınırsız değer olduğundan, problemin de sınırsız çözümü vardır denilir.

Örnek 2.6.

Amaç fonksiyonu;

Maksimize $f = x_1 + 2x_2$

Kısıtlayıcı şartlar;

(1) $x_1 - x_2 < 10$

(2) $2x_1 < 40$

ve pozitiflik şartı;

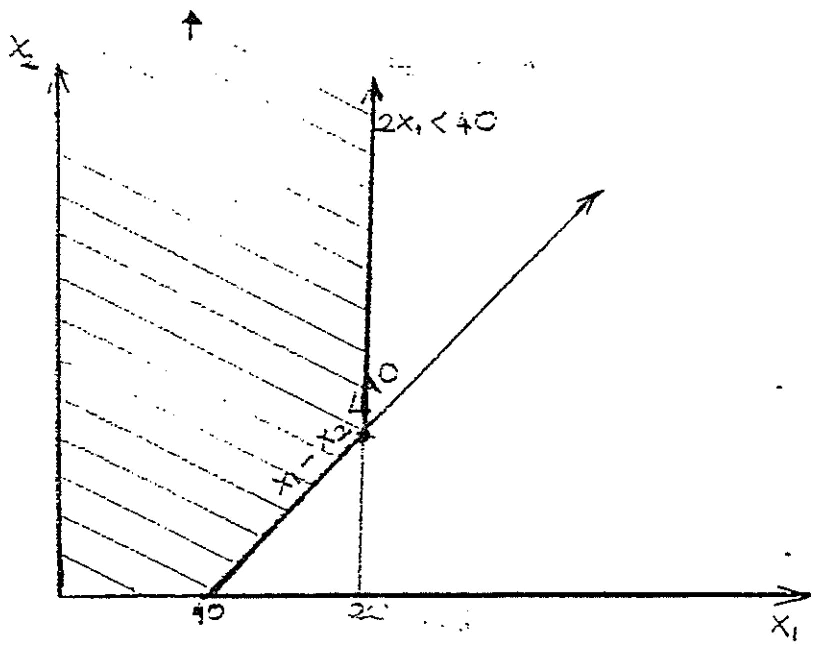
x_1 , $x_2 > 0$ olan maksimizasyon problemi ele alınacaktır.

Temele girecek değişkenin x_2 olduğu amaç fonksiyonundan kolayca görülmektedir. Temelden ayrılacak değişkenin tesbiti için oran değerleri hesaplanırsa;

TD	TDD	x_2	oran
y_1	10	-1	-10
y_2	40	0	∞

bulunur. Görüldüğü gibi pozitif katsayılı oran mevcut değildir. Yani temelden ayrılacak değişken yoktur.

Şekil 2.5'den de görüleceği gibi kısıtlayıcılar tarafından sınırlanan bir çözüm alanı ve uygun çözüm noktaları mevcut değildir. Dolayısıyla sınırsız çözüm vardır.



Şekil 2.5. Sınırsız Çözüm

Pratik olarak problemin aslında doğrusal bir problem olmadığı sonucuna varılır. Çünkü doğrusal programlama modelleri kısıtlayıcı şartlara sahiptir.

2.4.4. Mümkün Çözüm Olmaması

Bir doğrusal programlama modelinde kısıtlayıcı şartlar uygun bir konveks çözüm alanı oluşturmuyorsa modelin çözümü yoktur. Bu durum tüm kısıtlayıcıların küçük-eşit olması durumunda görülmez. Çünkü boş değişkenler her zaman mümkün bir çözüm temin eder.

Mümkün çözümün olmaması optimal sonuca ulaşmış bir problemde herhangi bir yapay değişkenin sıfırdan farklı bir değerle temelde kalması durumunda görülür.

Pratikte mümkün çözümün olmaması durumu kısıtlayıcılar birbirleri ile çeliştiğinden, modelin düzgün bir şekilde formüle edilmediğini gösterir.

Önek 2.7.

Amaç fonksiyonu ;

Maksimize $f = x_1 + 5x_2$

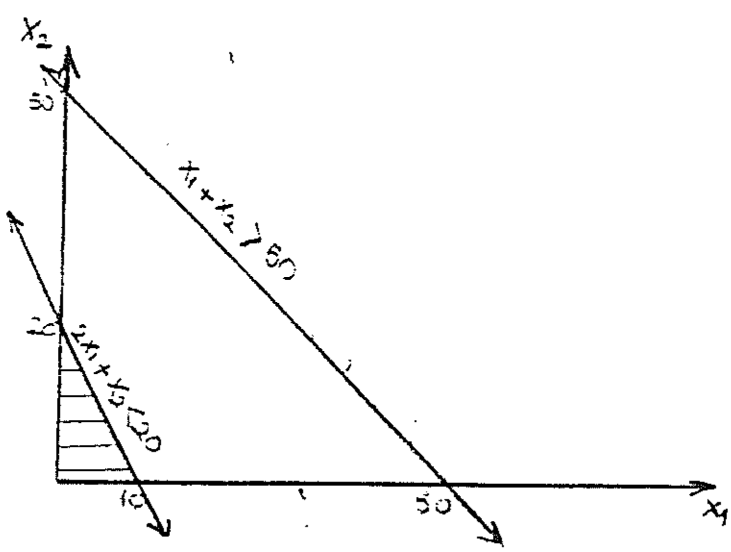
Kısıtlayıcılar;

(1) $x_1 + x_2 > 50$

(2) $2x_1 + x_2 < 20$

ve pozitiflik şartı;

$x_1, x_2 > 0$ olan problem grafik çözümle ele alınacaktır.



Şekil 2.6. Mümkün Çözümün Olmaması

Şekil 2.6'dan da görüldüğü gibi ortak çözüm bölgesi yoktur.

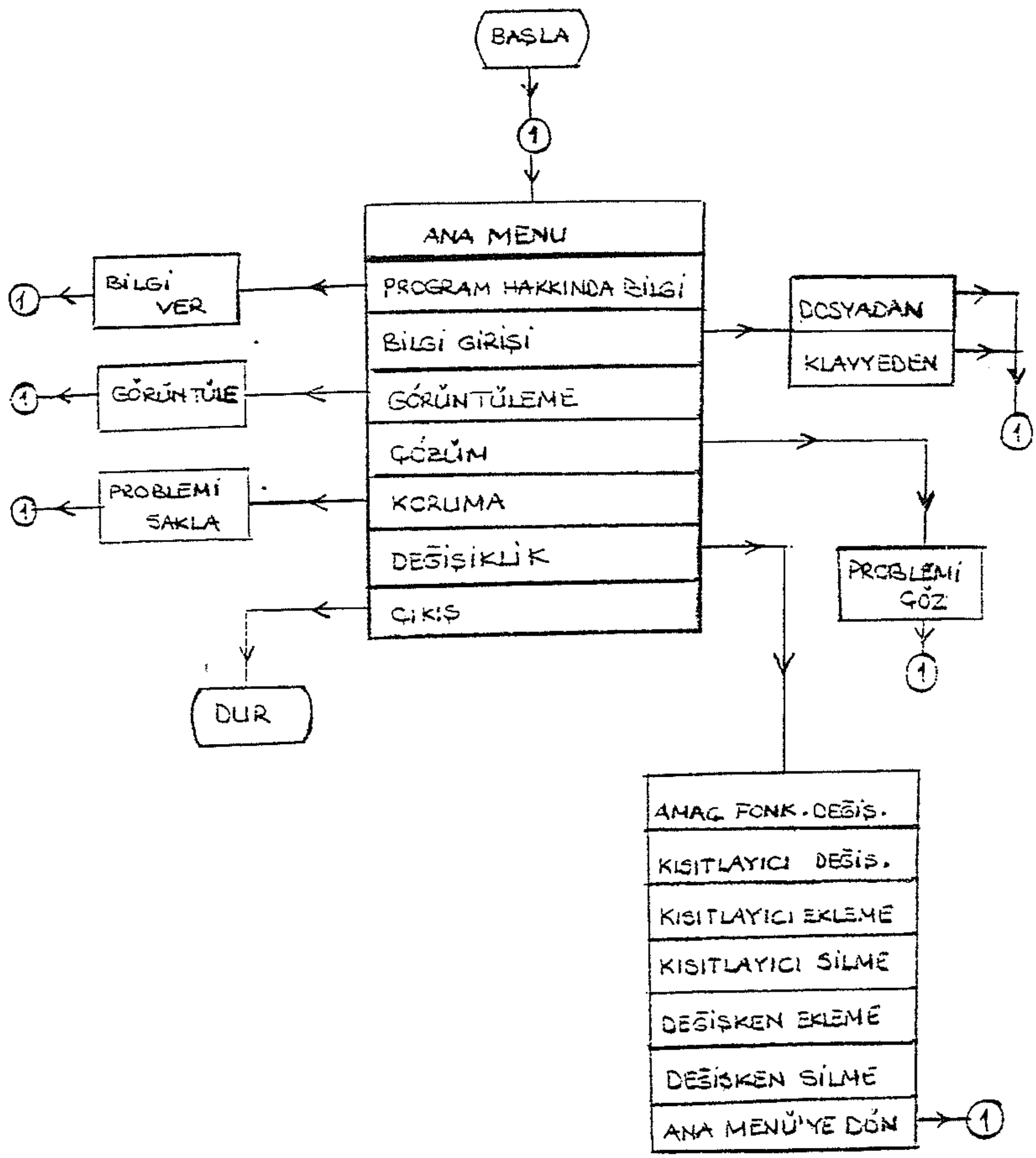
3. BİLGİSAYAR PROGRAMI GELİŞTİRME UYGULAMASI (ATADP)

3.1. Giriş

Bu bölümde QBASIC programlama dili kullanılarak geliştirilen ATADP programı incelenecektir. ATADP adı verilen program Revised Simpleks çözüm metodunu kullanarak en çok 100x100 boyutlu doğrusal programlama modellerini çözebilir. ATADP programının kodlaması yapılırken yapısal olmasına özen gösterilmiş, takibinin kolay olması için açıklama satırlarına yer verilmiştir. Tüm etiket isimleri sayısaldır.

3.2. Programın Şematik Görünümü

ATADP programının şematik görünümü şekil 3.1' de verilmiştir. ATADP programı çalıştırıldığında ekrana bir menü gelir. Bu ana menüde yedi seçenek vardır. Bunlardan PROGRAM HAKKINDA BİLGİ seçildiğinde bir doğrusal programlama modelinin bu programa nasıl aktarılacağı hakkında bilgi verilir. Bu seçeneğin modülü olan Bilgi.Bas ek 6' da verilmiştir. BİLGİ GİRİŞİ seçildiğinde, DOSYADAN ve KLAVYEDEN seçeneklerini bulunduran bir alt menu gelir. Bunlardan DOSYADAN seçeneği daha önce disk veya diskette koruma altına alınmış olan bir doğrusal programlama modelini okuyarak hafızaya getirir. Bu seçeneğin blok şeması ek 1.1' de verilmiştir. KLAVYEDEN seçeneği ise amac fonksiyonu ve kısıtlayıcıları bilinen bir modelin bilgisayara aktarılmasına yarar. Bu seçeneğin blok şeması ek 1.2' de verilmiştir. GÖRÜNTÜLEME seçeneği o anda hafızada bulunan modelin amaç fonksiyonu ve kısıtlayıcılarını görüntüler. Blok şeması ek 2' de verilmiştir. ÇÖZÜM seçeneği ise eğer varsa modelin optimal çözümünü bulur. Çözümü yapılan modelin optimal değerleri tablolar halinde verilmektedir. Tabloda iterasyon sayısı, amaç fonksiyonunun optimal değeri, temele girmiş değişkenlere ait optimal değerler ve temele girmemiş değişkenler için fırsat maliyeti değerleri verilmektedir. Bu seçeneğin akış şeması ek 3' de verilmiştir. KORUMA seçeneği ise o an hafızada bulunan modeli disk veya diskette saklamaya yarar. Bu seçeneğin blok şeması ek 4' de verilmiştir. DEĞİŞİKLİK seçildiğinde ise yedi alt seçeneği olan bir alt menu gelir. Bu alt menude bir model üzerinde yapılabilecek tüm değişiklik seçenekleri bulunur. Bunlardan ilki amaç fonksiyonu katsayılarını değiştirmeye yarar. Blok şeması ek 5.1' de verilmiştir. İkinci seçenek istenen kısıtlayıcının katsayılarını değiştirmeye yarar. Blok şeması ek 5.2' de verilmiştir. Bu seçeneklerden üçüncüsü modele yeni bir kısıtlayıcı eklemeye yarar. Blok şeması ek 5.3' de verilmiştir. Dördüncü seçenek modelden istenen kısıtlayıcıyı siler. Blok şeması ek 5.4' de verilmiştir. Beşinci seçenek modele yeni bir değişken eklemeye, altıncı seçenek istenen bir değişkeni modelden silmeye yarar. Blok şemaları sırasıyla ek 5.5 ve ek 5.6'da verilmiştir. Son seçenek ise ana menüye dönüşü sağlar.



Şekil 3.1. ATADP programının şematik görünümü

3.3. Programda Kullanılan Değişkenlerin Listesi

var: Değişken Sayısı

cons: Kısıtlayıcı sayısı

opt: Amaç değeri

TGD( ): Temele Giren Değişken

TDD( ): Temel Değişken Değeri

a(,): Amaç fonksiyonu ve kısıtlayıcı katsayılarını içeren matris

0. Satırda amaç fonksiyonu, diğer satırlarda kısıtlayıcı katsayıları mevcuttur.

aa$( ): Temele girecek değişken isimlerinin saklandığı matris

bb$( ): Temelden ayrılacak değişken isimlerinin saklandığı matris

b(,): Kuruluş tablosu içeriğinini taşıyan matris

Mat(,): B^{-1} birim matrisinin içeriğini taşıyan matris

km( ): Amaç fonksiyonu katsayılarının hesaplanarak taşındığı matris

oran( ): Temelden ayrılacak değişkeni belirleyen oran değerlerinin taşındığı matris (TDD( )/ TGD( ))

it: iterasyon sayısı

ek( ): En küçük oran değerinin satır indisini taşıyan matris

c( ): En büyük negatif teriminin indisini taşıyan matris

va: Büyük-eşit kısıtlayıcı sayısı

t: Küçük-eşit kısıtlayıcı sayısı

c: Büyük-eşit ve eşit kısıtlayıcı sayısı

esay( ): Eşit oran değerine sahip temel değişken değerlerinin indislerinin saklandığı matris

ek: En küçük değerlerin taşındığı değişken

3.4. Programın Yazıldığı Dil

Geliştirilen ATADP programı QBASIC programlama dili ile yazılmıştır. Yüksek seviyeli programlama dillerinden olan QBASIC, yapısal programlama ve kütüphaneleri kullanma imkanına sahiptir.

Yapısal programlama program tasarımı ve yazılımını kurallara bağlayan ve disiplin altına alan bir yaklaşımdır. Yapısal programlamada ana ilke GOTO deyimini kaldırmak ve temel denetim yapılarını kullanmaktır(11). Program tasarımı aşamasında problem her modülün kolaylıkla çözülebileceği düzeye kadar modüllere bölünür. Modüller arasında hiyerarşik bir bağ vardır.

QBASIC yapısal programlamanın kabul etmediği fakat bazı durumlarda kullanılması mecburi olan şartsız dallanma komutlarını da ihtiva eder.

QBASIC dilinin özellikleri kısaca şu başlıklar altında incelenebilir.

3.4.1. Bilgi Türleri

QBASIC'te sabitler tamsayı sabitler ve kesirli sabitler olarak ikiye ayrılır. Tamsayı sabitler kendi arasında kısa tamsayı ve uzun tamsayı, kesirli sabitler de tek duyarlıklı (single precision) ve çift duyarlıklı (double precision) olarak ikiye ayrılır. Tek duyarlıklı sayılar hafızada 4 byte, çift duyarlıklı sayılar ise 8 byte yer kaplar. Tek duyarlıklı sayılar hafızada en fazla 7 anlamlı rakam bulundururken çift duyarlıklı sayılar 15 anlamlı rakam bulundurur (12).

3.4.2. Sapma ve Kontrol Deyimleri

Yapısal programlama da kullanılmadığı halde QBASIC programlama dilinde sapma deyimleri vardır. Bunlar şartsız sapma deyimleri GOTO, GOSUB ile şartlı sapma deyimleri ON şart GOTO ve ON şart GOSUB deyimleridir.
QBASIC' te kullanlan kontrol deyimleri ise IF ve SELECT CASE komutlarıdır.
IF bir durum için kontrol yaparken, SELECT CASE bir çok durum için kontrol yapabilir.

3.4.3. Döngü Deyimleri

FOR / NEXT, DO / LOOP, WHILE / WEND komut çiftleri döngü deyimleridir. FOR / NEXT şartsız , diğer iki komut çifti ise şartlıdır. Bu iki komut çiftinde aranan şart sağlandığı sürece program döngü içinde kalacaktır.

3.4.4. Prosedürler

Yapısal programlama için önemli tanımlardan biridir. Fonksiyon ve alt program prosedürü olarak ikiye ayrılır. Fonksiyonda bir değer döndürülürken , altprogramda birden fazla değer döndürülür.

3.4.5. Modüler İmkanlar

Daha öncede belirtildiği gibi yapısal programlama bir problemi modüler olarak ele almaktadır. Program modüllerden oluşuyorsa bir modülden diğerine CHAIN deyimi ile geçilir. Bu imkan sayesinde her modül bağımsız bir program olabilir. Çalışan bir program RUN deyimi ile yeniden çalıştırılabilir.

3.4.6. SHELL İmkanı

SHELL deyimi ile DOS işletim sistemi komutları program içerisinde kullanılabilir.

3.4.7. Dosyalar

Manyetik disk veya disket gibi yardımcı hafıza birimlerinden bilgi alış verişi dosya özelliği sayesinde sağlanır.

3.5. Programın Çalışması

ATADP programı doğrusal programlama modellerini çözmek için hazırlanmıştır. Doğrusal programlama modelleri bir önceki bölümde anlatıldığı gibi simpleks ve revised simpleks çözüm metodları ile çözülebilir. Revised simpleks metod simpleks metoda göre daha az işlem gerektirir. Simpleks metod her iterasyonda yeni bir tablo oluştururken , revised simpleks metod sadece temele girecek ve temelden ayrılacak değişkenleri belirleyecek satır ve sütunları oluşturur (13). Teorik olarak bilinen bu üstünlüğün bir bilgisayar programı için kazandıracağı ise hız ve bellektir. Programın doğrusal programlama modellerini çözerken revised simpleks metod algoritmasını kullanmasının sebebi de teorik olarak bilinen bu üstünlüğü pratik olarak ortaya koymaktır. Programda iki evreli revised simpleks metod kullanılmıştır. Programın kodlaması ek 7' de verilmiştir.

Bir doğrusal programlama modelinin amaç, amaç fonksiyonu ve kısıtlayıcılardan oluştuğu bilinmektedir. ATADP' de cons(kısıtlayıcı sayısı), var(Değişken sayısı), opt(Amaç) olmak üzere üç önemli değişken vardır. Programda girilen yeni bir model iki boyutlu A matrisinde saklanmaktadır. A matrisinin ilk satırında (a(0,var)) amaç fonksiyon değerleri, diğer satırlarında ise kısıtlayıcı değerleri bulunmaktadır. Model A matrisine aktarılırken kısıtlayıcıların kısıt yönü(küçük-eşit, eşit, büyük-eşit) sayısal hale getirilir. Küçük kısıtlayıcılar sıfır, eşit kısıtlayıcılar bir ve büyük-eşit kısıtlayıcılar iki değeri ile A matrisine aktarılır. Model görüntülenirken de bu sayısal değerlerin karakter karşılıkları görüntülenir. Klavyeden girilen bir model daha sonra kullanılmak üzere disk veya diskette saklanabilir ve model üzerinde istenen değişiklikler yapılabilir.

Verilen bir modelin çözümünde program amaç fonksiyonunu maksimize kabul ederek işlem yapmaktadır. Çözülecek model minimize ise önce amaç fonksiyonu (-1) ile çarpılarak maksimize hale getirilir. Bir modelde küçük-eşit, eşit, büyük-eşit olmak üzere üç tür kısıtlayıcı mevcut olabilir. Kuruluş tablosu oluşturulabilmesi için verilen tüm eşitsizlikler eşitlik haline getirilek kanonik bir form oluşturulur. Küçük-eşit kısıtlayıcılara y_i boş değişkeni eklenerek, eşit kısıtlayıcılara z_i yapay değişkeni eklenerek, büyük-eşit kısıtlayıcılara ise $-t_i$ boş değişkeni ve z_i yapay değişkeni eklenerek kanonik hale getirilmiş olur. İlk durumda temelde bulunan değişkenler , bu yapay ve boş değişkenlerdir. Amaç ise temelde bulunan bu değişkenleri temelden atmak yerine modelin gerçek değişkenlerini temele almaktır. Bu yapıldığı takdirde yani temele girecek değişken kalmadığında ve tüm yapay değişkenler temelden atıldığı anda modelin optimum çözümü bulunmuş olur. Programda temele girecek değişken isimleri aa$, temelden ayrılacak değişken isimleri ise bb$ dizilerinde saklanmıştır.

Bu işlemler yapıldıktan sonra eldeki verilerle kuruluş tablosu oluşturulur. Kuruluş tablosunun temele girecek değişken sütunları içeriği B matrisinde taşınmaktadır. B matrisinin ilk iki satırında f_c ve f_m amaç fonksiyonlarının katsayıları, diğer satırlarda ise kısıtlayıcı değerleri vardır. B matrisinin ilk sütununda ise kısıt değerleri (Temel değişken değerleri) bulunmaktadır.

Büyük-eşit kısıtlayıcılara eklenen t_i boş değişkenleri de temele girecek değişkenlerdir.

Mat matrisi kuruluş tablosunun birim matrisi B^{-1} 'in içeriğini taşımaktadır. f_m optimize edilirken bu matrisin birinci satırı , f_c optimize edilirken de sıfırıncı satırı km dizisinin değerlerini belirlemede kullanılmaktadır. Mat matrisi satırları sırasıyla f_c , f_m ,1. kısıtlayıcı,2. kısıtlayıcı ve cons. kısıtlayıcı temel değişkenlerinin f_c , f_m, 1. kısıtlayıcı ,2.kısıtlayıcı, cons.kısıtlayıcıdaki katsayılarını içerir.

Buraya kadar anlatılanlar modeli çözülmeye hazır hale getirmiştir. Bundan sonra yapılacak işlemler optimum sonuca ulaşıncaya kadar bir mümkün çözümden diğerine geçmektir. Bu sebeple f_m satırındaki mutlak değerce en büyük katsayılı negatif terim aranır. Bu terim temele girecek değişkeni belirler. f_m satırı değerleri km dizisi içerisinde saklanmaktadır. Km dizisi mat matrisindeki optimize edilmeye çalışılan amaç fonksiyonu satırı (Birinci evre için f_m satırı) ile B matrisinin çarpımından elde edilir. Eğer km dizisinde negatif katsayılı eleman kalmamışsa optimize edilmiş olur. Bu durumda modelde yapay değişken varsa modelin mümkün bir çözümü yoktur. Temeldeki tüm yapay değişkenler temelden atılmış veya 0 katsayı ile temelde bulunuyorsa f_m'ye ait satır ve sütunlar mat matrisinden ve B matrisinden silinir. Aynı işlemler f_c için gerçekleştirilir. Ancak f_c için km dizisi hesaplanırken mat matrisinin sıfırıncı satırı(f_c satırı) esas alınır. Km katsayıları negatif fakat sıfıra çok yakın değerler alıyor olabilir. Programda bu durum göz önünde bulundurulmuş ve katsayı (-1.10^{-5})'den küçükse sıfır kabul edilmiştir.

Revised simpleks metodun tercih edilme sebebi her iterasyon için B matrisini hesaplamak yerine km dizisi (Temele girecek değişkenleri belirlemeye yarar.), anahtar sütun değerleri (TGD dizisi) ve temel değişken değerleri(TDD dizisi) dizilerini hesaplamasıdır. Böylece m kısıtlayıcı ve n değişkene sahip olan bir modelde her bir iterasyonda m*n+m değer yerine 3*m tane değer hesaplanır. Temelden ayrılacak değişkeni tesbit için de temel değişken değerleri temele girecek değişken değerlerine bölünür. (TDD/TGD) Elde edilen en küçük pozitif katsayılı terimin bulunduğu satır temelden ayrılacak değişkeni belirler. Pozitif katsayılı bir oran değeri mevcut değilse modelin sınırsız çözümü vardır.

Mat matrisinde temelden ayrılan yapay değişkenin yerini temele gireceği belirlenen değişken alır. Bu sütun kanonik hale getirilerek bir değişkenin amaç fonksiyonunda hangi değerle bulunacağı belirlenmiş olur. Bu işlemler modelin optimal çözümü bulununcaya kadar tekrar edilir. Optimal çözümü bulunan modelin sonuç değerleri bir tablo halinde verilmektedir. Aşağıda 15 değişkenli, 10 kısıtlayıcılı bir doğrusal programlama modelinin ATADP'deki sonuç tablosu (Tablo 3.1) verilmiştir.

Tablo 3.1. Bir doğrusal programlama modelinin ATADP sonuç tablosu

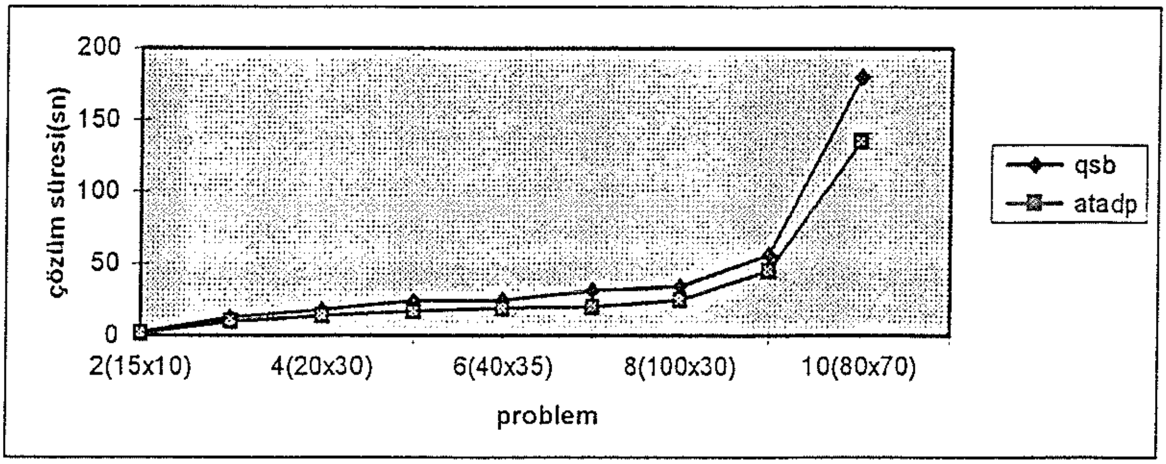
SONUC TABLO	iterasyon: 6		fmin= 18.10717		Sayfa: 1
Degisken	cözüm	firs. mal.	Degisken	cözüm	firs. mal.
1 x 1	30	0	16 t 1	0	.21
2 x 2	15	0	17 t 2	0	.02
3 x 3	30	0	18 t 3	0	.08
4 x 4	2	0	19 t 4	0	.06
5 x 5	0	.7	20 t 5	0	.1594134
6 x 6	0	1	21 y 1	30.7	0
7 x 7	0	.001	22 z 1	0	-.21
8 x 8	0	3.999293E-02	23 z 2	0	-.02
9 x 9	33.46969	0	24 z 3	0	-.08
10 x 10	0	.8	25 y 2	248	0
11 x 11	0	.26	26 z 4	0	-.06
12 x 12	0	.38	27 y 3	60	0
13 x 13	0	.44	28 z 5	0	
14 x 14	63.76534	0	29 y 4	60	0
15 x 15	0	.078264	30 z 6	0	-.1594134

3.6. Çözüm Süresi

Bir doğrusal programlama modelinin çözüm süresindeki artış yaklaşık olarak kısıtlayıcı sayısındaki artışın kübü ile doğru orantılıdır (Kaçtıoğlu,1984). ATADP benzer işlevli QSB programı ile karşılaştırıldığında aynı modeli ATADP'nin daha kısa sürede çözdüğü görülmüştür. Çünkü ATADP revised simpleks metod algoritması kullanırken, diğer program simpleks metod algoritması kullanmaktadır. Çözüm süresi kısıtlayıcı sayısı ve değişken sayısına bağlı olarak ta artmaktadır. Şekil 3.2' ye ait farklı boyutlardaki modellerin ATADP ve QSB' deki çözüm süreleri tablo 3.2' de verilmiştir.

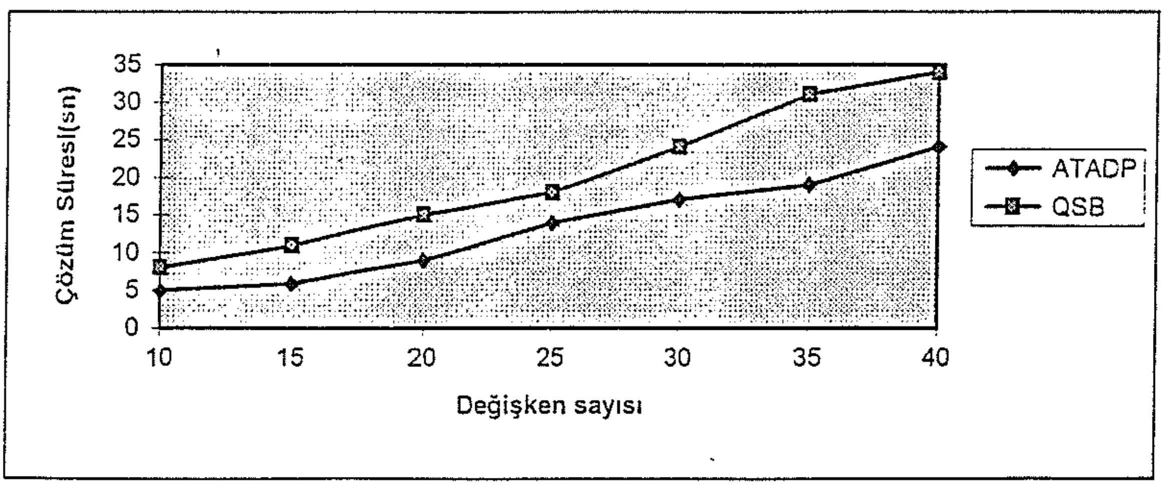
Tablo 3.2. Farklı boyuttaki modellerin ATADP ve QSB'deki çözüm süreleri

problem	Çözüm Süresi(sn)		atadp/qsb
	QSB	ATADP	
1(15x10)	3	2	1,5
2(90x13)	13	10	1,3
3(20x30)	18	14	1,285714
4(30x40)	24	17	1,411765
5(40x35)	25	19	1,315789
6(35x40)	31	20	1,55
7(100x30)	34	25	1,36
8(40x51)	56	45	1,244444
9(80x70)	180	135	1,333333

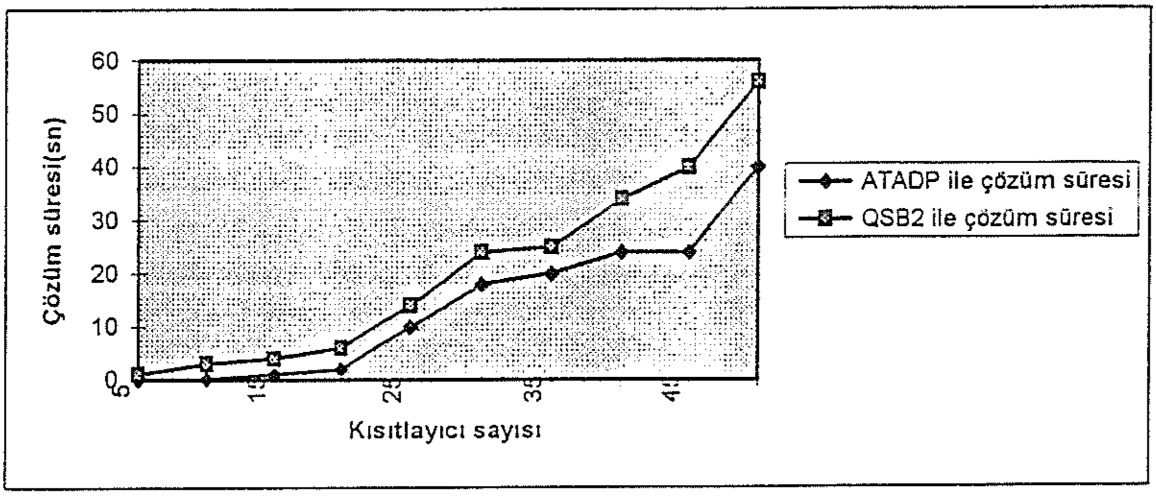


Şekil 3.2. Farklı boyuttaki modellerin ATADP ve QSB 'deki çözüm süreleri

Şekil 3.2 'de farklı kısıtlayıcı sayıları ve değişken sayılarına sahip modellerin ATADP ve diğer programdaki çözüm süreleri, Şekil 3.3' de kısıtlayıcı sayısı sabit iken(40) değişken sayısı artışına karşı çözüm süreleri, Şekil 3.4' de ise değişken sayısı sabit iken(40) kısıtlayıcı sayısı artışına karşı çözüm süreleri grafik olarak verilmiştir.



Şekil 3.3. Değişken sayısı artışına karşı çözüm süreleri



Şekil 3.4. Kısıtlayıcı sayısı artışına karşı çözüm süreleri

4. SONUÇ

Bu çalışmada yöneylem araştırması optimizasyon tekniklerinden olan doğrusal programlama incelenmiş, çözüm teknikleri ele alınmış ve bu model için söz konusu olan bazı özel durumlar incelenmiştir. Program geliştirme bölümünde ise QBASIC programlama dili kullanılarak doğrusal programlama modellerini Revised Simpleks algoritmayla çözen bir program geliştirilmiştir. Bu algoritmanın tercih edilme sebebi ise daha az işlem ve daha az bellek kapasitesi gerektirmesidir. Geliştirilen program (ATADP) programı kişisel bilgisayarlarda yaygın olarak kullanılan benzer amaçlı QSB programı ile kıyaslandığında şu sonuçlara varılmıştır.

- ATADP QSB'den en az %40 daha hızlıdır. Program teorik olarak bekleneni vermiştir. Kısıtlayıcı sayısı arttıkça ATADP programının %40'tan daha hızlı olacağı söylenebilir. Küçük boyutlu modellerde revised simpleks metodu işlem sayısı simpleks metod işlem sayısına yakın olmasına rağmen boyut büyüdükçe (kısıtlayıcı sayısı ve değişken sayısı arttıkça) Revised Simpleks Metod daha iyi sonuçlar verecektir(14).

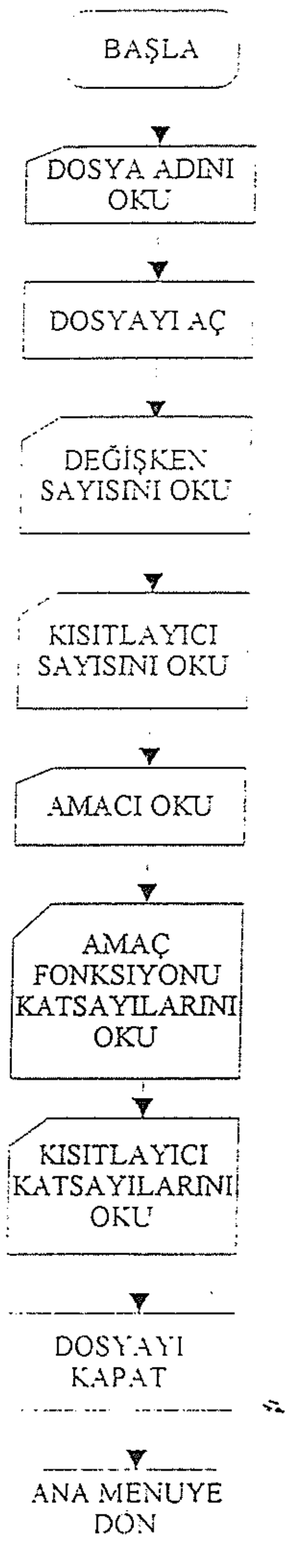
- QSB'de dejenerasyon durumları göz önüne alınmamıştır. Dejenerasyon oluşacak bir problem girildiğinde ise program sonsuz döngüye girmektedir. ATADP'de ise dejenerasyonu önleyen iki altprogram bulunmaktadır. Böylece dejenere bir problemin sebep olacağı sonsuz döngü veya fazla iterasyonlar önlenmiştir.

- Hem ATADP'de hem de QSB'de çözülen modellerde kısıtlayıcı sayısı artışının değişken sayısı artışına göre çözüm süresini daha çok artırdığı görülmüştür.

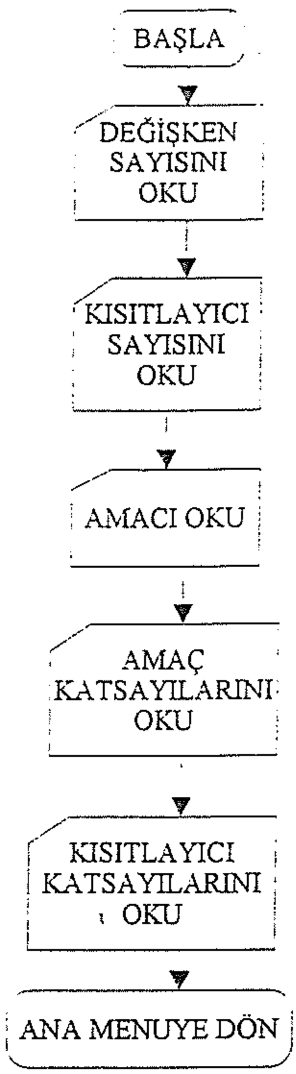
5. EKLER

Ek 1. Bilgi Girişi Blok Şemaları

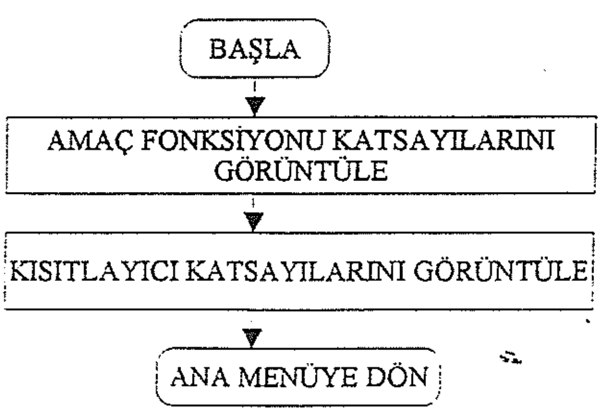
Ek 1.1. Dosyadan Bilgi Girişi Blok Şeması



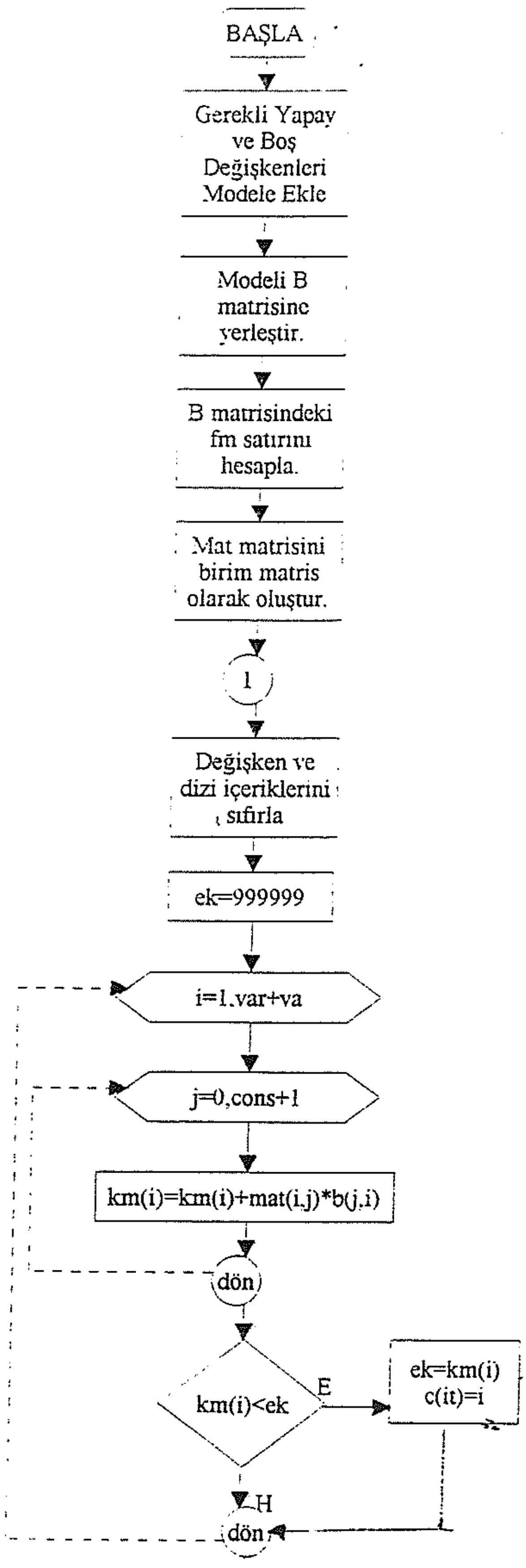
Ek 1.2. Klavyeden Bilgi Girişi Blok Şeması

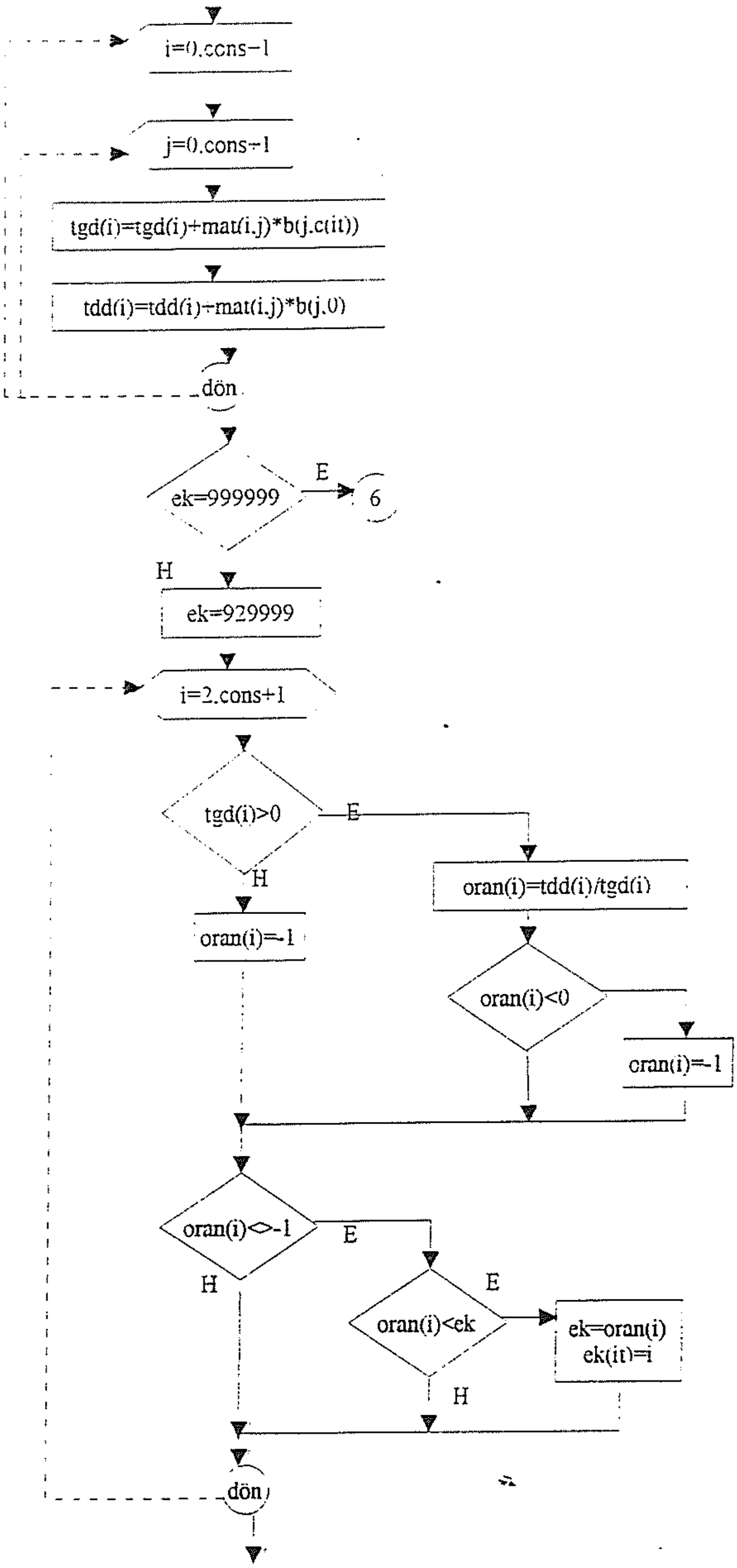


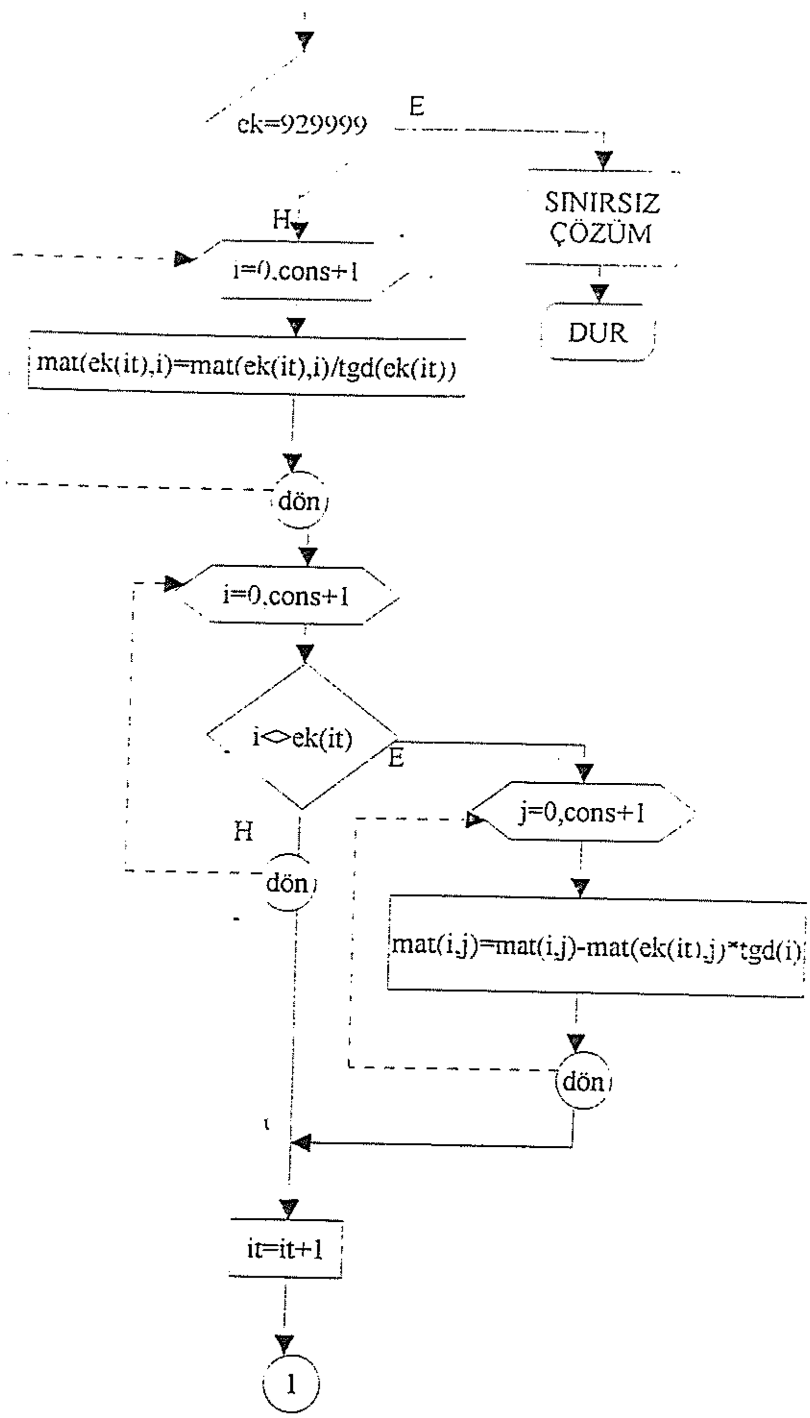
Ek 2. Görüntüleme Blok Şeması

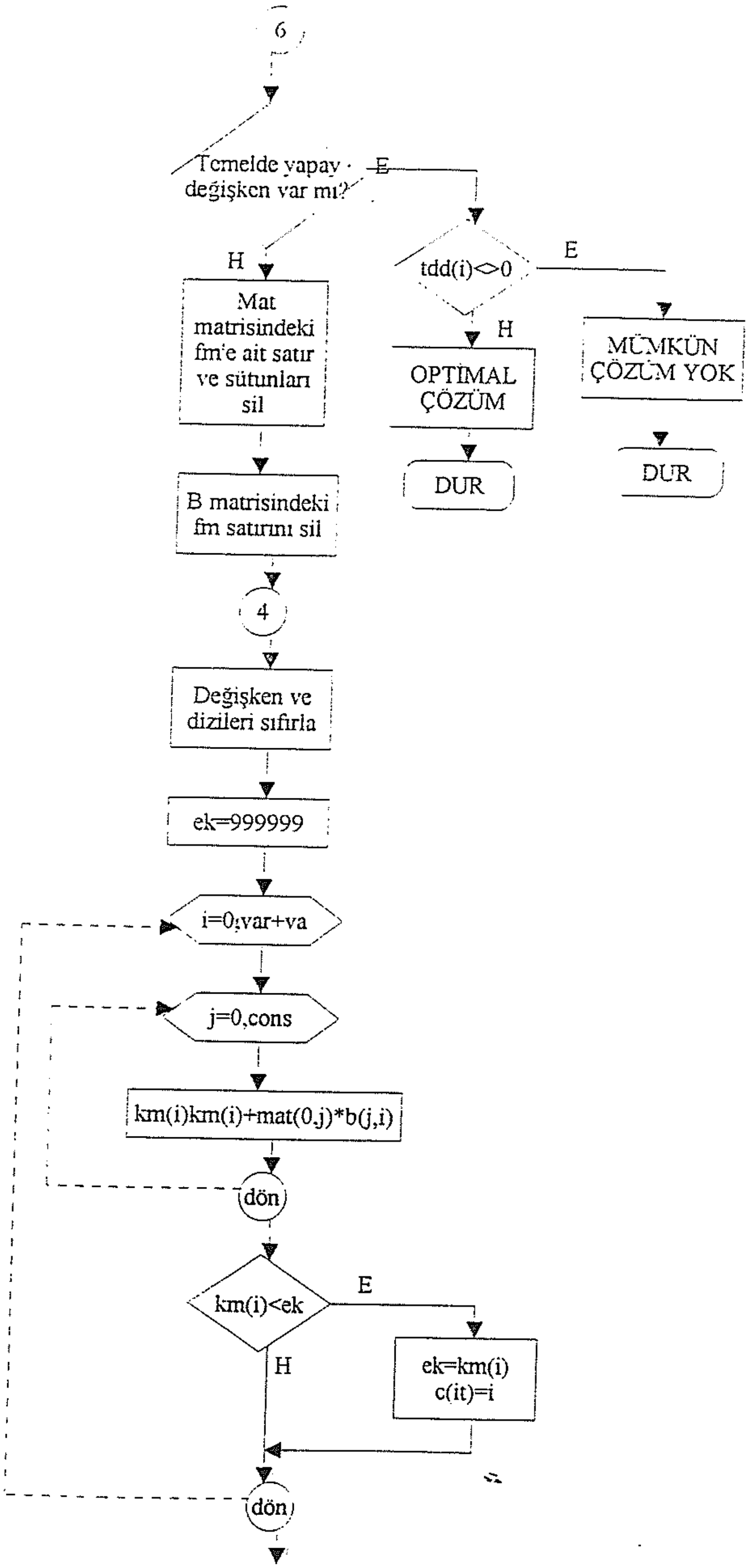


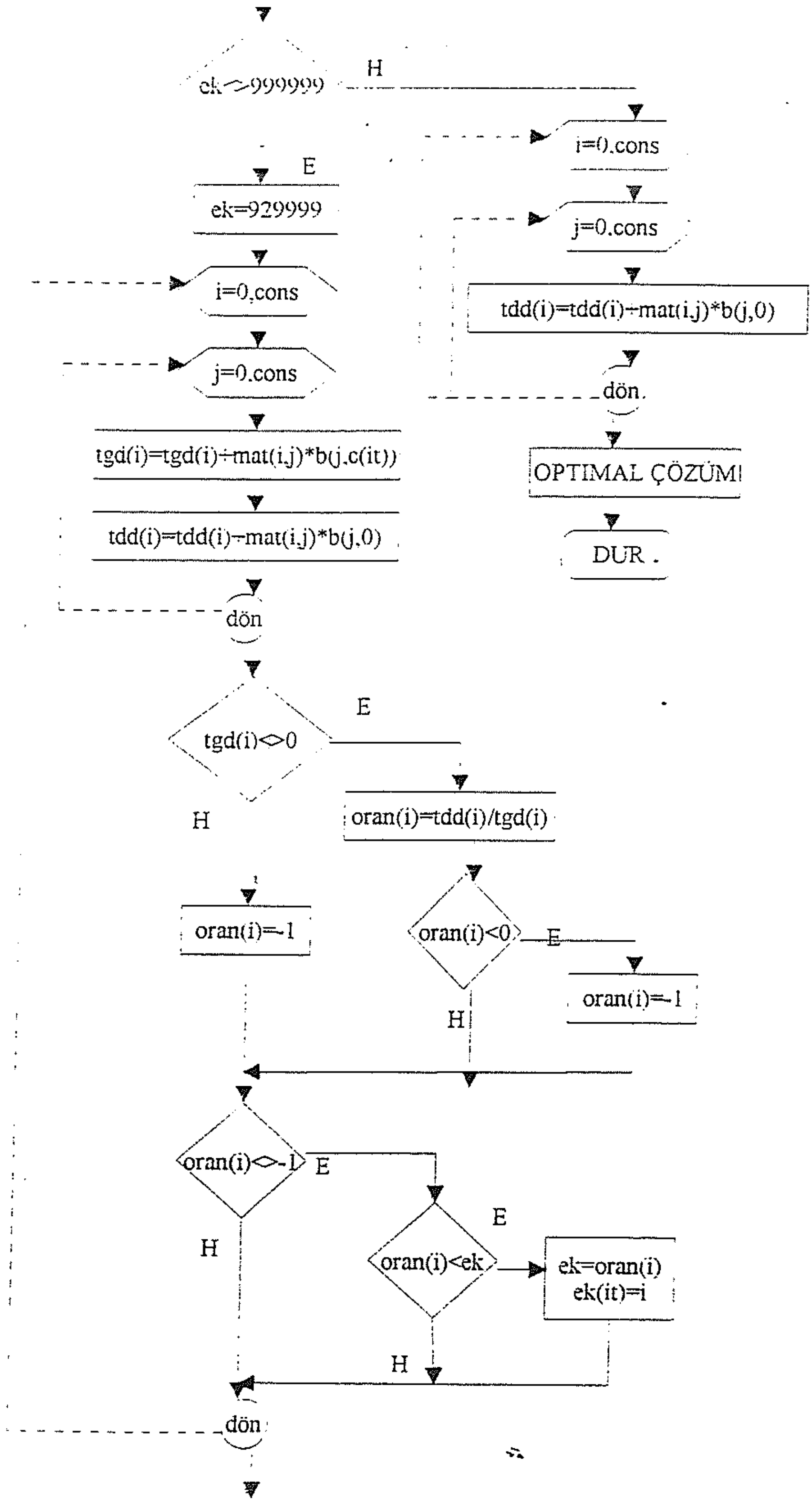
Ek 3. Çözüm Akış Şeması

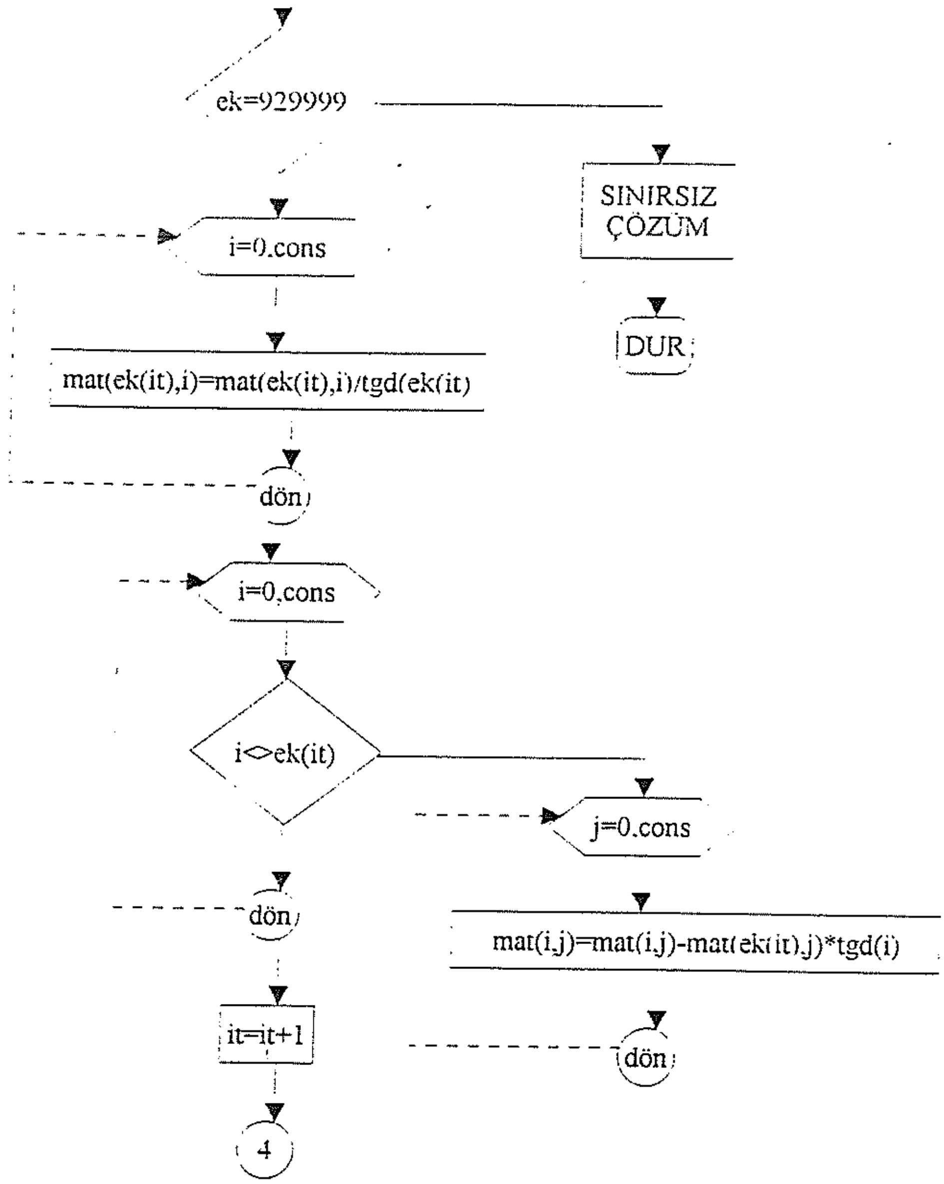



i=0.cons-1
j=0.cons-1
tgd(i)=tgd(i)+mat(i.j)*b(j.c(it))
tdd(i)=tdd(i)+mat(i.j)*b(j.0)
dön
ek=999999
E
6
H
ek=929999
i=2.cons+1
tgd(i)>0
E
H
oran(i)=tdd(i)/tgd(i)
oran(i)=-1
oran(i)<0
oran(i)=-1
oran(i)<>-1
E
H
oran(i)<ek
E
ek=oran(i)
ek(it)=i
H
dön

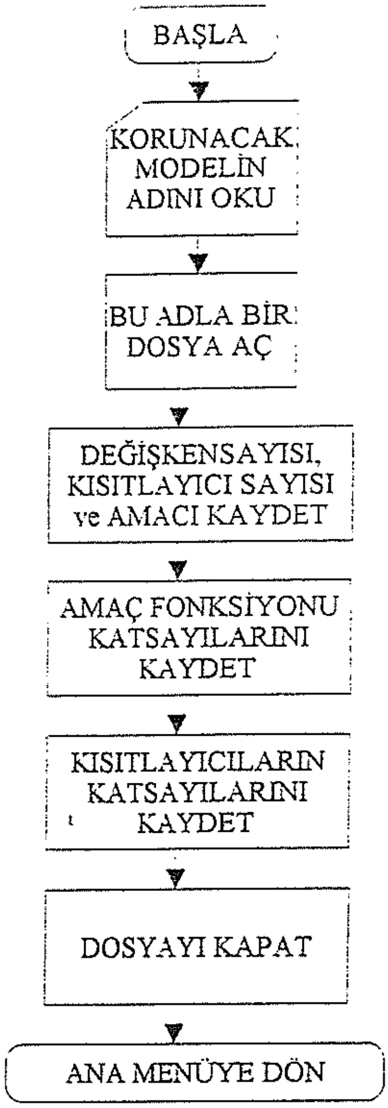

ek=929999
E
SINIRSIZ ÇÖZÜM
DUR
H
i=0,cons+1
mat(ek(it),i)=mat(ek(it),i)/tgd(ek(it))
dön
i=0,cons+1
i<>ek(it)
E
H
j=0,cons+1
dön
mat(i,j)=mat(i,j)-mat(ek(it),j)*tgd(i)
dön
it=it+1
1


6
Temelde yapay değişken var mı?
E
H
Mat matrisindeki fm'e ait satır ve sütunları sil
B matrisindeki fm satırını sil
4
Değişken ve dizileri sıfırla
ek=999999
i=0,var+va
j=0,cons
km(i)km(i)+mat(0,j)*b(j,i)
dön
km(i)<ek
E
H
ek=km(i)
c(it)=i
dön
tdd(i)<>0
E
H
OPTİMAL ÇÖZÜM
DUR
MÜMKÜN ÇÖZÜM YOK
DUR


ek<>999999
H
E
ek=929999
i=0.cons
j=0.cons
tgd(i)=tgd(i)+mat(i,j)*b(j,c(it))
tdd(i)=tdd(i)+mat(i,j)*b(j,0)
dön
i=0.cons
j=0.cons
tdd(i)=tdd(i)+mat(i,j)*b(j,0)
dön
OPTIMAL ÇÖZÜM!
DUR
tgd(i)<>0
E
H
oran(i)=tdd(i)/tgd(i)
oran(i)=-1
oran(i)<0
E
oran(i)=-1
H
oran(i)<>-1
E
H
E
oran(i)<ek
ek=oran(i)
ek(it)=i
H
dön


ek=929999
SINIRSIZ ÇÖZÜM
DUR
i=0.cons
mat(ek(it),i)=mat(ek(it),i)/tgd(ek(it)
dön
i=0.cons
i<>ek(it)
j=0.cons
mat(i,j)=mat(i,j)-mat(ek(it),j)*tgd(i)
dön
dön
it=it+1
4

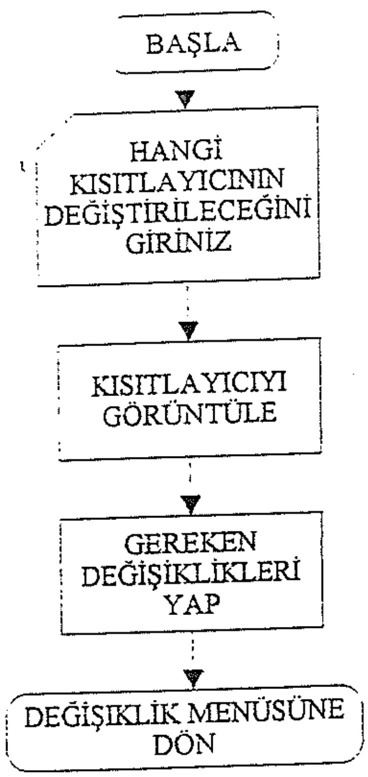
Ek 4. Koruma Blok Şeması



Ek 5. Değişiklik Alt Menüsü Blok Şemaları

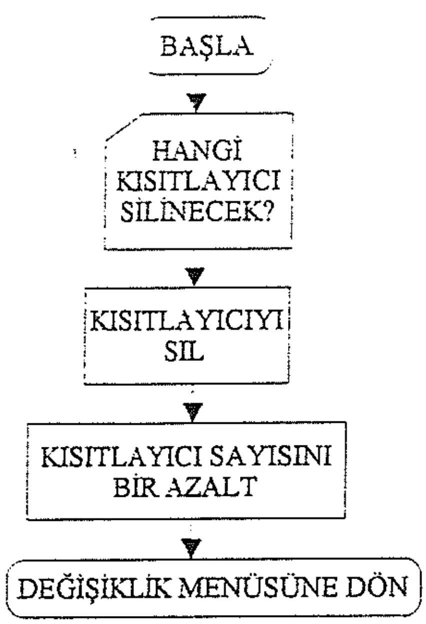
Ek 5.1. Amaç Fonksiyonunun Katsayılarını Değiştirme Blok Şeması

Ek 5.2. Kısıtlayıcı Değişikliği Blok Şeması

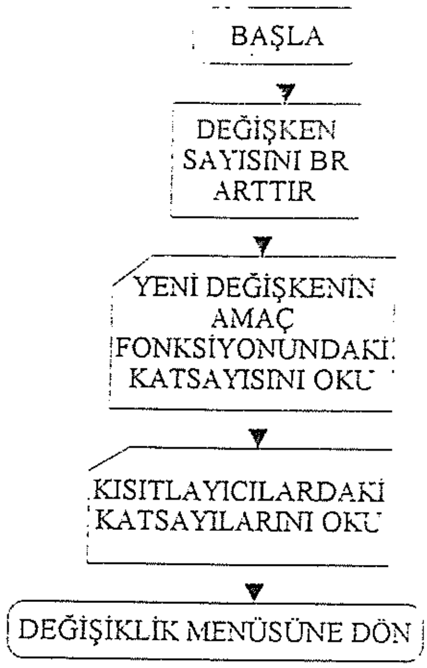


Ek 5. 3. Kısıtlayıcı Ekleme Blok Şeması

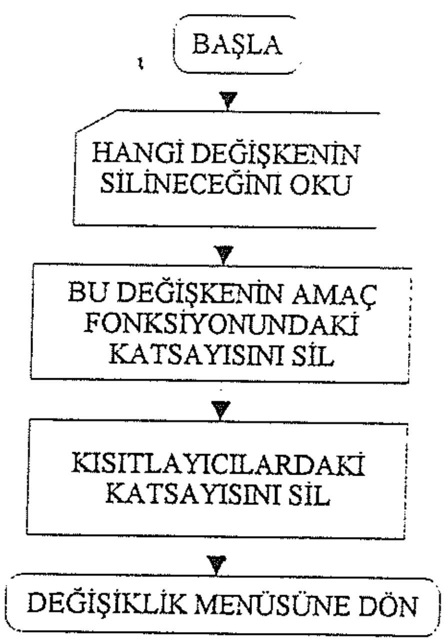
Ek 5.4. Kısıtlayıcı Silme Blok Şeması



Ek 5. 5. Değişken Ekleme Blok Şeması



Ek 5.6. Değişken Silme Blok Şeması



Ek 6. Program Hakkında Bilgi Seçeneği Modül Programı (Bilgi.Bas)

```
COLOR 10, 0, 5: CLS
PRINT "BU PROGRAM DOGRUSAL PROGRAMLAMA MODELLERINI COZMEK"
PRINT "ICIN GELISTIRILMISTIR."
PRINT "Programda ok tuşları ile aşağıya veya yukarıya hareket ederek istediğiniz "
PRINT "seçeneği n üzerine gelip.enter tuşuna basarak seçiminizi yapabilirsiniz."
PRINT "   Bu program en cok 100 degişkenli ve 100 kısıtlayıcılı dogrusal "
PRINT "programlama problemlerini çözer."
PRINT "   Aşagıda dogrusal programlama ile çözülecek bir modelin nasıl "
PRINT " kurulacagına ait bir örnek verilmiştir. "
                                                      "
PRINT "              Örnek:
PRINT " Bir firma ürün1 ve ürün2 olarak iki farklı mamülü üretmek için makine1 ve "
PRINT " makine2'yi kullanıyor. Ürün1den 5TL,ürün2den ise 7TL kar ediliyor. Makinelerin "
PRINT "çalışma kapasiteleri sırasıyla 60 saat ve 80 saattir. "
PRINT "Ürün1'i üretmek için makine1 2 saat ve makine2 4 saat,ürün2'yi üretmek için ise
makine1 3 saat ve makine2 2 saat çalışmalıdır. "
PRINT "Bu durumda maximum kar etmek için ürün1 ve ürün2'den kaç tane üretilmelidir?  "
PRINT "       Veriler :   "
PRINT "       Max      f=5*ürün1+7*ürün2"
PRINT "      Subject to  (1) 2ürün1+3ürün2<=60"
PRINT "                  (2) 4ürün1+2ürün2<=80 "
PRINT "                  ürün1>=0 ,  ürün2>=0"
PRINT
PRINT "Probleminizi bu formata getirdikten sonra katsayıları girerek problemi"
PRINT "çözebilirsiniz";
PRINT : DO: LOOP UNTIL INKEY$ <> "": CLS
LOCATE 10, 2:
PRINT "Dogrusal program modelinizdeki degişken isimleri program tarafından"
PRINT "otomatik olarak x1.x2,x3... şeklinde belirlenecektir."
PRINT "Problemi girmek için ana menuden bilgi girişini seciniz. Karsınıza "
PRINT "Klavyeden,Dosyadan secenekleri gelecek. Klavyeden secenegini secerseniz"
PRINT "sırasıyla degisken sayısını,kısıtlayıcı sayısını ve optimizasyonu soracak."
PRINT "Problemimizde ürün1 ve ürün2 olarak 2 değişken ,makine1 ve makine2'nin"
PRINT "kapasitelerinden dolayıda 2 kısıtlayıcı vardır. Amac ise maksimizasyon "
PRINT "oldugundan 1 dir. Bunları girdikten sonra karsımıza:"
DO: LOOP UNTIL INKEY$ <> "": CLS
PRINT "Amac fonksiyonu"
PRINT "      ----x1  ----x2"
PRINT "1. kisitlayici"
PRINT " ----x1  ----x2 < ----"
PRINT "2.kisitlayici"
PRINT " ----x1  ----x2 < ----"
PRINT "gelir .Sorulan bu değerleri girdikten sonra problem artık çözüme"
PRINT "hazırdır."
PRINT : DO: LOOP UNTIL INKEY$ <> "": COLOR 15, 1, 0: CLS
CHAIN "atadp"
```

Ek 7. ATADP programının QBASIC dilindeki kodlaması (ATADP.BAS)

```
'
' *************************   A T A D P   ****************************
'
' Bu program verilen bir dogrusal programlama modelini Revised Simpleks
' Metod kullanarak cözer. Progam Quick Basic ile yazilmistir.
' Bir dogrusal programlama modeli maksimizasyon veya minimizasyon olabilecek
' bir amaca,bu amaci gerceklestirebilecek bir fonksiyona ve eldeki mevcut
' kaynaklarin hangisinden ne kadar kullanilacagini belirten kisitlayicilara
' sahiptir.
' Program en fazla 100 degiskenli ve 100 kisitlayicili bir modeli cozebilir.
' Modeli girerken önce modeldeki degisken sayisi,kisitlayici sayisi ve amac
' degeri istenmektedir. Amac maksimizasyon (1) veya minimizasyon(2) olabilir.
' Degisken ve kisitlayici sayisi en fazla 100 olabilir. Modeldeki degisken
' isimleri otomatik olarak program tarafindan x1,x2,x3...seklinde verilir.
' Modelin cozumu ise tablolar halinde verilmektedir.
  DECLARE SUB MENU1 (x%, y%, SECIMSAYISI!, SECILEN!)
  DECLARE SUB YAZ (x%, y%, e$, e!)
  DECLARE SUB SOLVE (sol, var, cons, opt, it)
  DIM SHARED a$(1 TO 20)
  DIM SHARED b$(1 TO 20)
  DIM tdd(101), tgd(101), oran(101), mat(100, 100)
  REDIM b(101, 200), km(200), aa$(200), bb$(100)
  DIM c(150), ek(150), esay(100)
  REDIM a(100, 102)
  COLOR 15, 1, 0
  CLS
  DO
    LOCATE 7, 30: PRINT "A N A   M E N U"
    a$(1) = "      PROGRAM HAKKINDA BILGI   "
    a$(2) = "      BILGI GIRISI             "
    a$(3) = "      ÇÖRÜNTÜLEME              "
    a$(4) = "      ÇÖZÜM                    "
    a$(5) = "      KORUMA                   "
    a$(6) = "      DEGISIKLIK               "
    a$(7) = "      CIKIS                    "
    MENU1 9, 20, 7, sd
    ON sd GOSUB 1000, 2000, 4000, 5000, 6000, 7000, 8000
  LOOP
1000
'
' Program hakkinda bilgi
'
  CHAIN "bilgi"
2000
'
' Problem bilgi girisi. Model bilgisayara ilk defa giriliyorsa klavyeden
' secenegi,eger onceden bir dosyaya kaydedilmisse dosyadan secenegi ter
' cih edilir.
'
  COLOR 15, 1, 0
  CLS
  DO
    LOCATE 7, 28: PRINT "BILGI GIRISI"
    a$(1) = "      KLAVYEDEN      "
    a$(2) = "      DOSYADAN       "
    MENU1 9, 20, 2, sd
    ON sd GOSUB 100, 200
```

```
      RETURN
    LOOP
  100
  '
  ' KLAVYEDEN bilgi girisi. Once modeli tanimlayici uc unsur olan degisken
  ' sayisi,kisitlayici sayisi ve amac girilir. Daha sonrada verilen
  ' boyutlarda bir modelin amac fonksiyonu ve kisitlayicilardaki degisken
  ' lerinin katsayilari girilir.
  '
    COLOR 10, 0, 1: CLS
    COLOR 15
  '
  ' Degisken sayisi girisini kontrol eder. Degisken sayisi 100' den kücük
  ' ve sayisal olmalidir. Backspace tusu ile de yanlis girilen bilgiler
  ' duzeltilebilir.
  '
    DO
      LOCATE 10, 20: PRINT "Degisken sayisini giriniz?"
      karek$ = ""
      t$ = "_"
      LOCATE 10, 57: PRINT t$
      FOR i = 1 TO 3
        DO
          DO
            k$ = INKEY$
          LOOP UNTIL k$ <> ""
          IF ASC(k$) = 8 THEN
  '
  ' Eger backspace tusuna basilmis ise bilgi girisi bir geriye alinir.
  '
            i = i - 1
            IF i < 1 THEN i = 1
            IF LEN(karek$) <> 0 THEN
              karek$ = LEFT$(karek$, LEN(karek$) - 1)
              LOCATE 10, 55 + i
              PRINT "_"; "                    "
            END IF
          END IF
          IF ASC(k$) = 13 THEN EXIT FOR
  '
  ' Enter tusuna basildi ise bilgi girisi tamamlanmistir. Karakter olarak
  ' alinan bilgi sayisal hale donusturulur.
  '
          IF k$ < CHR$(48) OR k$ > CHR$(57) THEN
  '
  ' Klavyeden rakamdan farkli bir veri girilirse kabul edilmez.
  '
            BEEP
            a = 2
          ELSE
            a = 0
          END IF
        LOOP UNTIL a <> 2
        k$ = k$ + t$
        LOCATE 10, 56 + i: PRINT k$
        k$ = LEFT$(k$, 1)
        karek$ = karek$ + k$
      NEXT
      var = VAL(karek$)
      LOCATE 10, 57: PRINT var
      IF var = 0 OR var > 100 THEN
```

```
'
' Degisken sayisi olarak 0 veya 100'den buyuk bir sayi girilmis kabul edilmez.
' Yeniden uygun bir degisken sayisi girilmesi istenir.
'
        BEEP
        b = 1
      ELSE
        b = 0
      END IF
    LOOP UNTIL b <> 1
'
' Kisitlayici sayisi girisini kontrol eder. Kisitlayici sayisi en fazla 100
' olabilir. Karakter bilgi girilemez. Redo from Start hata mesaji onlenmistir.
'
    DO
      LOCATE 12, 20: PRINT "Kisitlayici sayisini giriniz  "
      karek$ = ""
      COLOR 15: t$ = "_"
      LOCATE 12, 56: PRINT t$
      FOR i = 1 TO 3
        DO
          DO
            k$ = INKEY$
          LOOP UNTIL k$ <> ""
          IF ASC(k$) = 8 THEN
            i = i - 1
            IF i < 1 THEN i = 1
            IF LEN(karek$) <> 0 THEN
              karek$ = LEFT$(karek$, LEN(karek$) - 1)
              LOCATE 12, 55 + i
              PRINT "_"; "                    "
            END IF
          END IF
          IF ASC(k$) = 13 THEN EXIT FOR
          IF k$ < CHR$(48) OR k$ > CHR$(57) THEN
            BEEP
            a = 2
          ELSE
            a = 0
          END IF
        LOOP UNTIL a <> 2
        k$ = k$ + t$
        LOCATE 12, 55 + i: PRINT k$
        k$ = LEFT$(k$, 1)
        karek$ = karek$ + k$
      NEXT
      cons = VAL(karek$)
      LOCATE 12, 56: PRINT cons
      IF cons = 0 OR cons > 100 THEN
        BEEP
        b = 1
      ELSE
        b = 0
      END IF
    LOOP UNTIL b <> 1
'
' Amac degeri girisini kontrol eder. Amac 1(maksimizasyon) veya 2(minimizasyon)  dan farkli deger alamaz.
'
    DO
      LOCATE 14, 20
```

```
      PRINT "Amaci giriniz. Maksimizasyon(1)? Minimizasyon (2)?"
      DO
         k$ = INKEY$
      LOOP UNTIL k$ <> ""
      IF k$ <> CHR$(49) AND k$ <> CHR$(50) THEN
         BEEP
         a = 1
      ELSE
         a = 0
      END IF
   LOOP UNTIL a <> 1
   LOCATE 14, 67 + i: PRINT k$
   opt = VAL(k$)
   COLOR 10
   LOCATE 18, 10
   PRINT "Bu program  x1,x2,..xn  gibi default degisken isimleri kullanir"
   LOCATE 22, 10
   PRINT "Degisiklik icin ( "; CHR$(27); ")backspace tusuna basiniz"
   LOCATE 21, 10
   PRINT "Devam icin bir tusa basiniz..."
   DO
      k$ = INKEY$
   LOOP UNTIL k$ <> ""
   IF ASC(k$) = 8 THEN 2000
   CLS
   IF opt = 1 THEN opt$ = "MAX" ELSE opt$ = "MIN"
   KIS1 = 1
   aa = 0: ss = 0
   COLOR 10: CLS
   LOCATE 1, 1: PRINT opt$; " Amac fonksiyonunun katsayilari": a = 0
'
' Amac fonksiyonunun degiskenlerinin yazdirilmasi
'
   DO
      FOR i = 1 TO 10
         FOR j = 1 TO 5
            a = a + 1
            LOCATE i * 2, j * 15 - 10
            PRINT "----"; : COLOR 9
            PRINT "X"; a: COLOR 10
            IF a = var THEN
               EXIT FOR
            END IF
         NEXT j
         IF a = var THEN
            EXIT FOR
         END IF
      NEXT i
'
' Eger amac fonksiyonu bir sayfaya sigabilecek boyutta ise kisitlayicilar
' da ayni sayfaya yazdirilir.
'
      IF a = var THEN
         kis = 1: deg = 0
         DO
            b = CSRLIN
            ai = (b + 1) / 2
            IF ai <= 10 THEN
               LOCATE ai * 2 - 1, 5: PRINT kis; " . kisitlayici"
               LOCATE ai * 2, 1
               c$ = STR$(kis)
```

```
            d$ = "(" + c$ + ")"
            PRINT d$
         END IF
         FOR i = ai TO 10
            FOR j = 1 TO 5
               deg = deg + 1
               LOCATE i * 2, j * 15 - 10
               PRINT "——"; : COLOR 9
               PRINT "X"; deg: COLOR 10
               IF deg = var THEN
                  sat = i
                  sut = j
                  EXIT FOR
               END IF
            NEXT j
            IF deg = var THEN
               EXIT FOR
            END IF
         NEXT i
         IF deg = var THEN
'
' Kisitlayicilarin degisken katsayilarinin girilmesi tamalandi ise
' kisitlama yonu(<,=,>) ve kisitlayici deger girilir.
'
            IF j = 5 THEN
               j = 1: i = i + 1
               IF i > 10 THEN
                  GOSUB 2120
                  CLS
                  i = 1: j = 1
               END IF
            ELSE
               j = j + 1
            END IF
            z$ = "ς"
            LOCATE i * 2, j * 15 - 10: PRINT z$: j = j + 1
            IF j = 6 THEN
               j = 1: i = i + 1
               IF i > 10 THEN
'
' Bir sayfa doldu ise bu sayfada goruntulenen degisken katsayilarinin giril
' mesi icin program 2120 nolu satira dallanir.
'
                  GOSUB 2120
                  j = 1
                  i = 1
                  CLS
               END IF
            END IF
            LOCATE i * 2, j * 15 - 10: PRINT "——"
            IF kis = cons THEN
'
' Son kisitlayici degiskenleri de goruntulandikten sonra bu degiskenlere
' ait katsayilarin girimesi icin program 2120 nolu satira dallanir.
'
               GOSUB 2120
               COLOR 15, 1: CLS
               RETURN
            END IF
            kis = kis + 1: deg = 0
         ELSE
```

```
            GOSUB 2120
            CLS
          END IF
        LOOP
      ELSE
            GOSUB 2120
            CLS
      END IF
    LOOP
2120
    ss = 1
    DO
      IF aa = var THEN EXIT DO
'
' Amac fonksiyonu katsayılari girilmis ise bu donguden cıkar.
'
      FOR i = 1 TO 10
        FOR j = 1 TO 5
          aa = aa + 1
          LOCATE i * 2, j * 15 - 10
          INPUT " ", a(0, aa)
          IF aa = var THEN
            ss = (CSRLIN + 1) / 2
            EXIT DO
          END IF
        NEXT j
      NEXT i
      RETURN
    LOOP
'
' Kisitlayicilara ait katsayilar girilir.
'
    FOR i = ss TO 10
      FOR j = 1 TO 5
        DEG1 = DEG1 + 1
        IF DEG1 = var + 1 THEN
          DO
            b = 0
            LOCATE i * 2, j * 15 - 10
            INPUT " ", BES
            IF BES <> "<" AND BES <> "=" AND BES <> ">" THEN
              BEEP
              b = 1
            END IF
          LOOP UNTIL b <> 1
          IF BES = "<" THEN a(KIS1, DEG1) = 0
          IF BES = ">" THEN a(KIS1, DEG1) = 2
          IF BES = "=" THEN a(KIS1, DEG1) = 1
        ELSE
          LOCATE i * 2, j * 15 - 10: INPUT " ", a(KIS1, DEG1)
        END IF
        IF DEG1 = var + 2 THEN
          KIS1 = KIS1 + 1: DEG1 = 0
          IF KIS1 = cons + 1 THEN
            DO
            LOCATE 22, 10: PRINT "Devam icin bir tusa basiniz..."
'
' Modele ait tum katsayilar girildi ise ana programa geri donulur.
'
            LOOP UNTIL INKEY$ <> ""
            RETURN
```

```
          END IF
          EXIT FOR
        END IF
      NEXT j
    NEXT i
    RETURN

200
'
' Bilgi girisi alt menusunde 'DOSYADAN' secenegi secilirse program
' bu satira dallanir.(Diskte veya disketteki bir dosyayi okuma)
'
    a = 1
    DO
      a = 0
      COLOR 15, 0
      CLS
      LOCATE 4, 15
      PRINT "Ana menuye dönmek icin sadece entere basiniz"
      LOCATE 5, 10
      PRINT "Disk(et)teki mevcut data dosyalarini listelemek icin dir yaziniz"
      LOCATE 6, 10
      PRINT "Disk(et)ten okunacak dosyanin adini giriniz."
      LOCATE 7, 10: INPUT " (a:pak.dat)", a$
      IF a$ = "" THEN
        COLOR 15, 1, 0
        CLS
        RETURN
      END IF
      IF a$ = "dir" THEN
        SHELL "dir/p *.dat"
        DO
        LOOP UNTIL INKEY$ <> ""
        a = 1
      END IF
      IF LEFT$(a$, 2) <> "a:" AND LEFT$(a$, 2) <> "A:" THEN
        IF LEFT$(a$, 2) <> "c:" AND LEFT$(a$, 2) <> "C:" THEN
          CLS
          a = 1
        END IF
      END IF
      IF RIGHT$(a$, 4) <> ".dat" AND RIGHT$(a$, 4) <> ".DAT" THEN
        a = 1
      END IF
    LOOP UNTIL a <> 1
    CLOSE #1
    ON ERROR GOTO 2222
    OPEN a$ FOR INPUT AS #1
    INPUT #1, opt, var, cons
    PRINT opt, var, cons
    FOR j = 1 TO var
'
' Amac fonksiyonu katsayilarini okur.
'
      INPUT #1, a(0, j)
    NEXT j
    FOR i = 1 TO cons
      FOR j = 1 TO var + 2
        INPUT #1, a(i, j)
'
' Kisitlayici katsayilarini okur.
```

```
'
    NEXT j
  NEXT i
  CLOSE #1
  LOCATE 21, 10: PRINT a$; " okundu"
  DO: LOOP UNTIL INKEY$ <> "": COLOR 15, 1, 0: CLS
  RETURN
4000
  pg = 1
'
' Modelin görüntülemesi
'
  COLOR 10: CLS
  IF var = 0 THEN
    LOCATE 5, 5
    PRINT "Görüntülenecek bir sey yok.Devam icin bir tusa basiniz."
    DO
    LOOP UNTIL INKEY$ <> ""
    COLOR 15, 1, 0
    CLS
    RETURN
  END IF
  CLS : pg = 1
  IF opt = 1 THEN opt$ = "MAX" ELSE opt$ = "MIN"
  LOCATE 1, 5: PRINT a$; " Probleminin "; opt$; " "; "Amaç fonksiyonu katsayilari"
  a = 1
'
' Amac fonksiyonunun goruntulenmesi
'
  DO
    FOR i = 1 TO 10
      FOR j = 1 TO 5
        LOCATE i * 2, j * 15 - 10
        PRINT a(0, a);
        COLOR 9
        PRINT "X"; a
        COLOR 10
        IF a = var THEN EXIT DO
        a = a + 1
      NEXT j
    NEXT i
    LOCATE 1, 70: PRINT "Sayfa"; pg
    DO
      LOCATE 22, 5:   PRINT "Devam icin bir tusa basiniz..."
    LOOP UNTIL INKEY$ <> "": CLS
    pg = pg + 1
  LOOP UNTIL a = var
  IF CSRLIN >= 20 THEN
    DO
    LOCATE 22, 5:   PRINT "Devam icin bir tusa basiniz..."
    LOOP UNTIL INKEY$ <> ""
    CLS
  END IF
'
'Kisitlayicilarin goruntulenmesi
'
  kis = 1: deg = 1: a = 1
  DO
    b = CSRLIN
    ai = (b + 1) / 2
    LOCATE 1, 70: PRINT "Sayfa"; pg
```

```
      LOCATE ai * 2 - 1, 10: PRINT kis; ". kisitlayici"
      IF ai <= 10 THEN
         LOCATE ai * 2, 1
         c$ = STR$(kis)
         d$ = "(" + c$ + ")"
         PRINT d$
      END IF
      FOR i = ai TO 10
         FOR j = 1 TO 5
            LOCATE i * 2, j * 15 - 10
            PRINT a(kis, deg);
            COLOR 9
            PRINT "X"; deg
            COLOR 10
            IF deg = var THEN
               sat = i
               sut = j
               deg = deg + 1
               IF j = 5 THEN j = 1: i = i + 1 ELSE j = j + 1
               IF a(kis, deg) = 0 THEN
                  z$ = "<"
               ELSE
                  IF a(kis, deg) = 1 THEN z$ = "=" ELSE z$ = ">"
               END IF
               LOCATE i * 2, j * 15 - 10
               PRINT z$
               deg = deg + 1
               j = j + 1
               IF j = 6 THEN j = 1: i = i + 1
               LOCATE i * 2, j * 15 - 10: PRINT a(kis, deg)
               LOCATE i * 2 + 1, 10
               IF kis < cons THEN
                  IF i * 2 + 1 < 20 THEN PRINT kis + 1; ". kisitlayici"
               END IF
               IF kis = cons THEN
                  DO: LOOP UNTIL INKEY$ <> ""
                  COLOR 15, 1, 0: CLS
                  RETURN
               END IF
               kis = kis + 1: deg = 1
               a = 2
               EXIT FOR
            END IF
            deg = deg + 1: a = 1
         NEXT j
         IF deg = var THEN EXIT FOR
      NEXT i
      pg = pg + 1
      IF CSRLIN >= 20 THEN
         DO
         LOCATE 22, 10: PRINT "Devam için bir tuşa basınız..."
         LOOP UNTIL INKEY$ <> ""
         CLS
      END IF
   LOOP UNTIL a = 0
   DO: LOOP UNTIL INKEY$ <> "": RETURN

5000
'
' Revised Simplex Method Genel Cözümü
' Programda iki evreli Revised Simpleks Metod Cozum algoritmasi izlenmistir.
```

```
' Modelde optimize edilecek bir amac fonksiyonu ve kisitlayicilar mevcuttur.
' Amac(hedef) ise bu kisitlayici sartlar altinda amac fonksiyonunu minimum
' veya maksimum yapan degisken degerlerini bulmaktir.
'
' Degiskenlerin sifirlanmasi
'
  FOR i = 1 TO cons + 1
    tgd(i) = 0
    tdd(i) = 0
  NEXT i
  t = 0: c = 0: va = 0: it = 0
  CLS
'
' Amac minimizasyon ise amac fonksiyonu katsayilari -1 ile carpilir.
'
  IF opt = 2 THEN
    FOR i = 1 TO var
      a(0, i) = -a(0, i)
    NEXT i
  END IF
'
' Eger kisitlayicilardan herhangi birinin kisit degeri negatif ise kisitin
' tamami -1 ile carpilarak kisit degeri pozitif hale getirilir.
'
  FOR j = 1 TO cons
    IF a(j, var + 2) < 0 THEN
      FOR k = 1 TO var
        a(j, k) = -a(j, k)
      NEXT k
      IF a(j, var + 1) = 0 THEN
        a(j, var + 1) = 2
      ELSE
        IF a(j, var + 1) = 2 THEN a(j, var + 1) = 0
      END IF
      a(j, var + 2) = -a(j, var + 2)
    END IF
  NEXT j
'
' Yapay ve bos degiskenlerin eklenmesi
' Kucuk kisitlayicilara yi bos degiskeni eklenerek esitlik haline getirilir.
' Esit kisitlayicilara zi yapay degiskeni eklenerek kanonik hale getirilir.
' Buyuk kisitlayicilardan once yi bos degiskeni(Problemde kucuk kisitlayici
' larin bos degiskeni ile karismamasi icin ti dendi) cikarilarak esitlik ha
' line,zi bos degiskeni eklenerek de kanonok hale getirilir.
'
  FOR i = 1 TO cons
    IF a(i, var + 1) = 0 THEN
      t = t + 1
      bb$(i + 1) = "y" + STR$(t)
    ELSE
      c = c + 1
      bb$(i + 1) = "z" + STR$(c)
    END IF
    IF a(i, var + 1) = 2 THEN
      va = va + 1
      aa$(var + va) = "t" + STR$(va)
    END IF
  NEXT i
  FOR i = 1 TO var
    aa$(i) = "x" + STR$(i)
  NEXT i
```

```
'
' Iki evreli simpleks metodda amac fonksiyonu fc ve fm olarak iki kisma
' ayrilir.Modelin cozumunun olabilmesi icin,yapay degiskenlerin bir an
' once temelden atilmasi gerekir. Bu sebeple once yapay degiskenlerin
' olusturdugu fm amac fonksiyonu optimize edilir.Eger temelde yapay degisken
' kalmis ise modelin mumkun cozumu yoktur.
' Sonra fm satir ve sutunu silinerek fc amac fonksiyonu optimize edilir.
' Temele girecek degiskenler aa$(i),temelden ayrilacaklar bb$(i) dizilerinde
' saklanmaktadir.
' Temele girecek olan degiskenler, modelin degiskenleri ile büyük esit
' kisitlayicilardan cikarilan ti bos degiskenleridir.
' Temelden ayrilacak degiskenler ise yi bos degiskenleri ile zi yapay
' degiskenleridir.
'
  c = 0: t = 0: b(0, 0) = 0: b(1, 0) = 0
'
' KURULUS TABLOSU OLUSTURMA
' Simpleks tablo B matrisinde olusturulur. B matrisinin ilk sutununda(0.sutun)
' kisitlayicilarin kisit degerleri yani temel degisken degerleri saklanır.
' B matrisinin ilk satirinda (0.satir) fc fonksiyonu,ikinci satirinda
' fm fonksiyonu saklanmaktadir.
' Ucuncu satirdan itibaren ise kisitlayicilar yerlestirilmistir.
'
  FOR i = 2 TO cons + 1
    b(i, 0) = a(i - 1, var + 2)
  NEXT i
  FOR i = 0 TO var + va
    b(1, i) = 0
  NEXT i
  FOR i = 1 TO var
    b(0, i) = -a(0, i)
  NEXT i
  FOR i = 2 TO cons + 1
    FOR j = 1 TO var
      b(i, j) = a(i - 1, j)
    NEXT j
    IF a(i - 1, var + 1) = 2 THEN
      sv = sv + 1
      b(i, var + sv) = -1
    END IF
  NEXT i
'
' KURULUS TABLOSUNDAN BASLANGIC TABLOSUNA GECIS
' ">" veya "=" kisitlayicilarda Mat inverse matrisini kanonik hale
' getirmek icin fm satirindaki fm sutunu disindaki diger katsayilari
' sifirlamak icin islemler yapilarak Mat matrisi kanonik hale getirilir.
'
  FOR i = 2 TO cons + 1
    IF a(i - 1, var + 1) <> 0 THEN
      FOR j = 0 TO var + va
        b(1, j) = b(1, j) - b(i, j)
      NEXT j
    END IF
  NEXT i
  FOR i = 0 TO cons + 1
    FOR j = 0 TO cons + 1
      IF i = j THEN mat(i, j) = 1 ELSE mat(i, j) = 0
    NEXT j
  NEXT i
'
'******************************** COZUM ****************************************
```

```
5500
'
'************************BIRINCI EVRE**********************************
'
  FOR i = 1 TO var + va
    km(i) = 0
  NEXT i
  ek = 999999
'
' fm katsayilarinin hesabi
' Km(i) dizisi fm amac fonksiyonu katsayilarini icerir.
'
  FOR i = 1 TO var + va
    FOR j = 0 TO cons + 1
      km(i) = mat(1, j) * b(j, i) + km(i)
    NEXT j
'
' Temele girecek degiskenin(anahtar sutunun belirlenmesi)
' En buyuk negatif katsayili degisken temele giren degisken olarak secilir.
'
     IF km(i) < -.00001 THEN
       IF km(i) < ek THEN ek = km(i): c(it) = i
    END IF
  NEXT i
  FOR i = 0 TO cons + 1
    tdd(i) = 0: tgd(i) = 0
  NEXT i
  FOR i = 0 TO cons + 1
    FOR j = 0 TO cons + 1
      tgd(i) = tgd(i) + mat(i, j) * b(j, c(it))
      tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
    NEXT j
  NEXT i
'
' Temelden ayrilacak degiskenin(anahtar satirin belirlenmesi)
' En kucuk pozitif katsayili degisken temelden ayrilacak degisken olarak
' secilir.
'
  IF ek <> 999999 THEN
    FOR i = 2 TO cons + 1
      IF tgd(i)  > 0 THEN
        oran(i) = tdd(i) / tgd(i)
        IF oran(i) < 0 THEN oran(i) = -1
      ELSE
        oran(i) = -1
      END IF
    NEXT i
'
' Burada dejenerasyona bakilir(Birden fazla en kücük oran degeri)
'
    ek = 923655
    FOR i = 2 TO cons + 1
      IF oran(i) <> -1 THEN
        IF oran(i) < ek THEN ek = oran(i): ek(it) = i
      END IF
    NEXT i
'
' Oran degerlerinin hepsi 0dan kücükse temelden ayrilacak degisken yoktur.
' Yani sinirsiz cözüm vardir.
'
    IF ek = 923655 THEN
```

```
        sol = 1
        SOLVE sol, var, cons, opt, it
        RETURN
      END IF
      t = 1
'
' Dejenerasyon arastirmasi
'
      FOR i = 2 TO cons + 1
        IF i <> ek(it) THEN
          IF oran(i) = ek THEN t = t + 1: esay(t) = i
        END IF
      NEXT i
      IF t <> 1 THEN
        IF ek = 0 THEN GOSUB 5800 ELSE GOSUB 5700
      END IF
'
' aa$(c(it))  temele giriyor.
' bb$(ek(it)) temelden cikiyor
' Temele girecek degisken sütunu kanonik hale getirilir.
'
      bb$(ek(it)) = aa$(c(it))
      FOR i = 0 TO cons + 1
        mat(ek(it), i) = mat(ek(it), i) / tgd(ek(it))
      NEXT i
      FOR i = 0 TO cons + 1
        IF i <> ek(it) THEN
          FOR j = 0 TO cons + 1
            mat(i, j) = mat(i, j) - mat(ek(it), j) * tgd(i)
          NEXT j
        END IF
      NEXT i
      it = it + 1
      GOTO 5500
'
' Fm optimize edildikten sonra temelde 0 dan farkli bir katsayi ile yapay
' degisken varsa modelin mumkun cozumu yoktur.
' Eger bir yapay degisken mevcut fakat katsayisi 0 ise model optimize
' edilmistir. Artik ikinci evreye gecilmez.
'
  ELSE
    a = 0
      FOR i = 2 TO cons + 1
        IF LEFT$(bb$(i), 1) = "z" THEN
          IF tdd(i) <> 0 THEN
            sol = 2
            SOLVE sol, var, cons, opt, it
            RETURN
          ELSE
            a = 2
          END IF
        END IF
      NEXT i
    IF a = 2 THEN
      sol = 4
      SOLVE sol, var, cons, opt, it
      RETURN
    END IF
  END IF
'
' **********IKINCI EVRE************************
```

```
' fc amac fonksiyonunun optimize edilmesi
'
  FOR i = 2 TO cons + 1
    bb$(i - 1) = bb$(i)
  NEXT i
  k = 0: l = 0
  FOR i = 0 TO cons + 1
    IF i = 1 THEN i = i + 1
'
' fm'e ait satir ve sütunlar silinir.
'
    FOR j = 0 TO cons + 1
      IF j = 1 THEN j = j + 1
      mat(k, l) = mat(i, j)
      l = l + 1
    NEXT j
    k = k + 1: l = 0
  NEXT i
  k = 0
'
' Yeni B matrisi
'
  FOR i = 0 TO cons + 1
    IF i = 1 THEN i = i + 1
    FOR j = 0 TO var + va
      b(k, j) = b(i, j)
    NEXT j
    k = k + 1
  NEXT i
5600
  FOR i = 1 TO var + va
    km(i) = 0
  NEXT i
  FOR i = 0 TO cons
    tdd(i) = 0: tgd(i) = 0: oran(i) = 0
  NEXT i
  ek = 999999
  FOR i = 1 TO var + va
    FOR j = 0 TO cons
      km(i) = mat(0, j) * b(j, i) + km(i)
    NEXT j
    IF km(i) < -.000001 THEN
      IF km(i) < ek THEN ek = km(i): c(it) = i
    END IF
  NEXT i
  IF ek <> 971570 THEN
    FOR i = 0 TO cons
      FOR j = 0 TO cons
        tgd(i) = tgd(i) + mat(i, j) * b(j, c(it))
        tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
      NEXT j
      IF tgd(i)  > 0 THEN
        oran(i) = tdd(i) / tgd(i)
        IF oran(i) < 0 THEN oran(i) = -1
      ELSE
        oran(i) = -1
      END IF
    NEXT i
'
' aa$(c(it)) temele giriyor.
' Burada dejenerasyona bakilir.
```

```
'
    ek = 953655
    FOR i = 1 TO cons
       IF oran(i) <> -1 THEN
          IF oran(i) < ek THEN ek = oran(i): ek(it) = i
       END IF
    NEXT i
    IF ek = 953655 THEN
       sol = 1
       SOLVE sol, var, cons, opt, it
       RETURN
    END IF
    t = 1
'
' Dejenerasyon arastirmasi
'
    FOR i = 1 TO cons
       IF i <> ek(it) THEN
          IF oran(i) = ek THEN t = t + 1: esay(t) = i
       END IF
    NEXT i
    IF t <> 1 THEN
       IF ek = 0 THEN GOSUB 5800 ELSE GOSUB 5700
    END IF
'
' bb$(ek(it)) temelden cikiyor.
' Temele girecek degisken sütununu kanonik hale getirmek
'
    bb$(ek(it)) = aa$(c(it))
    FOR i = 0 TO cons
       mat(ek(it), i) = mat(ek(it), i) / tgd(ek(it))
    NEXT i
    FOR i = 0 TO cons
       IF i <> ek(it) THEN
          FOR j = 0 TO cons + 1
             mat(i, j) = mat(i, j) - mat(ek(it), j) * tgd(i)
          NEXT j
       END IF
    NEXT i
    it = it + 1
    GOTO 5600
  END IF
  FOR i = 0 TO cons
    FOR j = 0 TO cons
       tdd(i) = tdd(i) + mat(i, j) * b(j, 0)
    NEXT j
  NEXT i
  sol = 4
  SOLVE sol, var, cons, opt, it
  RETURN

5700
'
' Anahtar satir seciminde 0 dan farkli en kucuk oran degeri birden fazla
' olursa dejenerasyon ile karsilasilir. Bu dejenerasyonun onlenmesi icin
' esit oranli satirdaki her eleman bulundugu satirin anahtar sutun uzerin
' deki elemanina bolunerek cesitli degerler elde edilir. u degerler soldan
' saga dogru esit oranli satirlarin her sutunu icin mukayese edilerek,esit
' olmayan ilk mukayesede en kucuk oran degerini saglayan satir anahtar satir
' olarak secilir.
'
```

```
  esay(1) = ek(it)
  FOR j = 1 TO cons + 1
    ek = 34986
    FOR i = 1 TO t
      oran(i) = mat(esay(i), j) / tgd(esay(i))
      IF oran(i) < ek THEN ek = oran(i): ek(it) = esay(i)
    NEXT i
    FOR i = 1 TO t
      IF i <> ek(it) THEN
        IF oran(i) = ek THEN esp = 1
      END IF
    NEXT i
    IF esp = 0 THEN j = cons
  NEXT j
  RETURN
5800
'
' Perturbation metod
' Bu dejenerasyon durumuna birden fazla en kucuk oran degerinin sifir oldugu
' durumlarda karsilasilir. Sıfır olan temel degisken degerlerine sifirdan
' buyuk fakat birden cok kucuk bir sayinin kuvvetleri eklenir.
' Bu temel degisken degerlerine gore en kucuk oran degeri belirlenir.
'
  car = 1: esay(1) = ek(it)
  ek = 67667
  FOR i = 1 TO t
    car = car * .1175
    tdd(esay(i)) = car
    oran(i) = tdd(esay(i)) / tgd(esay(i))
    IF oran(i) < ek THEN ek = oran(i): ek(it) = esay(i)
  NEXT i
  RETURN


6000
' Dosya saklama
  COLOR 15, 0: CLS
6100
  IF var = 0 THEN
    PRINT "Kaydedilecek birsey yok"
    DO
    LOOP UNTIL INKEY$ <> ""
    COLOR 15, 1, 0: CLS
    RETURN
  END IF
  a = 1
  DO
    a = 0
    LOCATE 5, 10
    INPUT "Korunacak model icin bir isim giriniz.(example a:pak.dat)"; a$
' Surucu ismi A veya C den farklı ise yeniden isim girilir.
' Dosya uzantısı .Dat dan farkli ise yeniden isim girilir.
    IF LEFT$(a$, 2) <> "a:" AND LEFT$(a$, 2) <> "A:" THEN
      IF LEFT$(a$, 2) <> "c:" AND LEFT$(a$, 2) <> "C:" THEN
        CLS : a = 1
      END IF
    END IF
    IF RIGHT$(a$, 4) <> ".dat" AND RIGHT$(a$, 4) <> ".DAT" THEN
      CLS : a = 1
    END IF
  LOOP UNTIL a <> 1
```

```
ON ERROR GOTO 3333
OPEN a$ FOR OUTPUT AS #1
PRINT #1, opt, var, cons
'
' Amac fonksiyonu katsayilari saklanir.
'
FOR j = 1 TO var
   PRINT #1, a(0, j)
NEXT j
'
' Kisitlayici katsayilari saklanir.
'
FOR i = 1 TO cons
   FOR j = 1 TO var + 2
      PRINT #1, a(i, j)
   NEXT j
NEXT i
CLOSE #1
LOCATE 21, 10: PRINT a$; " saklandi"
DO: LOOP UNTIL INKEY$ <> "": COLOR 15, 1, 0: CLS
RETURN
3333
 IF ERR = 71 THEN
    PRINT "Disk hazir degil."
    DO WHILE INKEY$ <> "": LOOP
    RESUME 6100
 END IF
 IF ERR = 72 THEN
    PRINT "Disk-media hatasi."
    DO WHILE INKEY$ <> ""
    LOOP
    RESUME 6100
 END IF
 IF ERR = 70 THEN
    PRINT "Erisim engellendi"
    DO WHILE INKEY$ <> ""
    LOOP: RESUME 6100
 END IF
7000
'
' Model uzerinde degisiklik yapma imkani saglar.
' Bu alt menude amac fonksiyonunun katsayilarini degistirilebilir.
' Mevcut bir kisitlayici üzerinde degisiklikler yapilabilir.
' Modele yeni kisitlayicilar eklenebilir.
' Mevcut bir kisitlayici silinebilir.
' Modele yeni bir degisken eklenebilir veya cikarabilir.
'
 COLOR 15, 1
 CLS
 a$(1) = "  Amac Fonksiyon Katsayilarini degistirme    "
 a$(2) = "  Kisitlayiciyi degistirme                    "
 a$(3) = "  Kisitlayici Ekleme                          "
 a$(4) = "  Kisitlayici Silme                           "
 a$(5) = "  Degisken Ekleme                             "
 a$(6) = "  Degisken Silme                              "
 a$(7) = "  Ana Menu' ye Dönüs                          "
 DO
    LOCATE 20, 15
    PRINT "Degisiklikleri saklamayi unutmayiniz."
    LOCATE 7, 28: PRINT "PROBLEM UZERINDE DEGISIKLIK"
    MENU1 9, 18, 7, sd
```

```
      ON sd GOSUB 111, 222, 333, 444, 555, 666, 722
      COLOR 15, 1: CLS
      IF sd = 7 THEN COLOR 15, 1, 0: CLS : RETURN
   LOOP
111
'
' Amac fonksiyon katsayilarini degistirir.
'
   IF opt = 1 THEN opt$ = "MAX" ELSE opt$ = "MIN"
   ps = 1: COLOR 15, 0: CLS
   DO
      IF var = 0 THEN
         PRINT "Degistirilecek birsey yok"
         PRINT "Devam icin bir tusa basiniz..."
         DO: LOOP UNTIL INKEY$ <> ""
         RETURN
      END IF
      LOCATE 1, 10
      PRINT opt$; " Amac fonksiyonun katsayilarini degistiriniz."
      FOR i = 1 TO 10
         FOR j = 1 TO 5
            LOCATE i * 2, (j - 1) * 15 + 1
            COLOR 9: PRINT a(0, j + 5 * (i - 1) + 50 * (ps - 1));
            COLOR 10: PRINT "  "; "x"; j + 5 * (i - 1) + 50 * (ps - 1)
            IF j + 5 * (i - 1) + 50 * (ps - 1) = var THEN
               a = 1
               EXIT FOR
            ELSE
               a = 0
            END IF
         NEXT j
         IF a = 1 THEN
            EXIT FOR
         END IF
      NEXT i
      FOR i = 1 TO 10
         FOR j = 1 TO 5
            gd = a(0, j + 5 * (i - 1) + 50 * (ps - 1))
            LOCATE i * 2, (j - 1) * 15 + 1
            COLOR 9: PRINT "";
            INPUT "", a(0, j + 5 * (i - 1) + 50 * (ps - 1))
            IF a(0, j + 5 * (i - 1) + 50 * (ps - 1)) = 0 THEN
               a(0, j + 5 * (i - 1) + 50 * (ps - 1)) = gd
            END IF
            IF j + 5 * (i - 1) + 50 * (ps - 1) = var THEN
               a = 1
               EXIT FOR
            ELSE
               a = 0
            END IF
         NEXT j
         IF a = 1 THEN
            EXIT FOR
         END IF
      NEXT i
      IF j + 5 * (i - 1) + 50 * (ps - 1) <> var THEN
         ps = ps + 1
         PRINT "Devam icin bir tusa basiniz..."
         DO: LOOP UNTIL INKEY$ <> "": CLS
      ELSE
         EXIT DO
```

```
    END IF
  LOOP
  PRINT "Devam icin bir tusa basiniz...": DO: LOOP UNTIL INKEY$ <> ""
  RETURN

222
'
' Modelin istenen kisitlayicisina ait katsayilari degistirir.
'
  COLOR 15, 0: CLS
  IF var = 0 THEN
    PRINT "Degistirilecek bir kisitlayici mevcut degil"
    PRINT "Devam icin bir tusa basiniz...": DO: LOOP UNTIL INKEY$ <> ""
    RETURN
  END IF
  g = 1: h = 1
  INPUT "Hangi kisitlayici degistirilecek?"; w: CLS
  LOCATE 1, 10: PRINT w; ".Kisitlayicinin katsayilari"
  DO
    h = 1
    FOR i = 1 TO 10
      FOR j = 1 TO 5
        LOCATE i * 2, (j - 1) * 15 + 1
        COLOR 9: PRINT a(w, h + 50 * (g - 1));
        COLOR 10: PRINT " "; "X"; h + 50 * (g - 1)
        IF h + 50 * (g - 1) = var THEN
          IF j = 5 THEN j = 1: i = i + 1 ELSE j = j + 1
          IF a(w, h + 1 + 50 * (g - 1)) = 0 THEN
            z$ = "<"
          ELSE
            IF a(w, h + 1 + 50 * (g - 1)) = 1 THEN
              z$ = "="
            ELSE
              z$ = ">"
            END IF
          END IF
          LOCATE i * 2, (j - 1) * 15 + 1
          PRINT z$
          j = j + 1
          IF j = 6 THEN j = 1: i = i + 1
          LOCATE i * 2, (j - 1) * 15 + 1
          PRINT a(w, h + 2 + 50 * (g - 1))
          EXIT FOR
        END IF
        h = h + 1
      NEXT j
      IF h + 50 * (g - 1) = var THEN EXIT FOR
    NEXT i
    h = 1
    FOR i = 1 TO 10
      FOR j = 1 TO 5
        od = a(w, h + 50 * (g - 1))
        LOCATE i * 2, (j - 1) * 15 + 1: COLOR 9
        IF h + 50 * (g - 1) = var + 3 THEN
          PRINT "Devam icin bir tusa basiniz..."
          DO: LOOP UNTIL INKEY$ <> ""
          RETURN
        END IF
        IF h + 50 * (g - 1) = var + 1 THEN
          DO
            b = 0
```

```
                LOCATE i * 2, (j - 1) * 15 + 1
                INPUT " ", BE$
                IF BE$ <> "<" AND BE$ <> "=" AND BE$ <> ">" THEN
                   BEEP
                   b = 1
                END IF
             LOOP UNTIL b <> 1
             IF BE$ = "<" THEN a(w, h + 50 * (g - 1)) = 0
             IF BE$ = ">" THEN a(w, h + 50 * (g - 1)) = 2
             IF BE$ = "=" THEN a(w, h + 50 * (g - 1)) = 1
          ELSE
             INPUT "", s$
             IF s$ = "" THEN
                a(w, h + 50 * (g - 1)) = od
             ELSE
                a(w, h + 50 * (g - 1)) = VAL(s$)
             END IF
          END IF
          h = h + 1
       NEXT j
    NEXT i
    g = g + 1
    LOCATE 22, 10:
    PRINT "Devam icin bir tusa basiniz..."
    DO: LOOP UNTIL INKEY$ <> ""
    CLS
 LOOP
333
'
' Modele kisitlayici ekler.
'
 COLOR 15, 0: CLS
 IF var = 0 THEN
    PRINT "Degisecek bir problem yok "
    PRINT "Devam için bir tusa basiniz..."
    DO: LOOP UNTIL INKEY$ <> ""
    RETURN
 END IF
 g = 1: cons = cons + 1: w = cons
 LOCATE 1, 10: PRINT w; ".kisitlayicinin katsayilarini giriniz."
 DO
    h = 1
    FOR i = 1 TO 10
       FOR j = 1 TO 5
          LOCATE i * 2, (j - 1) * 15 + 1
          COLOR 9: PRINT "________";
          COLOR 10: PRINT " "; "X"; h + 50 * (g - 1)
          IF h + 50 * (g - 1) = var THEN
             IF j = 5 THEN j = 1: i = i + 1 ELSE j = j + 1
             z$ = "<"
             LOCATE i * 2, (j - 1) * 15 + 1: PRINT z$
             j = j + 1
             IF j = 6 THEN j = 1: i = i + 1
             LOCATE i * 2, (j - 1) * 15 + 1: PRINT "________"
             EXIT FOR
          END IF
          h = h + 1
       NEXT j
       IF h + 50 * (g - 1) = var THEN EXIT FOR
    NEXT i
    h = 1
```

```
        FOR i = 1 TO 10
           FOR j = 1 TO 5
              IF h + 50 * (g - 1) = var + 1 THEN
                 DO
                    b = 0
                    LOCATE i * 2, (j - 1) * 15 + 1: INPUT " ", BE$
                    IF BE$ <> "<" AND BE$ <> "=" AND BE$ <> ">" THEN
                       BEEP
                       b = 1
                    END IF
                 LOOP UNTIL b <> 1
                 IF BE$ = "<" THEN a(w, h + 50 * (g - 1)) = 0
                 IF BE$ = ">" THEN a(w, h + 50 * (g - 1)) = 2
                 IF BE$ = "=" THEN a(w, h + 50 * (g - 1)) = 1
              ELSE
                 IF h + 50 * (g - 1) = var + 3 THEN
                    PRINT "Devam icin bir tusa basiniz..."
                    DO: LOOP UNTIL INKEY$ <> ""
                    RETURN
                 END IF
                 LOCATE i * 2, (j - 1) * 15 + 1
                 COLOR 9: INPUT "", a(w, h + 50 * (g - 1))
              END IF
              h = h + 1
           NEXT j
        NEXT i
        g = g + 1
        LOCATE 22, 10:
        PRINT "Devam icin bir tusa basiniz..."
        DO: LOOP UNTIL INKEY$ <> "": CLS
     LOOP
   444
   '
   ' Modelden kisitlayici siler.
   '
     COLOR 15, 0: CLS
     IF var = 0 THEN
        PRINT "Silinecek bir kisitlayici yok"
        PRINT "Devam icin bir tusa basiniz..."
        DO: LOOP UNTIL INKEY$ <> ""
        RETURN
     END IF
     COLOR 15, 0: CLS
     INPUT "     Hangi kisitlayici silinecek?", kis
     INPUT "     Emin misiniz?(e/h)", F$
     IF F$ <> "e" AND F$ <> "E" THEN RETURN
     cons = cons - 1
     FOR i = kis TO cons
        FOR j = 1 TO var + 2
           a(i, j) = a(i + 1, j)
        NEXT j
     NEXT i
     LOCATE 20, 25: PRINT kis; ".kisitlayici silindi."
     PRINT "Devam icin bir tusa basiniz..."
     DO: LOOP UNTIL INKEY$ <> ""
     RETURN
   555
   '
   ' Modele degisken ekler.
   '
     COLOR 15, 0: CLS
```

```
    IF var = 0 THEN
      PRINT "Degistirilecek bir problem yok"
      PRINT "Devam icin bir tusa basiniz..."
      DO: LOOP UNTIL INKEY$ <> ""
      RETURN
    END IF
    CLS
    INPUT "Modele degisken eklemek istediginizden emin misiniz(e/h)?", F$
    IF F$ <> "e" AND F$ <> "E" THEN RETURN
    var = var + 1
    PRINT "Default degisken ismi  X"; var
    PRINT "X"; var; "in amac fonksiyonundaki katsayisini giriniz."
    INPUT " ", a(0, var)
    FOR i = 1 TO cons
      k = a(i, var): kd = a(i, var + 1)
      PRINT " X"; i; "kisitlayicisindaki katsayi "; i;
      INPUT a(i, var)
      a(i, var + 1) = k: a(i, var + 2) = kd
    NEXT i
    PRINT "X"; var; "degiskeni modele eklendi"
    PRINT "Devam icin bir tusa basiniz..."
    DO: LOOP UNTIL INKEY$ <> ""
    RETURN
666
'
' Modelden degisken siler.
'
    CLS
    IF var = 0 THEN
      PRINT "Silinecek bir problem yok"
      PRINT "Devam icin bir tusa basiniz..."
      DO: LOOP UNTIL INKEY$ <> "": RETURN
    END IF
    INPUT "Degisken silmek istediginizden emin misiniz(e/h)?", F$
    IF F$ <> "e" AND F$ <> "E" THEN RETURN
    INPUT "Silinecek degisken ismini giriniz", dv
    var = var - 1
    FOR i = 1 TO cons
      FOR j = dv TO var + 2
        a(i, j) = a(i, j + 1)
      NEXT j
    NEXT i
    FOR j = dv TO var
      a(0, j) = a(0, j + 1)
    NEXT j
    PRINT "X"; dv; "degiskeni bu modelden silindi"
    PRINT "Devam icin bir tusa basiniz...": DO: LOOP UNTIL INKEY$ <> ""
    RETURN
722
  RETURN
8000
  COLOR 15, 0: CLS
  LOCATE 12, 25
  PRINT "BU PROGRAMI KULLANDIGINIZ ICIN TESEKKURLER"
  END
2222
  IF ERR = 72 OR 71 OR 70 THEN
    PRINT "Uyari! Bu kütük acilamiyor."
    PRINT "Devam etmek icin disk protectini kontrol ediniz."
    PRINT " Yoksa böyle bir dosya mevcut degil"
    DO: LOOP UNTIL INKEY$ <> ""
```

```
    PRINT "Bu isimde bir dosya bulunamadi."
    DO: LOCATE 20, 10: PRINT "Devam icin bir tusa basiniz..."
    LOOP UNTIL INKEY$ <> ""
    RESUME 200
END IF
```

```
SUB MENU1 (x%, y%, SECIMSAYISI, SECILEN)
' Ana menu alt programi
    nRD% = 1
    FOR Q% = 1 TO SECIMSAYISI
    YAZ ((x% - 1) + Q%), y%, a$(Q%), 0
    NEXT
    YAZ x%, y%, a$(1), 1
    DO
    DO
    w$ = INKEY$
    LOOP WHILE w$ = ""
    IF LEN(w$) = 1 THEN
        SELECT CASE ASC(w$)
        CASE 13
        SECILEN = nRD%
        EXIT DO
        CASE ELSE
        SOUND 200, 1
        END SELECT
    ELSE
        ww$ = RIGHT$(w$, 1)
        SELECT CASE ASC(ww$)
        CASE 80
        YAZ ((x% - 1) + nRD%), y%, a$(nRD%), 0
        nRD% = nRD% + 1: IF nRD% > SECIMSAYISI THEN nRD% = 1
        YAZ ((x% - 1) + nRD%), y%, a$(nRD%), 1
        CASE 72
        YAZ ((x% - 1) + nRD%), y%, a$(nRD%), 0
        nRD% = nRD% - 1: IF nRD% < 1 THEN nRD% = SECIMSAYISI
        YAZ ((x% - 1) + nRD%), y%, a$(nRD%), 1
        CASE ELSE
        SOUND 200, 1
        END SELECT
    END IF
    LOOP
    COLOR 7, 0
END SUB
```

```
SUB SOLVE (sol, var, cons, opt, it)
SHARED a(), tdd(), bb$(), km(), aa$(), mat()
DIM aa(160), bb(160)
' Cozum alt programi
    m = 0: n = 0
    IF sol = 1 THEN
        CLS
        COLOR 15, 2, 7: LOCATE 12, 32: PRINT "SINIRSIZ COZUM"
        DO: LOOP UNTIL INKEY$ <> ""
        COLOR 15, 1, 0: CLS
    END IF
    IF sol = 2 THEN
        CLS
        COLOR 15, 2, 7: LOCATE 12, 12: PRINT "MUMKUN COZUM YOK"
        DO: LOOP UNTIL INKEY$ <> ""
        COLOR 15, 1, 0: CLS
    END IF
    IF sol = 4 THEN
        IF opt = 1 THEN
            opt$ = "max"
        ELSE
            opt$ = "min"
        tdd(0) = -tdd(0)
        END IF
        FOR k = 1 TO cons
          IF a(k, var + 1) = 2 THEN m = m + 1: ' <>0 veya =2
        NEXT k
        FOR i = 1 TO var + m
          IF km(i) = 0 OR km(i) > 0 THEN
            aa(i) = km(i)
          END IF
        NEXT i
        j = 0
        FOR k = 1 TO cons
          IF a(k, var + 1) <> 0 THEN
            j = j + 1: aa$(var + m + k) = "z" + STR$(j)
            aa(var + m + k) = mat(0, k)
          ELSE
            n = n + 1
            aa$(var + m + k) = "y" + STR$(n)
            aa(var + m + k) = mat(0, k)
          END IF
        NEXT k
        FOR i = 1 TO var + m + cons
          FOR j = 1 TO cons
            IF aa$(i) = bb$(j) THEN
              aa(i) = 0: bb(i) = tdd(j)
              EXIT FOR
            ELSE
              bb(i) = 0
            END IF
          NEXT j
        NEXT i
        sf = INT((var + cons + m) / 30)
        IF sf * 30 = var + cons + m THEN syf = sf ELSE syf = sf + 1
        p = 1
        FOR l = 1 TO syf
          CLS
          LOCATE 1, 2: PRINT CHR$(201): LOCATE 1, 3
          PRINT STRING$(77, 205): LOCATE 1, 80: PRINT CHR$(187)
          LOCATE 2, 2: PRINT CHR$(186): LOCATE 2, 80: PRINT CHR$(186)
```

```
LOCATE 3, 2: PRINT CHR$(204)
LOCATE 3, 3:  PRINT STRING$(12, 205)
LOCATE 3, 15: PRINT CHR$(203)
LOCATE 3, 16: PRINT STRING$(12, 205)
LOCATE 3, 28: PRINT CHR$(203)
LOCATE 3, 29: PRINT STRING$(12, 205)
LOCATE 3, 41: PRINT CHR$(203)
LOCATE 3, 42: PRINT STRING$(12, 205)
LOCATE 3, 54: PRINT CHR$(203)
LOCATE 3, 55: PRINT STRING$(12, 205)
LOCATE 3, 67: PRINT CHR$(203)
LOCATE 3, 68: PRINT STRING$(12, 205)
LOCATE 3, 80: PRINT CHR$(185)
LOCATE 4, 2:  PRINT CHR$(186)
LOCATE 4, 15: PRINT CHR$(186)
LOCATE 4, 28: PRINT CHR$(186)
LOCATE 4, 41: PRINT CHR$(186)
LOCATE 4, 54: PRINT CHR$(186)
LOCATE 4, 67: PRINT CHR$(186)
LOCATE 4, 80: PRINT CHR$(186)
LOCATE 5, 2: PRINT CHR$(204)
LOCATE 5, 3: PRINT STRING$(12, 205)
LOCATE 5, 15: PRINT CHR$(206)
LOCATE 5, 16: PRINT STRING$(12, 205)
LOCATE 5, 28: PRINT CHR$(206)
LOCATE 5, 29: PRINT STRING$(12, 205)
LOCATE 5, 41: PRINT CHR$(206)
LOCATE 5, 42: PRINT STRING$(12, 205)
LOCATE 5, 54: PRINT CHR$(206)
LOCATE 5, 55: PRINT STRING$(12, 205)
LOCATE 5, 67: PRINT CHR$(206)
LOCATE 5, 68: PRINT STRING$(12, 205)
LOCATE 5, 80: PRINT CHR$(185)
FOR i = 6 TO 20
   IF p + (l - 1) * 15 = cons + var + m + 1 THEN EXIT FOR
   LOCATE i, 2: PRINT CHR$(186): LOCATE i, 80: PRINT CHR$(186)
   LOCATE i, 15: PRINT CHR$(186): LOCATE i, 28: PRINT CHR$(186)
   LOCATE i, 41: PRINT CHR$(186): LOCATE i, 54: PRINT CHR$(186)
   LOCATE i, 67: PRINT CHR$(186)
   LOCATE i, 3
   PRINT p + (l - 1) * 15; " "; aa$(p + (l - 1) * 15)
   LOCATE i, 16: PRINT bb(p + (l - 1) * 15)
   LOCATE i, 29: PRINT aa(p + (l - 1) * 15)
   LOCATE i, 42: PRINT p + l * 15; " "; aa$(p + l * 15)
   LOCATE i, 55: PRINT bb(p + l * 15)
   LOCATE i, 68: PRINT aa(p + l * 15)
   p = p + 1
NEXT i
LOCATE i, 2: PRINT CHR$(200)
LOCATE i, 3: PRINT STRING$(12, 205)
LOCATE i, 15: PRINT CHR$(202)
LOCATE i, 16: PRINT STRING$(12, 205)
LOCATE i, 28: PRINT CHR$(202)
LOCATE i, 29: PRINT STRING$(12, 205)
LOCATE i, 41: PRINT CHR$(202)
LOCATE i, 42: PRINT STRING$(12, 205)
LOCATE i, 54: PRINT CHR$(202)
LOCATE i, 67: PRINT CHR$(202)
LOCATE i, 55: PRINT STRING$(12, 205)
LOCATE i, 80: PRINT CHR$(188)
LOCATE i, 68: PRINT STRING$(12, 205)
```

```
            LOCATE 2, 5: PRINT "SONUC TABLO"
            LOCATE 2, 20: PRINT "iterasyon:"; it
            LOCATE 2, 40: PRINT "f"; opt$; "="; tdd(0)
            LOCATE 2, 68: PRINT "Sayfa:"; I
            LOCATE 4, 3: PRINT "Degisken"
            LOCATE 4, 16: PRINT "cözüm"
            LOCATE 4, 29: PRINT "firs. mal."
            LOCATE 4, 42: PRINT "Degisken"
            LOCATE 4, 56: PRINT "cözüm"
            LOCATE 4, 68: PRINT "firs. mal."
            DO: LOOP UNTIL INKEY$ <> ""
        NEXT I
        DO: LOOP UNTIL INKEY$ <> "": COLOR 15, 1, 0: CLS
    END IF
END SUB
```

```
SUB YAZ (x%, y%, e$, e)
    IF e = 1 THEN
        COLOR 0, 15
    ELSE
        COLOR 15, 0
    END IF
    LOCATE x%, y%: PRINT e$
END SUB
```

KAYNAKLAR

1. Serper, Ö. ve Gürsakal, N., 1982, Doğrusal Programlama. BİTİA İşletme Fakültesi, Bursa, s45-56.
2. Kaçtıoğlu, S., 1987, Doğrusal Programlama ve Ulaştırma Modeli. Atatürk Üniversitesi İktisadi ve İdari Bilimler Fakültesi Araştırma Merkezi Ders Notları:141, Erzurum, s37-108.
3. Kara, İ., 1980, Yöneylem Araştırması. Anadolu Üniversitesi Basımevi, Eskişehir, s127-132.
4. Gottfried, B. S. and Wefsman, J.,1973, Introduction to Optimization Theory. Prentice-Hall, p151.
5. Dantzig, G. B. and Orchard Hays, W., 1953, Alternate Algorithm for the Revised Simplex Method. The RAND Corporation, Santa Monica, Calif, RAND Report RM-1268.
6. Dantzig, G. B., 1951, Maximization of a Linear Function of Variables Subject to Linear Inequalities. john Wiley&Sons, New York, p359-373.
7. Charnes, A., 1952, Optimality and Degeneracy in Linear Programming. Econometrica, New York, vol20.
8. Wolfe, P.,1962, A technique for Resolving Degeneracy in Linear Programming. The RAND Corporation, Santa Monica, Calif, RAND Report RM-2995.
9. Dantzig, G. B., Orden, A. and Wolfe, P.,1954, Generalized Simplex Method for Minimizing a Linear form under Linear İnequality Restraints. The RAND corporation, Santa Monica, Calif, RAND Report-1264.
10. Hoffman, A. J., 1955, How to Solve a Linear Programming Problem. Directorate of Manegement Analysis, Wasingtan, p124.
11. Eskicioğlu, A. M., 1988, Pascal ile Yapısal Programlama, Evrim Basım Yayın, Ankara, s12.
12. Kaçtıoğlu, S.,1995, Basic Programlama Dili, Erzurum Kültür ve Eğitim Vakfı Yayınları, Erzurum, s10-60.

13. Sezginman, İ.,1993, Lineer Programlama Teori ve Problemleri. Yıldız Teknik Üniversitesi, İstanbul, s231.

14. Bazaraa, M. S.,1977, Linear Programming and Network Flows. John Wiley&Sons, Canada, p194-195.