KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK BOYUTLU STOK KESME PROBLEMİ ve YENİ BİR ÇÖZÜM YAKLAŞIMI

YÜKSEK LİSANS

Bilgisayar Müh. Durdu Mehmet ÖZKAN

Anabilim Dalı : Bilgisayar Mühendisliği

Danışman : Doç. Dr. Yaşar BECERİKLİ

KOCAELİ, 2008

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

ÇOK BOYUTLU STOK KESME PROBLEMİ VE YENİ BİR ÇÖZÜM YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Durdu Mehmet ÖZKAN

Tezin Enstitüye Verildiği Tarih: 26 Mayıs 2008

Tezin Savunulduğu Tarih: 07 Temmuz 2008

Tez Danışmanı
Doç.Dr. Yaşar BECERİKLİ

Üye
Doç.Dr. Adnan KAVAK

Üye
Doç.Dr. Aydoğan SAVRAN

KOCAELİ, 2008

ÖNSÖZ

Kaynakların her geçen gün azaldığı günümüzde optimizasyonu önemi ivmeli bir artış göstermektedir. Rekabet şartlarından dolayı tüm işletmeler kaynaklarını verimli kullanarak maliyetlerini azaltma yolunda adımlar atmaktadır.

Bu çalışmada, cam, ağaç, metal endüstrilerinde çok sık karşılaşılan iki boyutlu stok kesme problemine yeni bir yaklaşımla çözüm aranmıştır. Çalışma sonucunda mevcut yöntemleri kullanan bir yazılımla, yerleşim süreleri ve yerleşim yüzdeleri bakımından karşılaştırmalar yapılmıştır.

Bana bu konuda çalışma fikrini veren, çalışmam süresince fikir ve yapıcı eleştirileriyle benden desteğini esirgemeyen değerli hocam Sayın Doç.Dr.Yaşar BECERİKLİ'ye teşekkür ederim.

Ayrıca çalışmam ve tüm hayatım boyunca hep yanımda olup beni destekleyen değerli aileme ve çalışmamda yardımlarını esirgemeyen tüm arkadaşlarıma teşekkür ederim.

Mayıs 2008, KOCAELİ Durdu Mehmet ÖZKAN

İÇİNDEKİLER

ÖNSÖZ i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iv
TABLOLAR DİZİNİ v
SİMGELER vi
ÖZET vii
ABSTRACT viii

BÖLÜM 1. GİRİŞ 1

BÖLÜM 2. STOK KESME PROBLEMİ KLASİK ÇÖZÜM YÖNTEMLERİ 5
2.1. Giriş 5
2.2. Lineer Programlama 5
2.2.1. Lineer programlama ile ilgili çeşitli varsayımlar ve tanımlar 6
2.2.1.1. Doğrusallık varsayımı 6
2.2.1.2. Toplanabilirlik varsayımı 7
2.2.1.3. Bölünebilirlik varsayımı 7
2.2.1.4. Kesinlik varsayımı 7
2.2.2. Bir lineer programlama problemi nedir? 9
2.3. Simpleks Metodu 13
2.3.1. Simpleks yönteminin uygulama alanları 14
2.3.2. Simpleks yönteminin uygulanması 15
2.3.3. Problemlerin standart forma getirilmesi 15
2.3.3.1. Kısıtlamalar 16
2.3.3.2. Değişkenler 17
2.3.3.3. Amaç fonksiyonu 17
2.3.3.4. Problemin standart forma getirilmesi üzerine genel bir örnek 18
2.3.4. Simpleks metodu ile çözüm 18
2.3.4.1. Problemin standart forma getirilmesi 20
2.3.4.2. Tablonun oluşturulması ve çözümü 21
2.3.4.3. Büyük M metodu 24
2.4. Tamsayılı Programlama 36
2.4.1. Tamsayılı programlamanın formülasyonu 36
2.4.2. Tamsayılı programlama ile ilgili çeşitli örnekler 37
2.4.3. Tamsayılı programlamanın çözümü 41
2.4.3.1. Grafiksel çözüm yöntemi 41
2.4.3.2. Gomory'nin kesme düzlemi yöntemi 41
2.4.3.3. Dal sınır yöntemi 42
2.4.3.4. Tamsayı olmayan çözümün yuvarlanması yöntemi 47

BÖLÜM 3. STOK KESME PROBLEMLERİ ve YENİ BİR ÇÖZÜM YAKLAŞIMI 48
3.1. Giriş 48
3.2. Tek Boyutlu Stok Kesme Problemi 49
3.3. İki Boyutlu Stok Kesme Problemi 51
3.4. Uygulama 52
3.5. Yeni Çözüm Yöntemi 53

BÖLÜM 4. PERFORMANS DEĞERLENDİRMESİ 63
4.1. Yeni Çözüm Yaklaşımının İşlem Külfeti 63
4.2. SmartCutPro Yazılımı İle Karşılaştırılması 64
4.3. Yeni Çözüm Yaklaşımı ve Simpleks Yöntemi ile Stok Kesme Problemi Çözümüne Monte Carlo Simülasyonunun Uygulanması 73
SONUÇLAR ve ÖNERİLER 78
KAYNAKLAR 79
ÖZGEÇMİŞ 82

ŞEKİLLER DİZİNİ

Şekil 2.1: Simpleks maksimizasyon metodunun akış şeması 35
Şekil 2.2: Tamsayılı Programlama Dal Sınır Yöntemi Adım 1 44
Şekil 2.3: Tamsayılı Programlama Dal Sınır Yöntemi Adım 2 45
Şekil 2.4: Tamsayılı Programlama Dal Sınır Yöntemi Adım 3 45
Şekil 2.5: Tamsayılı Programlama Dal Sınır Yöntemi Adım 4 46
Şekil 3.1: Tek Boyutlu Stok Kesme Problemi 49
Şekil 3.2: İki Boyutlu Stok Kesme Problemi 51
Şekil 3.3: İki Boyutlu Stok Kesme Problemi yerleşim örneği 51
Şekil 3.4: Üzerinde kesme işlemi yapılacak kaynak stok 55
Şekil 3.5: Kaynak stoğa ilk adımda yapılan yerleşim. 57
Şekil 3.6: Adım 2 sonucunda kaynak stoğa yapılan yerleşim. 60
Şekil 3.7: Adım 3 sonucunda kaynak stoğa yapılan yerleşim. 61
Şekil 3.8: Tüm adımlar tamamlandığında kaynak stoğa yapılan yerleşim. 62
Şekil 4.1: SmartCutPro kaynak ve sipariş ürünlerin tanımlama ekranı 64
Şekil 4.2: SmartCutPro ile yatay yönde hızlı modda tarama yerleşim sonucu ekranı 65
Şekil 4.3: SmartCutPro ile yatay yönde hızlı modda tarama kalan stok durumu ekranı 65
Şekil 4.4: SmartCutPro ile dikey yönde hızlı modda tarama yerleşim sonucu ekranı 66
Şekil 4.5: SmartCutPro ile dikey yönde hızlı modda tarama kalan stok durumu ekranı 66
Şekil 4.6: SmartCutPro ile yatay+dikey yönde hızlı modda tarama yerleşim sonucu ekranı 67
Şekil 4.7: SmartCutPro ile yatay+dikey yönde hızlı modda tarama kalan stok durumu ekranı 67
Şekil 4.8: SmartCutPro ile yatay yönde yavaş modda tarama yerleşim sonucu ekranı 68
Şekil 4.9: SmartCutPro ile yatay yönde yavaş modda tarama kalan stok durumu ekranı 68
Şekil 4.10: SmartCutPro ile dikey yönde yavaş modda tarama yerleşim sonucu ekranı 69
Şekil 4.11: SmartCutPro ile dikey yönde yavaş modda tarama kalan stok durumu ekranı 69
Şekil 4.12: SmartCutPro ile yatay+dikey yönde yavaş modda tarama yerleşim sonucu ekranı 70
Şekil 4.13: SmartCutPro ile yatay+dikey yönde yavaş modda tarama kalan stok durumu ekranı 70
Şekil 4.14: Geliştirilen yeni yöntem ile yatay yönde tarama yerleşim sonucu ekranı 71
Şekil 4.15: Geliştirilen yeni yöntem ile dikey yönde tarama yerleşim sonucu ekranı 71
Şekil 4.16: Tarama zaman grafiği 72
Şekil 4.17: Tarama yerleşim yüzdesi grafiği 72
Şekil 4.18: Monte Carlo simülasyonu tarama zaman grafiği 76
Şekil 4.19: Monte Carlo simülasyonu tarama yerleşim yüzdesi grafiği 76

TABLOLAR DİZİNİ

Tablo 2.1: İç ve Dış cephe boya üretimi için gereken hammadde miktarları ve kullanılabilecek maksimum hammadde miktarları 19
Tablo 2.2: Başlangıç Simpleks Tablosu 21
Tablo 2.3: Simpleks Tablosu Adım 1 22
Tablo 2.4: Simpleks Tablosu Adım 2 22
Tablo 2.5: Simpleks Tablosu Adım 23
Tablo 2.6: Sonuç Simpleks Tablosu 24
Tablo 2.7: Büyük M Metodu Adımları Tablosu 26
Tablo 3.1: Kesme işleminin yapılacağı sipariş ürünlerin listesi 55
Tablo 3.2: Yerleşim taramasının yapılacağı hedef uzay matrisi 56
Tablo 3.3: Adım 2 için yerleşim taramasının yapılacağı hedef uzay matrisi 58
Tablo 4.1: Kesme işleminin yapılacağı sipariş ürünlerin listesi 63
Tablo 4.2: Monte Carlo simülasyonu sonuçları 74

SİMGELER

LP : Lineer Programlama
TP : Tamsayılı Programlama

ÇOK BOYUTLU STOK KESME PROBLEMİ ve YENİ BİR ÇÖZÜM YAKLAŞIMI

Durdu Mehmet ÖZKAN

Anahtar Kelimeler : Stok Kesme, Lineer Programlama, Tamsayı Programlama

Özet : Bu çalışmada, lineer programlama ve tamsayı programlama yöntemlerinden ve uygulama alanlarından bahsedilerek stok kesme problemine nasıl uygulandığı anlatılmıştır. Stok kesme probleminin klasik çözüm yöntemlerinden lineer programlama ve tamsayılı programlama tekniklerinden bahsedilmiştir. Uygulamada iki boyutlu stok kesme probleminin çözümünde yeni bir yaklaşım kullanılarak klasik yöntemlere göre performansı incelenmiştir. Uygulamada kaynak stokların ökçüleri ve miktarları ile hedef stokların ölçüleri ve miktarları girilerek en az fire verecek şekilde ve en hızlı biçimde çözümü elde edip ekranda yerleşimlerin çizdirilmesi amaçlanmıştır.

MULTIDIMENSIONAL STOCK CUTTING PROBLEM AND A NEW SOLUTION APPROACH

Durdu Mehmet ÖZKAN

Keywords : Stock Cutting, Linear Programming, Integer Programming.

Abstract : In this thesis, linear programming and integer programming methods and their application areas are discussed and this method is implemented to stock cutting problem. One of the traditional solution method of stock cutting problem that linear programming and integer programing methods are discussed. In the application a new approach is used on solution of two dimensional stock cutting problem and performance of new method is compared to traditional methods. The aim of application is to obtain minimal leakage with fastest solution by using sizes and quantities of resource and target stocks and plotting the layout on screen.

BÖLÜM 1. GİRİŞ

İnsanoğlu yüzyıllardır bilinçli ya da bilinçsiz olarak yaptığı işlerin tümünde her zaman en iyi yi yapmayı planlamış ve istemiştir. Bu çabalar, bazen bir işçinin belirli bir kuvvetle maksimum yükü kaldırabilmesi, bazen de uçaklardaki sürtünmenin minimum a indirgenmesi olarak karşımıza çıkagelmiştir. Yadsınamaz bir gerçektir ki, her zaman bir eniyileme çabası var olmuştur. Ünlü matematikçi Leonhard Euler'in ifadesiyle "Doğada maksimum ya da minimum duyusunun bulunmadığı hiç bir olay yoktur" (Çetin, 2004).

Bir matematik sözlüğü optimizasyon (eniyileme) kavramını, "bir probleme en iyi mümkün çözüm bulma süreci olarak" tanımlamaktadır. Matematikte, bu süreç genellikle bir fonksiyonun değerinin verilen kısıtlar altında maksimize ya da minimize edilmesinden oluşur (Clapham, 1996).

Bir işin en iyi yolun seçilerek başarılması fikri uygarlık tarihi kadar eskidir. Örneğin, Yunan tarihçisi Herodotus'a göre, Mısırlılar Nil nehrinin her yıl taşması sonucu arazi sınırlarının yeniden belirlenmesi ve yeni sınırlara göre vergilendirme işleminin en iyi yolla yapılabilmesi için çaba sarfetmişlerdir. Bu çabalar, ölçme ve karar verme aracı olarak düzlem geometrisinin temel kavramlarının oluşturulmasına yol açmıştır, Gottfried ve Weisman (1973). Mısırlılar, Nil nehrinin bahar dönemlerindeki yıllık taşmalarında nehir kıyısından toplu halde uzaklaşıp sular çekildiğinde yine büyük topluluklar halinde geri dönüyorlardı. Çekilme işlemi çok kısa sürede yapılamamaktaydı. Bunun için günlerce önceden halk uyarılmalıydı. Bu amaçla, Mısırlılar en iyi çekilme zamanını hesaplayabilmek için bir tür takvim bile geliştirmişlerdi. Söz konusu takvimi de sayma ve geometri konusundaki birikimlerini kullanarak yapmışlardı (Cooke, 1997).

Newton ve Leibniz tarafından Kalkülüs'ün (Calculus) 17. yüzyılda geliştirilmesi optimizasyon teorisinin gelişiminde önemli bir kilometre taşı olmuştur. Kalkülüs,

hem matematiksel bir fonksiyonun hem de fonksiyon oluşturabilen bağımsız değişkenlerin maksimum veya minimum cinsinden optimal koşullarının elde edilmesine olanak sağlamaktadır. Kalkülüs'ün kullanımı düzgün-davranışlı fonksiyonlarla sınırlandırılmıştır. Ancak, Kalkülüs uygulamalarında karşılaşılan cebirsel problemlerin çözümü bazen güç olabilmektedir. Dolayısıyla, Kalkülüs pragmatik anlamda gerçek dünya problemlerinin optimizasyonunda yeterli ve güçlü bir araç olamamaktadır, Gottfried ve Weisman (1973).

J.L. Lagrange'ın 1788 yılında Lagrange çarpanları yöntemini bilim dünyasının hizmetine sunması önemli bir adım olmuştur. 1939'da W. Karush'un kısıtlandırılmış problemler için optimallik koşullarını bulması optimizasyon teorisinde yeni bir atılım olmuştur. II. Dünya Savaşı'nın başlamasıyla 1942'de İngiltere ve Amerika Birleşik Devletleri'nin Yöneylem Araştırması gruplarını oluşturması optimizasyon dünyası için bir dönüm noktası olmuştur. Sezgisel optimizasyon araçlarından olan yapay sinir ağları 1943'de W. McCulloch ve W. Pitts tarafından çalışıldı. Ertesi yıl ise, J. Von Neumann ve O. Morgenstern tarafından "Oyun Teorisi ve Ekonomik Davranış" adlı eserle oyun kuramı tanıtıldı (Gass, 2002).

II. Dünya Savaşı'ndan sonra yeni sınıf optimizasyon teknikleri geliştirildi. Sözkonusu teknikler daha karmaşık problemlere başarıyla uygulandı. Bunda, yüksek hızlı dijital bilgisayarların geliştirilmesi ve optimum değerlerin elde edilmesi için nümerik tekniklere matematiksel analizin uygulanması son derece etkili olmuştur. Nümerik teknikler Kalkülüs'ün bir takım zorluklarını ortadan kaldırmıştır, Gottfried ve Weisman (1973).

Lineer programların çözümü için Simpleks yöntem 1947'de G.B. Dantzig tarafından geliştirildi. Bu, optimizasyon dünyasında gerçekten bir devrim sayılmaktadır. R. Bellman 1950'de dinamik programlama modelini ve çözümünü geliştirdi. 1951'de H. Kuhn ve A. Tucker daha önce Karush'un önerdiği kısıtlandırılmış problemler için optimallik koşullarını tekrar formüle ederek doğrusal olmayan programlama modelleri üzerinde çalıştılar. Yine aynı yıl, J. Von Neumann, G. Dantzig ve A. Tucker primal-dual lineer programlama modellerini geliştirdiler. Yine önemli bir katkı 1955'de stokastik programlama adı altında G. B. Dantzig tarafından yapıldı.

Kuadratik programlama 1956'da M. Frank ve P. Wolfe tarafından geliştirildi. 1958'deki önemli bir katkı R. Gomory tarafından tamsayılı programlama olarak adlandırıldı. A. Charnes ve W. Cooper şans kısıtlı programlama modellerini 1959'da optimizasyon dünyasına armağan ettiler. 1960'da sezgisel optimizasyon araçlarından birisi olan yapay zeka ve yöneylem araştırması ilişkilerini içeren çalışmalar yapıldı. Hedef programlama modeli yine A. Charnes ve W. Cooper tarafından 1965 yılında geliştirildi. 1975'de çok amaçlı karar verme teorisinin temelleri M. Zeleny, S. Zionts, J. Wallenius, W. Edwards ve B. Roy tarafından atıldı. L. Khachian lineer programlama modellerinin çözümü için farklı bir algoritma olan elips yöntemini 1979'da geliştirildi. 1984'te, N. Karmarkar lineer programlama için alternatif bir çözüm algoritması olan içnokta algoritmasını geliştirdi. 1992'de J.H. Holland tarafından bir sezgisel optimizasyon tekniği olarak kabul edilen genetik algoritma geliştirildi. Çağdaş optimizasyon dünyasında da her geçen gün artan bir ivmeyle önemli katkılar yapılmakta ve bilimin hizmetine sunulmaktadır (Gass, 2002).

Stok kesme problemi endüstriyel alanlarda artıkların minimize edilmesi ve kaynakların verimli kullanılması amacıyla yaygın bir kullanım alanı bulmaktadır. Stok kesme problemleri, 1 boyutlu ve 2 boyutlu olmak üzere ikiye ayrılmaktadır.

1 boyutlu stok kesme problemleri birbirinden farklı uzunluktaki nesneleri en optimum yerleştirmeyi sağlayarak, en az artık bırakacak şekilde uzun kaynak nesnelerden çıkarma işlemidir.

2 boyutlu stok kesme problemi, birbirinden farklı boyutlardaki nesneleri en optimum yerleştirmeyi sağlayarak, en az artık alan bırakacak şekilde büyük boyutlu nesnelerden çıkarma işlemidir. Bu problemin çözümü seri üretim yapan sektörlerde, maliyetleri düşürmek ve verimi arttırmak için çok değerlidir. Stok kesme probleminin hedefi müşteri taleplerine göre en optimum planı yaparak 2 boyutlu bir parçadan talebe göre malzeme çıkarmaktır.

Bu tez beş bölümden oluşmaktadır. Birinci bölümde optimizasyon, Lineer Programlama, Tamsayı Programlama hakkında genel bilgi verilmiş ve uygulamanın amacı hakkında genel bilgi verilerek tezin genel bir tanımı yapılmıştır.

İkinci bölümde, klasik çözüm yöntemlerinden Lineer Programlamanın temel tanımlaması yapılarak uygulama alanlarıyla birlikte özellikleri ve problem türleri hakkında detaylı bilgi verilmiş, Simplek metodu ve aşamaları hakkında detaylı bilgi verilerek bir Lineer Programlama probleminin çözüm adımlarıyla ilgili bilgi verilmiş ve tamsayı programlama temel tanımlaması yapılarak uygulama alanlarıyla birlikte özellikleri ve problem türleri hakkında detaylı bilgi verilmiş, Lineer Programlama problemlerinden farkı ve çözüm yolları hakkında bilgi verilmiştir.

Üçüncü bölümde, stok kesme probleminin ne olduğu tanıtılmış ve bir boyutlu ve iki boyutlu stok kesme problemi tanıtılmış ve stok kesme problemi için yeni geliştirilen çözüm yöntemi detaylarıyla anlatılmıştır.

Dördüncü bölümde klasik yöntemlerle çözüm arayan yazılımlardan biriyle performansı karşılaştırılmıştır ve klasik yöntemlerle stok kesme problemine çözüm arayan bileşenlerden biriyle Monte Carlo simülasyonu yapılarak performans değerlendirmesi yapılmıştır.

Son bölümde, yapılan çalışmadan elde edilen sonuçlara yer verilerek, bunların daha sonraki çalışmalara nasıl yön verebileceği konusunda bazı önerilere değinilmiştir.

BÖLÜM 2. STOK KESME PROBLEMİ KLASİK ÇÖZÜM YÖNTEMLERİ

2.1. Giriş

Bu bölümde günümüzde, işletme, ekonomi ve muhasebe dallarını en yakından ilgilendiren konulardan bir olan, kaynakların optimal dağılımını elde etmeye, maliyetleri minimize, karı ise maksimize etmeye yarayan bir teknik olan lineer programlamadan, lineer programlama problemlerinin çözüm tekniklerinden ve ilk kez 1947 yılında G.B. Dantzig tarafından geliştirilerek ABD Hava Kuvvetlerinin planlanmasında ve daha sonra Charnes, Cooper ve diğerleri tarafından ekonomik analizler için kullanılan simpleks algoritmasından ve bulunan sayısal sonucun tamsayı değerler alması gerektiği durumlardaki sorunları çözmede kullanılan tamsayılı programlamaden bahsedilecektir.

2.2. Lineer Programlama

Günümüzde, işletme, ekonomi ve muhasebe dallarını en yakından ilgilendiren konulardan bir olan Lineer Programlama, aynı zamanda yöneylem araştırmasında da en önemli konulardan biridir. Lineer Programlama, kaynakların optimal dağılımını elde etmeye, maliyetleri minimize, karı ise maksimize etmeye yarayan bir tekniktir.

Lineer Programlama, optimizasyon problemlerinin çözümünde kullanılan bir yöntemdir. 1947'de, George Dantzig, Lineer Programlama problemlerinin çözümünde kullanılan etkin bir yol olan Simpleks Algoritma' yı buldu ve bu buluşla birlikte Lineer Programlama, sıklıkla ve hemen hemen her sektörde kullanılmaya başlandı. Özellikle bankacılık, eğitim sektörlerinde ve askeriyede, optimizasyon problemlerinin çözümünde Lineer Programlama, çok defa kullanılmıştır ve kullanılmaya devam edilmektedir. Firmalarda karşılaşılacak darboğazların

giderilmesinde, seçenekli üretim tekniklerinin kullanılmasının getirilerini belirlemede, kıt kaynakların etkin kullanımında ve bunların gölge fiyatlarının belirlenmesinde kullanılacak en uygun politikaların belirlenmesinde kullanılan bir tekniktir. Fortune 500' e üye firmalar arasında yapılan bir araştırma sonucunda, bu firmaların % 85' inin Lineer Programlama yöntemini kullandığı öğrenilmiştir. Lineer Programlama o kadar önemlidir ki, Yöneylem Araştırması kitaplarının çok büyük bir kısmını tek başına kaplar. (Öztürk, 2001)

Lineer Programlama, değişkenlere ve kısıtlara bağlı kalarak amaç fonksiyonunu en uygun (maksimum ya da minimum) kılmaya çalışır. Temel olarak, Lineer Programlama, kıt kaynakların optimum şekilde dağılımını içeren deterministik bir matematiksel tekniktir. (Kobu, 1997)

Lineer programlama modelinden tutarlı sonuçların elde edilmesi çeşitli varsayımlara bağlıdır.

2.2.1. Lineer programlama ile ilgili çeşitli varsayımlar ve tanımlar

Lineer Programlama modelinden tutarlı sonuçların elde edilebilmesi için aşağıdaki varsayımlar sağlanmalıdır.

2.2.1.1. Doğrusallık varsayımı

Bir Lineer Programlama modelinin amaç fonksiyonunun karar değişkenlerinin bir lineer fonksiyonu olması (doğrusal olması) gerçeğinin iki gerekçesi vardır:

Amaç fonksiyonuna her karar değişkeninden gelen eklemeler karar değişkenlerinin değerleri ile doğru orantılıdır. Örneğin, dört asker yapmayla amaç fonksiyonuna yapılacak katkı (4 * 3$ = 12$), bir asker yapmanın amaç fonksiyonuna yapacağı katkının (3$) tam olarak dört katıdır.

Bir amaç fonksiyonuna bir karar değişkeninin yaptığı katkı, diğer karar değişkenlerinin yaptığı katkıdan bağımsızdır. Örneğin, x_2 nin değeri ne olursa olsun, x_1 askerlerinin üretimi amaç fonksiyonuna her zaman $3x_1$ kadar katkı yapacaktır. (Tulunay, 1990)

2.2.1.2. Toplanabilirlik varsayımı

Bu varsayım değişik üretim faaliyetlerine kaynak olan üretim girdilerinin toplamının her bir işlem için ayrı ayrı kullanılan girdilerin toplamına eşit olduğunu gösterir. Örneğin bir iş iki saatte, diğeri üç saatte yapılıyorsa, iki işi birden yapmak için beş saate gerek vardır. (Tulunay, 1990)

2.2.1.3. Bölünebilirlik varsayımı

Bu varsayım, her karar değişkeninin ondalıklı bir sayı olabilmesine imkan verir. Ama, İnci probleminde bölünebilirlik varsayımına göre 1.5 adet asker ve 1.63 adet tren üretmek mümkün olmalıydı. Ama mümkün değildir. O zaman bu problem bölünebilirlik varsayımını sağlamıyor. Bu tip problemleri çözerken tamsayılı programlama yöntemi kullanılır. (Tulunay, 1990)

2.2.1.4. Kesinlik varsayımı

Bu varsayım, tüm parametrelerin (amaç fonksiyonu katsayısı, sağ el tarafı ve teknolojik katsayı) kesin olarak bilinmesini öngörür. Eğer bu değerler tam olarak bilinmiyorsa, sonuç güvenilir olmayacaktır. (Tulunay, 1990)

Bu aşamada, Lineer Programa ile ilgili Lineer Fonksiyon ve Lineer Eşitsizlik kavramlarını açıklayalım.

Tanım 1: $x_1, x_2, ...x_n$ in bir fonksiyonu olan $f (x_1, x_2, ...x_n)$, sadece ve sadece bir sabitler seti ile birlikte $(c_1, c_2, ...c_n)$ kullanıldığında bir lineer fonksiyondur.

$$f(x_1, x_2, ...x_n) = c_1x_1 + c_2x_2 + ...c_nx_n \quad (2.1)$$

Örneğin, $f(x_1, x_2) = 2x_1 + x_2$, x_1 ve x_2 nin bir lineer fonksiyonudur.
Fakat, $f(x_1, x_2) = x_1x_2$ fonksiyonu x1 ve x2 nin bir lineer fonksiyonu değildir.
Tanım 2: Herhangi bir $f(x_1, x_2, ...x_n)$ lineer fonksiyonu ve herhangi bir b sayısı için,

$$f(x_1, x_2, ...x_n) \leq b \text{ ve } f(x_1, x_2, ...x_n) \geq b \quad (2.2)$$

eşitsizlikleri birer lineer eşitsizliklerdir.

Örneğin, $2x_1 + 3x_2 \leq 3$ ve $3x_1 + x_2 \geq 3$ birer lineer eşitsizliktir. Fakat $x_1x_2 \geq 3$ bir lineer eşitsizlik değildir.

Tanım 3: Bir Lineer Programlama Problemi, aşağıdakilerin gerçekleştirilmesi ile yürütülen bir optimizasyon problemidir:

Karar değişkenlerinin oluşturduğu bir optimizasyon problemini maksimize ya da minimize etmeye çalışır. Maksimize ya da minimize edilmeye çalışılan fonksiyona amaç fonksiyonu denir. Karar değişkenlerinin değerleri bazı kısıtları sağlamalıdır. Her kısıt bir lineer eşitlik ya da lineer eşitsizlik olmalıdır. Bir işaret sınırı, her değişkenle ilgili olarak belirlenmelidir. Herhangi bir xi değişkeni için, bir işaret sınırı belirlenmelidir.

Tanım 4: Tüm uygun çözümler arasında amaç fonksiyonunu iyi karşılayanı optimal çözüme uygun çözüm denir.

Tanım 5: Çözümün bir veya birkaç temel değişkeninin değeri sıfırsa buna bozulan çözüm adı verilir.

2.2.2. Bir lineer programlama problemi nedir?

Belli bir amaca ulaşmak üzere en uygun seçimin yapılmasını sağlayan matematiksel bir yöntemtir. Genellikle günlük yaşamdan alınan bir problem lineer fonksiyon olarak formüle edilir ve amaç fonksiyonu denilen bu fonksiyonun bazı kısıtlamalar altında maksimumu ya da minimumu aranır. Çözüm aranan bölge dışbükey bir kümedir. Bu kümenin her bir noktası problemin olurlu bir çözümü ve fonksiyonu maksimum ya da minimum kılan değer de optimum çözüm adını alır.

Bu aşamada, Lineer Programlama tanıtılacak ve Lineer Programlama problemlerinin anlaşılması ve çözülmesinde yardımcı olacak bazı önemli terimler tanımlanacaktır. Bu tanımlamaların bir örnek üzerinden yürütülmesi anlaşılma açısından yararlı olacaktır.

Örnek;
İnci Oymacılık firması, iki çeşit tahta oyuncak üretmektedir. Bunlar asker ve tren dir. Bir asker, 27$' a satılmaktadır ve bir tanesi için 10$' lık hammadde kullanılmaktadır. Üretilen her asker, firmanın işgücü ve değişken maliyetlerini 14$ kadar arttırmaktadır. Bir tren ise 21$' a satılmaktadır ve 9$' lık hammadde kullanmaktadır. Her bir trenin yapımı, İnci Oymacılık' ın işgücü ve diğer değişken maliyetlerini 10$ arttırmaktadır. Tahta asker ve trenlerin üretiminde iki çeşit, kalifiye işgücü gerekmektedir: oymacılık ve düzeltme. Bir asker, 2 saatlik düzeltme ve 1 saatlik oymacılık işlemleri gerektirmektedir. Bir tren ise, 1 saatlik oymacılık ve 1 saatlikte düzeltme işlemleri gerektirmektedir. İnci Oymacılık firması, hammaddeyi istediği kadar sağlayabilmekteyken işgücünde sise, en çok 100 saatlik düzeltme ve 80 saatlik oymacılık imkanına sahiptir. Bunların yanında, trenlerin talebi piyasada limitsizken her hafta ancak 40 adet asker satılabilmektedir. İnci Oymacılık, haftalık karını maksimize etmek istemektedir. Bu durumu ifade edecek ve firmanın karını maksimize etmede kullanılacak matematiksel modeli kurunuz.

Çözüm: İnci firmasının modeli kurulurken, öncelikle, tüm Lineer Programlama problemleri için geçerli olan karakteristikleri belirlemek gerekir:

Karar Değişkenleri: Herhangi bir Lineer Programlama modelinde, karar değişkenleri, bizim sonucumuzda elde edeceğimiz etkenlerdir. Bu örnekte, İnci firması, her hafta kaç tane asker ve kaç tane tren üreteceğine karar vermelidir. Bu düşünceyle,

x_1= Her hafta üretilecek asker sayısı
x_2= Her hafta üretilecek tren sayısı

şeklinde karar değişkenlerini belirleyebiliriz.

Amaç Fonksiyonu: Herhangi bir Lineer Programlama probleminde, karar verici, karar değişkenlerinin bazı fonksiyonlarını maksimize (geliri ya da karı) veya minimize (maliyeti) etmek ister. Maksimize ya da minimize edilmek istenen fonksiyona amaç fonksiyonu denir. Yukarıdaki problem için, öncelikle, sabit maliyetlerin, x_1 ve x_2 nin alacağı değerlerden bağımsız olduğunu söylemeliyiz. İnci,

Maks [(haftalık gelir) - (hammadde maliyetleri) - (diğer değişken giderler)] (2.3)

Üzerine konsantre olmalıdır.

İnci'nin haftalık gelirleri ve maliyetleri x_1 ve x_2 terimleri kullanılarak ifade edilebilir. İnci'nin, satabileceğinden daha fazla asker üretmesi saçma olacaktır. Bu yüzden, üretilen tüm askerlerin satılabileceğini düşünüyoruz. O zaman,

Haftalık Gelir = Asker satışından elde edilen haftalık gelir + Tren satışından elde edilen haftalık gelir
= (dolar / asker). (asker / hafta) + (dolar / tren) . (tren / hafta)
= $27x_1 + 21x_2$ olacaktır.

Bunun yanında,

Haftalık hammadde maliyeti = $10x_1 + 9x_2$
Diğer haftalık değişken maliyetler = $14x_1 + 10x_2$ olacaktır.

Yani, İnci' nin maksimize etmek istediği,

$(27x_1 + 21x_2) - (10x_1 + 9x_2) - (14x_1 + 10x_2) = 3x_1 + 2x_2$ dir.

Sonuç olarak, İnci' nin amacı, $3x_1 + 2x_2$ ' yi maksimize edecek x_1 ve x_2 değerlerini bulmaktır. Herhangi bir Lineer Programlama probleminin amaç fonksiyonunu simgelemek için z simgesini kullanacağız. İnci' nin amaç fonksiyonu şudur:

$$\text{maks } z = 3x_1 + 2x_2 \qquad (2.4)$$

Amaç fonksiyonundaki bir değişkenin katsayısı amaç fonksiyonu katsayısı diye adlandırılır. Örneğin, x_1' in amaç fonksiyonu katsayısı 3, x_2' nin amaç fonksiyonu katsayısı 2' dir. Bu örnekte (çoğu örnekte olduğu gibi) amaç fonksiyonu katsayısı, her değişkenin, firmanın karına yaptığı katkıdır.

Kısıtlar: x_1 ve x_2 arttıkça, İnci' nin amaç fonksiyonunun değeri de artar. Bunun anlamı, eğer İnci, x_1 ve x_2 için istediği üretim miktarını seçmekte serbest olsa idi, firma x_1 ve x_2 üretimini keyfi olarak yüksek tutar ve karını sürekli arttırırdı. Maalesef, x_1 ve x_2 değerleri, aşağıdaki üç kısıtla sınırlandırılmıştır:

Kısıt 1: Her hafta, 100 saatten fazla düzeltme işçiliği kullanılamamaktadır.
Kısıt 2: Her hafta, 80 saatten fazla oymacılık işçiliği kullanılamamaktadır.
Kısıt 3: Kısıtlı talepten dolayı, her hafta en fazla 40 asker üretilebilmektedir.

Hammadde alımında bir sınır olmadığı varsayılmıştır, bu yüzden böyle bir kısıt yoktur.

Problemdeki bir diğer adım ise bu kısıtları, x_1 ve x_2 (karar değişkenleri) cinsinden göstermektir. Birinci kısıtı x_1 ve x_2 değişkenleri ile göstermek için,

Toplam düzeltme saati / hafta = (düzeltme saati / asker) . (yapılan asker / hafta)
+ (düzeltme saati / tren) , (yapılan tren / hafta)

$$= 2(x_1) + 1(x_2) = 2x_1 + x_2$$

O zaman, 1. kısıt şöyle ifade edilir:

$$2x_1 + x_2 \leq 100 \qquad (2.5)$$

Şuna dikkat edilmelidir ki 2. formüldeki tüm terimler haftalık düzeltme saatleridir. Bir kısıtın geçerli olabilmesi için, tüm terimlerin aynı cinsten olması gerekir. Aksi taktirde kısıt geçersiz olacaktır.

kısıtı belirtirken,
Toplam oymacılık saati / hafta = (oymacılık saati / asker). (asker / hafta)
+ (oymacılık saati / tren) . (tren / hafta)
$= 1(x_1) + 1(x_2) = x_1 + x_2$

Sonuçta, 2. kısıt şöyle yazılır:

$$x_1 + x_2 \leq 80 \qquad (2.6)$$

Yine bu kısıtlamada da tüm birimler aynıdır. (oymacılık saati / hafta)
Son olarak, haftalık asker üretimini 40 adette sınırlamamız gerçeği geliyor, çünkü ancak bu kadarı satılabilecektir. Bu olay da şu kısıtı doğuruyor;

$$x_1 \leq 40 \qquad (2.7)$$

Böylece, problemin kısıtları belirlenmiş oldu. Kısıtlardaki karar değişkenlerinin katsayıları teknolojik katsayılar diye adlandırılırlar. Bunun sebebi, bu katsayıların genellikle değişik ürünlerin üretiminde kullanılacak teknolojiyi belirtmeleridir. Örneğin, (3) deki x_2 nin teknolojik katsayısı 1 saatlik oymacılığın gerekli olduğunu anlatır. Her kısıtın sağındaki sayı ise kısıtların sağ el tarafı olarak adlandırılır. Genellikle bir kısıtın set (sağ el tarafı)' i eldeki kaynakların miktarını söyler.

İşaret Sınırı: Bir Lineer Programlama probleminin formülasyonunu tamamlamak için şu soru, her karar değişkeni için cevaplandırılmalıdır: Karar değişkeni sadece negatif olmayan değerleri mi içerir ya da hem negatif hem de pozitif değerleri kapsar mı?

Eğer bir karar değişkeni (x_i) sadece negatif olmayan değerleri içeriyorsa, işaret sınırı olarak $x_i \geq 0$' ı ekleriz. x_i değişkeni, hem pozitif hem de negatif değerleri içeriyorsa x_i sınırsızdır denir. İnci problemi için $x_1 \geq 0$ ve $x_2 \geq 0$ olduğu açıktır. Başka problemlerde, bazı değişkenler sınırsız olabilirler. Örneğin, x_i bir firmanın nakit balansını gösteriyorsa, x_i , eğer firma borçlu ise negatif olarak düşünülebilir. Böyle bir durumda, x_i' yi sınırsız olarak belirtmek uygundur.

İşaret sınırlarını da ekledikten sonra optimizasyon modelimiz şu duruma gelmiştir:

$$
\begin{array}{lrcrcll}
\text{maks } z & = & 3x_1 & + & 2x_2 & & \\
 & & 2x_1 & + & x_2 & \leq\ 100 & \text{(Düzeltme Kısıtı)} \\
 & & x_1 & + & x_2 & \leq\ 80 & \text{(Oymacılık Kısıtı)} \\
 & & x_1 & & & \leq\ 40 & \text{(Askerlerin Talep Kısıtı)} \\
 & & x_1, & x_2 & \geq & 0 & \text{İşaret sınırları}
\end{array}
\qquad (2.8)
$$

x1 ve x2 nin değerleri, tüm kısıtları sağlayacak şekilde amaç fonksiyonunu maksimize etmelidir.

2.3. Simpleks Metodu

Lineer programlama problemlerini çözmede yaygınca kullanılan simpleks yöntemi ilk kez 1947 yılında G.B. Dantzig tarafından geliştirilerek ABD Hava Kuvvetlerinin planlanmasında kullanılmıştır. Daha sonra Charnes, Cooper ve diğerleri ekonomik analizler için uygulamalı çalışmalar yapmışlardır.

Simpleks Yöntemi cebirsel tekrarlama (iterasyon) işlemine dayanır. Yöntemde önce başlangıç simpleks tablosu düzenlenir sonra tekrarlayıcı işlemler ile belirli bir hesap yöntemi içinde gelişen çözümlere doğru ilerleyerek optimal çözüme ulaşıncaya kadar işlemler sürdürülür. Gelişen çözüm tablolarında amaç fonksiyonunun ve karar değişkenlerinin değişen değerleri gözlenebilir.

Simpleks metodunun bir başka özelliği de her yeni adımın istikrarlı bir biçimde bizi optimum sonuca daha fazla yaklaştırmasıdır. Ve tabii en önemlisi de bu metodun bize optimum sonuca ulaştığımızı bildirebilmesidir.

2.3.1. Simpleks yönteminin uygulama alanları

Simpleks yönteminin bütün Doğrusal Programlama problemlerinde uygulanabilir olmasına karşın, bazı tür problemler başka yöntemler kullanılarak daha etkin bir şekilde çözülebilir. Bunun yanında Doğrusal Programlama dışında başka bir yöntemle çözümlenemeyen problemler de vardır. Burada, çözümlerinde Simpleks yöntemin kullanılması gereken bazı problemlere kısaca değinilecektir.

Bunlar; (Tulunay, 1990)

1. Karışım problemleri: Burada gıda sanayiinde, petrol sanayiinde, metalurjide maliyetleri de göz önüne alarak yapılacak karışımların bünyelenmesinde en uygun yol araştırılır.
2. Optimum üretim programının saptanması: Burada ise sınırlı kaynakların (kapasite, hammadde, işgücü, vs.) optimum kullanımı ile karı maksimize veya maliyetleri minimize etme problemleri söz konusudur.
3. İş ve ücret değerlemesi: Doğrusal Programlama değerleme işleminde açık olarak ilgilenilen faktörler için uygun ağırlıkların saptanması için çoklu korelasyon analizi yerine kullanılabilir.
4. Depolama problemleri: Belirli bir zaman içinde saptanmış depo kapasitesinin karı maksimize (veya maliyetleri minimize) edecek şekilde satışlarını, depolanmasını veya satın almaları saptamada optimum uygulamayı sağlamaya çalışır.
5. Malzeme kullanımını optimize eder: Eseasında bu optimum üreim programının saptanmasının başka bir görünümüdür. Ana konu, standart şekilli hammaddelerin (çelik levhalar, yassı veya yaprak metaller vs.) kayıplar en az olacak şekilde kullanımını sağlamaktır.
6. Uzun dönem planlama çalışmaları: Gelecekteki talepleri karşılayabilmek üzere verimli kapasitenin saptanması, bunun verimli bir program içinde sağlanması ve hatta eldeki optimum çözümlerin duyarlılık analizlerinin, öngörülerdeki hataları, fiyat değişimlerinin de dikkate alarak yapılmasına olanak verir.

7. Yapısal modellerin optimize edilmesi: Yapısal modellerde dayanıklılık, ağırlık ve çevre koşullarını dikkate alarak optimum planın hazırlanmasındaki kullanımlardır.

Bütün bunların yanında, bu yöntemin uygulanabileceği daha değişik problemlere de rastlanabilir. Kaldı ki, yöntem herhangi bir Doğrusal Programlama probleminin çözümünde genel olarak uygulanabilmektedir.

2.3.2. Simpleks yönteminin uygulanması

Simpleks Yönteminde en iyi çözüme ulaşmaya çalışırken çözüm alanı içinde farklı noktalar denenir. Bu deneme sırasında aynı standart hesaplamalar yenilenerek en iyi çözüme erişinceye kadar ardarda çözümler geliştirilir. Simpleks yönteminin bir başka özelliği de, her yeni çözümün en az bir önceki çözüm kadar iyi olmasıdır. Bu önemli özellik, kesinlikle en iyi çözüme yaklaşılmasını sağlar. Sonuçta yöntem, en iyi çözüme erişildiğini kendisi belirler. Bir çok sorun için belirli sayıda yineleme sonunda çözüme ulaşır. Yineleme özelliği ve standart hesaplama yolları nedeniyle bilgisayara uyarlanması oldukça kolaydır, Tütek ve Gümüşoğlu (2000).

Doğrusal Programlama sorununun Simpleks Yöntemi ile çözümü şu biçimde gerçekleştirilebilir:

- Problemin Standart Forma Getirilmesi,
- Sistematik Yordam ve Kurallar İzleyerek Çözüme Erişme.

2.3.3. Problemlerin standart forma getirilmesi

Simpleks yönteme başlamadan önce problemlerin doğru biçimde ifade edilmesi gerekir ve Doğrusal programlama problemlerinin standart şekilde olması için aşağıdaki özellikleri taşıması gerekmektedir:

1. Bütün kısıtlamalar eşitlik olmalıdır.

2. Bütün değişkenler pozitif olmalıdır.
3. Amaç fonksiyonu maksimizasyon ya da minimizasyon olabilir.

2.3.3.1. Kısıtlamalar

Standart formda kısıtlamaların eşitlik olması gerekmektedir. Bunu yapabilmek için eşitsizliğin türüne göre yeni değişkenler kullanılması gerekir.

Her bir "≤" eşitsizliği için negatif olmayan aylak değişkeni (s) eşitsizliğin sağ tarafı ile sol tarafı arasındaki farkı ifade eder, (Öztürk, 2001). Yani böyle bir eşitsizliğin sol tarafına bir aylak değişken eklenerek eşitsizlik, eşitlik haline getirilebilir.

Örnek:

$$x_1 + 2x_2 \leq 6 \quad \Leftrightarrow \quad x_1 + 2x_2 + s = 6 \, , \, s \geq 0 \tag{2.9}$$

Her bir "≥" Eşitsizliği için negatif olmayan artık değişken (s) vardır ki, bunun değeri eşitsizliğin sağ ve sol tarafı arasındaki farkı gösterir. Yani eşitsizliğin sağ tarafına bir değişken eklenerek eşitsizlik, eşitlik haline dönüştürülebilir.

Kısıtlamaların sağ taraflarında sadece değerlerin olması gerekmektedir. Bu nedenle eklenen yeni değişken eşitsizliğin sol tarafından çıkartılır.

Örnek:

$$3x_1 + 2x_2 - 3x_3 \geq 15 \quad \Leftrightarrow \quad 3x_1 + 2x_2 - 3x_3 - s_2 = 15 \, , \, s_2 \geq 0 \tag{2.10}$$

Negatif olmayan aylak değişken kısıtlamada kullanılmayan ve boşa harcanan yada yitirilen kaynakları, artık değişken ise genellikle fazla kaynak veya kapasiteyi gösterir.

Standart formda eşitliğin sağ tarafının pozitif olması gerekmektedir. Eğer eşitliğin sağ tarafında negatif bir değer var ise eşitlik "-1" ile çarpılır.

Örnek:

$$3x_1 + 2x_2 - 3x_3 - s_2 = -15 \quad \Leftrightarrow \quad -3x_1 - 2x_2 + 3x_3 + s_2 = 15 \tag{2.11}$$

2.3.3.2. Değişkenler

Standart formda bütün değişkenlerin sıfır veya sıfırdan büyük, pozitif değerler taşıyacak değişkenler olması gerekmektedir. Eğer problemde sınırlandırılmamış, negatif ya da pozitif değerler alabilecek bir değişken var ise, bu değişken negatif olmayan iki ayrı değişken arasındaki fark olarak tanımlanabilir. Bu durumda optimal simpleks çözümünde değişkenlerden sadece bir tanesi pozitif değer alacaktır.

Örnek:
x1 değişkeni sınırsız ise, $x_1 = x_{11} - x_{12}$ şeklinde bir tanımlama yapılarak iki yeni değişken sisteme eklenir.

2.3.3.3. Amaç fonksiyonu

Her ne kadar standart form her tür amaç fonksiyonu kullanmaya izin veriyorsa da bazı durumlarda maksimizasyon ve minimizasyon arasında dönüşüm yapmak gerekebilir.

Herhangi bir maksimizasyon problemi, minimizasyon problemi olarak veya bunun karşıtı minimizasyon problemi, maksimizasyon problemi olarak işlem görebilir. Böyle bir işlem amaç fonksiyonunda yer alan her katsayının işaretini değiştirmekle sağlanabilir.

Yani amaç fonksiyonu "-1" ile çarpılır.

Örnek:
Maksimum (Z) = $3x_1 + 2x_2 - 3x_3$ matematiksel olarak
Minimum (-Z) = $-3x_1 - 2x_2 + 3x_3$'e eşittir

Böyle işlem gördürülse dahi problemdeki karar değişkenlerinin optimal değerleri aynı olacaktır. Yani amaç fonksiyonunda yer alan karar değişkenlerinin optimal değerleri yalnız işaretleri değişmek üzere aynı kalır

2.3.3.4. Problemin standart forma getirilmesi üzerine genel bir örnek

$$
\begin{aligned}
\min z \quad = \quad & 2x_1 + 3x_2 \\
& x_1 + x_2 = 100 \\
& 2x_1 + 3x_2 \leq -5 \\
& 7x_1 - 4x_2 \leq 6 \\
& x_1 \quad \text{Sınırlandırılmamış} \\
& x_2 \geq 0
\end{aligned}
\tag{2.12}
$$

Bu bilgiler ışığında problemin standart forma getirilmesi şu şekilde olmaktadır: Problemde ikinci ve üçüncü kısıtlamalara atıl değişkenler eklenmeli, sınırsız değişken için iki yeni değişken eklenmelidir. Sağ tarafların pozitif olması istenirse, ikinci kısıtlama "-1" ile çarpılabilir. Problemin son hali:

$$
\begin{aligned}
\min z \quad = \quad & 2x_{11} - 2x_{12} + 3x_2 \\
& x_{11} - x_{12} + x_2 = 10 \\
& -2x_{11} + 2x_{12} + 3x_2 + s_2 = -5 \\
& 7x_{11} - 7x_{12} - 4x_2 + s_3 = 6 \\
& x_{11}, x_{12}, x_2, s_2, s_3 \geq 0
\end{aligned}
\tag{2.13}
$$

2.3.4. Simpleks metodu ile çözüm

Bir eşitlik sisteminde m eşitlik ve n değişken varsa ve değişken sayısı eşitlik sayısından büyükse tam çözüm bulmak matematiksel olarak mümkün değildir. Bu durumda (n–m) tane değişkene herhangi bir değer atanır ve geri kalan n değişkenin değeri bulunur. Simpleks metodunda da bu durumla karşılaşılabilir. Böyle bir durumda n tane değişken temel değişkenler olarak belirlenir ve geri kalan (n–m) tane değişken temel olmayan değişken olur, sıfır değerini alırlar. Bir temel çözümün uygun olabilmesi için bütün temel değişkenlerin değerinin sıfırdan büyük olması gereklidir, (Gürdoğan, 1981).

Simpleks metodu uygun bir çözüm ile başlar ve her aşamada daha iyi bir çözüm bularak ilerler. Sistemli bir şekilde ilerlerken daha iyi bir çözüm bulunamayacak duruma gelindiğinde optimal çözüm bulunmuş olur. Simpleks yönteminin çalışabilmesi için ilk olarak uygun temel çözümün bulunması gerekmektedir. İlk uygun temel çözüme ulaşabilmek için kısıtlamalarda ve amaç fonksiyonunda bazı değişiklikler yapmak durumunda kalınabilir.

Bir Doğrusal Programlama probleminin Simpleks yöntemi ile çözümünün daha iyi anlaşılabilmesi için yöntem aşağıdaki örnek üzerinde anlatılacaktır.

Örnek:
Süper Boya firması, iç ve dış cephe için boya üretmektedir. Boya üretimi için iki adet hammadde kullanılmaktadır. İç ve Dış cephe boya üretimi için gereken hammadde miktarları ve kullanılabilecek maksimum hammadde miktarları aşağıdaki Tablo 3.1'de verilmiştir.

Tablo 2.1: İç ve Dış cephe boya üretimi için gereken hammadde miktarları ve kullanılabilecek maksimum hammadde miktarları

	1 ton boya üretmek için gereken Hammadde (Ton)		En çok kullanılabilecek miktar
	Dış Boya	İç Boya	
Hammadde A	1	2	6
Hammadde B	2	1	8

Piyasa araştırmaları gösteriyor ki iç boya talebi dış boya talebinden en çok bir ton fazla olabilir. Ayrıca iç boya talebi en fazla 2 ton olabilir. Dış boyanın tonu 3.000$, iç boyanın tonu da 2.000$ dır. Firma kazancını maksimum yapmak için iç ve dış boyadan ne kadar üretmelidir?
Çözüme ilk olarak denklemler kurularak başlanmalıdır.

Değişkenler:
x_D = Dış Boya üretim miktarı ve
x_I = İç Boya üretim miktarı olarak tanımlanır.

Amaç fonksiyonu (1000 dolar olarak):

Max z =$3x_D + 2x_İ$

Kısıtlamalar:

$X_D + 2x_İ \leq 6$ (Hammadde A) (2.14)

$2x_D + x_İ \leq 8$ (Hammadde B)

$x_İ - x_D \leq 1$ (İç boya talebi, diğerinden en çok 1 ton fazla)

$x_İ \leq 2$ (Talep en çok 2 ton)

$x_D, x_İ \geq 0$

2.3.4.1. Problemin standart forma getirilmesi

Problem Standart Forma getirildiğinde:

Maks z = $3x_D + 2x_İ + 0s_1 + 0s_2 + 0s_3 + 0s_4$

Kısıtlamalar:

$X_D + 2x_İ + s_1 = 6$ (2.15)

$2x_D + x_İ + s_2 = 8$

$-x_D + x_İ + s_3 = 1$

$x_İ + s_4 = 2$

$x_D, x_İ, s_1, s_2, s_3, s_4 \geq 0$

Simpleks tablosundan amaç fonksiyonunun değeri şöyle hesaplanır:

$Z - 3x_D - 2x_İ = 0$ (2.16)

Tablonun hazırlanmasına geçmeden önce temel değişkenler seçilmelidir. Her eşitlikte bir artık değişken olması ve eşitliklerin sağ taraflarının pozitif olması bulanacak temel çözümün uygun (feasible) olacağını belirtir. Değişken sayısı 6, eşitlik sayısı 4 olduğundan 2 tane değişkeni temel olmayan değişken olarak seçip onlara sıfır değerini atanır. Temel değişkenleri seçerken eşitliklerde katsayısı 1 olanları seçmek kolaylık sağlayacaktır. Bu durumda x_D ve $x_İ$ sıfır kabul edilerek temel çözüm oluşturulur.

2.3.4.2. Tablonun oluşturulması ve çözümü

Simpleks metodu her adımda en çok kazanç sağlayacak değişkenin temel değişkenler grubuna katılmasını ve en az getiri sağlayanın temel değişken grubundan ayrılması esasına göre çalışmaktadır.

Simpleks Tablosu Tablo 2.2'deki gibi oluşturulur.

Tablo 2.2: Başlangıç Simpleks Tablosu

Temel	Z	X_D	X_i	s_1	s_2	s_3	s_4	Sonuç
Z	1	-3	-2	0	0	0	0	0
s_1	0	1	2	1	0	0	0	6
s_2	0	2	1	0	1	0	0	8
s_3	0	-1	1	0	0	1	0	1
s_4	0	0	1	0	0	0	1	2

Tablo oluşturulduktan sonra sıra temel değişkenlere girmesi gereken değişkeni seçmeye gelmektedir. Bu işlemi yaparken z amaç fonksiyonundaki katsayısı en büyük olan değişkeni temel gurubuna almak amaçlanır. Bunun için tabloda z satırına bakılarak katsayısı negatif olan bir değişken aranır. Eğer birden çok negatif katsayılı değişken varsa içlerinden en küçük katsayılı olanı giriş değişkeni olarak seçilir.

Giriş değişkeni belirlendikten sonra sıra temel değişkenlerden çıkacak değişkeni belirlemeye geliyor. Bunu yapmak için de ayrıldığında en az değer azalışına sebep olacak değişken aranır. Tabloyu kullanarak bunu yapmak için sonuç sütunundaki değerleri, giren değişken sütunundaki değerlere bölerek, en ufak olanı seçilir. Bu bölme işleminde giren değişkenin sütunundaki negatif ve sıfır değerler işleme katılmaz.

Tablo 2.3: Simpleks Tablosu Adım 1

Temel	Z	X_D	$X_İ$	s_1	s_2	s_3	s_4	Sonuç	
Z	1	-3	-2	0	0	0	0	0	
s_1	0	1	2	1	0	0	0	6	6/1=6
s_2	0	2	1	0	1	0	0	8	8/2=4
s_3	0	-1	1	0	0	1	0	1	
s_4	0	0	1	0	0	0	1	2	

s_2 burada çıkan değişken, x_D ise giren değişken olarak bulunmuştur. Böylece temel değişkenlerde bir değişiklik olur. Şimdi tablonun tekrar hesaplanması gerekmektedir. Bu hesaplama işlemini matris operasyonlarını kullanarak yapılır. Yapılması gereken temel değişkenlerin giriş kolonundaki bütün değerleri, çıkış değerinin bulunduğu satır hariç, 0 yapmaktır. Çıkış değişkeni ile giriş değişkeninin kesiştiği kutudaki değerin 1 olması gerekmektedir. Bu kesişimdeki kutucuğa pivot kutucuk adı verilmektedir.

İşlemlere başlarken ilk yapılması gereken pivot elemanının değerini 1 yapmaktır. Bunun için pivot elemanın bulunduğu satırdaki her değeri pivot elemanının değerine bölmek gerekmektedir. Burada kullanılan kural, matematikte Gaus-Jordan çıkarma kuralı olarak bilinen, bir matris'in bir satırını aynı değere bölmek ya da çarpmak ile matris'in değerinin değişmeyeceği kuralıdır. Ayrıca matris'in bir satırını diğer bir satıra eklediğimiz zaman da matris'in değeri değişmemektedir. Bu kuralları uygulayarak yeni tablo kolayca oluşturulur.
Giriş satırı düzenlendiğinde:

Tablo 2.4: Simpleks Tablosu Adım 2

Temel	Z	X_D	$X_İ$	s_1	s_2	s_3	s_4	Sonuç	
Z	1	-3	-2	0	0	0	0	0	
s_1	0	1	2	1	0	0	0	6	
s_2	0	1	½	0	½	0	0	4	
s_3	0	-1	1	0	0	1	0	1	
s_4	0	0	1	0	0	0	1	2	

Daha sonra da giriş sütunundaki pivot eleman dışında kalan bütün değerleri 0 değerine dönüştürmek için gerekli işlemler yapılmalıdır. Bu işlemleri yaparken işlem bütün bir satıra uygulanmak durumundadır. Örneğin, z satırındaki değer –3, bunu 0 yapmak için pivot elemanın değerini –3 ile çarpıp z satırına eklemek gerekmektedir. Bu işlemin diğer sütunlar için de yapılması gerekmektedir. İlk işlemlerden sonra durum şu şekildedir.

Eski z satırı:	(1	-3	-2	0	0	0	0	0)
+ (3)x pivot:	(0	3	3/2	0	3/2	0	0	12)
Yeni z satırı:	(1	0	-1/2	0	3/2	0	0	12)

Eski s_1 satırı:	(0	1	2	1	0	0	0	6)
+ (-1)x pivot:	(0	-1	-1/2	0	-1/2	0	0	-4)
Yeni s_1 satırı:	(0	0	3/2	1	-1/2	0	0	2)

İşlemler diğer satırlar için de yapılırsa aşağıdaki tablo oluşur.

Tablo 2.5: Simpleks Tablosu Adım

Temel	Z	X_D	$X_İ$	s_1	s_2	s_3	s_4	Sonuç	
Z	1	0	-1/2	0	3/2	0	0	12	
S_1	0	0	3/2	1	-1/2	0	0	2	4/3
x_D	0	1	1/2	0	1/2	0	0	4	8
s_3	0	0	3/2	0	1/2	1	0	5	10/3
s_4	0	0	1	0	0	0	1	2	2

Bu tabloya bakıldığında amaç fonksiyonunda hala negatif değerler olduğu görülmektedir. Şimdiki giriş değişkeni en küçük negatif değer olan (aslında tek negatif değer olan) $x_İ$ değişkenidir. Çıkış değişkeni için gene sonuç değerleri sütun değerlerine bölünür. Buradan da en ufak değer seçilerek, o değişken temel değişkenlerden çıkarılır.

Daha önceki işlemleri bu tablo üzerine uygulanılırsa aşağıdaki tabloya ulaşılır.

Tablo 2.6: Sonuç Simpleks Tablosu

Temel	Z	X_D	$X_İ$	s_1	s_2	s_3	s_4	Sonuç	
Z	1	0	0	1/3	4/2	0	0	38/3	12,6
$x_İ$	0	0	1	2/3	-1/3	0	0	4/3	
x_D	0	1	0	-1/3	2/3	0	0	10/3	
s_3	0	0	0	-1	1	1	0	3	
s_4	0	0	0	-2/3	1/3	0	1	2/3	

Bu tablo optimum düzeye ulaşmıştır. Z satırındaki hiçbir değer negatif değildir, yani temel değişkenlere katılması gerekecek bir değişken kalmamıştır. Firmanın üretmesi gereken İç boya 4/3 ton, Dış Boya ise 10/3 tondur. Bu durumda kazanç 38/3 bin dolar olacaktır.

2.3.4.3. Büyük M metodu

Eğer kısıtlamalar "≤" yerine "≥" olsaydı, standart formata geçirildiğinde, eklenecek değişkenlerin katsayıları "-1" olacak ve eşitliklerin sağ tarafının pozitif olması şartı sebebiyle temel değişkenlerin seçiminde eklenen değişkenler kullanılamayacaktı. Bu durumda sisteme yeni yapay değişkenler eklenerek problem çözülebilir. Yapay değişkenlerin çözüme etki etmemesi için, çözümde bu değişkenlerin 0 olmasının sağlanması gerekmektedir. Bunu yapmak için amaç fonksiyonuna "M" olarak göstereceğimiz büyük bir katsayı ile eklenir ve optimum çözümde sistemin bu değişkenleri 0 yapması sağlanır.

Örnek:

Min $z = 4x_1 + x_2$

Kısıtlamalar:

$3x_1 + x_2 = 3$ (2.17)

$4\,x_1 + 3\,x_2 \geq 6$

$x_1 + x_2 \leq 4$

$x_1, x_2 \geq 0$

Standart formda yazılınca:

Min $z = 4x_1 + x_2$

Kısıtlamalar:

$3x_1 + x_2 = 3$ (2.18)

$4\ x_1 + 3\ x_2 - x_3 = 6$

$x_1 + x_2 + x_4 = 4$

$x_1, x_2, x_3, x_4 \geq 0$

Birinci ve ikinci satırların istenilen değişkenleri bulunmamaktadır. Burada iki yapay değişken eklenerek sorun giderilir.

Min $z = 4x_1 + x_2 + MR_1 + MR_2$

Kısıtlamalar:

$3x_1 + x_2 + R_1 = 3$ (2.19)

$4\ x_1 + 3\ x_2 - x_3 + R_2 = 6$

$x_1 + x_2 + x_4 = 4$

$x_1, x_2, x_3, x_4, R_1, R_2 \geq 0$

Tabloyu oluştururken amaç satırında temel değişkenlerin katsayısının 0 olması gerekmektedir. Bunu başarabilmek için R_1 ve R_2 eşitliklerden çekilerek amaç fonksiyonunda yerine konulur ve yeni amaç fonksiyonu bulunur.

$R_1 = 3 - 3x_1 - x_2$ (2.20)

$R_2 = 6 - 4x_1 - 3x_2 + x_3$

Amaç

$Z = 4x_1 + x_2 + M(3 - 3x_1 - x_2) + M(6 - 4x_1 - 3x_2 + x_3)$ (2.21)

$Z = (4-7M)\ x_1 + (1- 4M)\ x_2 + Mx_3 + 9M$

Tabloda görünecek şekli:

$Z - (4-7M)x_1 - (1- 4M)x_2 - Mx_3 = 9M$ (2.22)

Dikkat edilmesi gereken bir diğer olay da problemin en küçükleme problemi olduğudur. Bu durumda giriş değişkeni seçilirken en küçük olan değil, en büyük katsayısı olan seçilecektir. Problem hiç pozitif katsayı kalmayınca duracaktır.

Tablo 2.7: Büyük M Metodu Adımları Tablosu

Adım	Temel	X_1	X_2	X_3	R_1	R_2	X_4	Sonuç	Oran
0 başlangıç	Z	-4+7M	-1+4M	-M	0	0	0	9M	
X_1 girer	R_1	3	1	0	1	0	0	3	3/3
R_1 çıkar	R_2	4	3	-1	0	1	0	6	6/4
	X_4	1	2	0	0	0	1	4	4/1

Adım	Temel	X_1	X_2	X_3	R_1	R_2	X_4	Sonuç	Oran
1	Z	0	(1+5M)/3	-M	(4-7M)/3	0	0	4+2M	
X_2	X_1	1	1/3	0	1/3	0	0	1	3
girer	R_2	0	5/3	-1	-4/3	1	0	2	1,2
R_2	X_4	0	5/3	0	-1/3	0	1	3	1,85
çıkar									

Adım	Temel	X_1	X_2	X_3	R_1	R_2	X_4	Sonuç	Oran
2	Z	0	0	1/5	(8/5)-M	(-1/5) - M	0	18/5	
X_3	X_1	1	0	1/5	3/5	-1/5	0	3/5	3
girer	X_2	0	1	-3/5	-4/5	3/5	0	6/5	Yok
X_4	X_4	0	0	1	1	-1	1	1	1
çıkar									

Adım	Temel	X_1	X_2	X_3	R_1	R_2	X_4	Sonuç	Oran
3	Z	0	0	0	(7/5)-M	-M	-1/5	17/5	
Optimum	X_1	1	0	0	2/5	0	-1/5	2/5	
	X_2	0	1	0	-1/5	0	3/5	9/5	
	X_3	0	0	1	1	-1	1	1	

Bu şekilde M metodunu kullanarak problem çözülür.

Örnek Problem;

Boyut Ltd şirketi masa ve iskemle üretmektedir. Her iki ürün de birincisi montaj, ikincisi ise cilalama aşamaları olmak üzere iki aşamadan geçmek zorundadır. Montaj bölümünün iş kapasitesi 60 saat, cilalama bölümünün iş kapasitesi ise 48 saattır. Bir masanın üretilebilmesi için montaj bölümünün 4 saat, cilalama bölümünün ise 2 saat çalışması gerekmektedir. Bir iskemlenin üretilebilmesi için ise montaj bölümümün 2 saat, cilalama bölümünün 4 saat çalışması gerekmektedir. Şirket masa başına 8 $

iskemle başına da 6 $ kazandığına göre en fazla karı elde edebilmesi için kaçar iskemle ve masa üretmelidir?

Başlangıç Çözümünün Oluşturulması;

İ = iskemle, M = masa olsun.

Cebirsel olarak problemi şöyle ifade edebiliriz:

Kısıtlamalar: Montaj : $4.M + 2.İ \leq 60$

Cilalama: $2.M + 4.İ \leq 48$

Tüm değişkenler ≥ 0

Maximum olması istenen fonksiyon:

$$Kar = 8.M + 6.İ$$

Eşitsizliklerin denklemlere dönüştürülmesi;

En iyi masa ve iskemle sayısı kombinasyonunu elde edebilmek için mutlaka her iki bölümünde tüm iş saati kapasitesini kullanmak gerekmeyebileceğini söyleyebiliriz. Dolayısıyla bunları kullanılmayan zaman değişkeni olarak niteleyip her iki eşitsizliğe eklemek suretiyle bu eşitsizliklerden birer eşitlik elde etmek mümkün olacaktır. Bu işlemden önce var olan değişkenleri ise yapısal değişkenler olarak adlandırabiliriz.

Sm: Montaj bölümünde kullanılmayan zaman

Sc : Cilalama bölümünde kullanılmayan zaman

Bu değişkenleri ekledikten sonra şu denklemleri elde ederiz:

$4.M + 2.İ + Sm = 60$ saat

$2.M + 4.İ + Sc = 48$ saat

Simpleks metodunun yürütülebilmesi için denklemlerden birinde kullanılan değişkenin diğer denklemlerde de kullanılması gerekmektedir. Eğer bir değişken

belli bir denklemi etkilemiyorsa katsayısı sıfır olarak alınmalıdır. Bu değişikliği de gerçekleştirdikten sonra denklemlerimiz şu şekli alacaktır:

Kısıtlamalar: Montaj: $4.M + 2.İ + Sm + 0.Sc = 60$ saat

Cilalama: $2.M + 4.İ + 0.Sm + Sc = 48$ saat

Tüm değişkenler ≥ 0

Maximum olması istenen fonksiyon:

$$Kar = 8.M + 6.İ + 0.Sm + 0.Sc$$

Simpleks tablosu;

Yukarıdaki denklemleri matris yapısında bir tablo ile de gösterebiliriz:

	M	İ	Sm	Sc
60	4	2	1	0
48	2	4	0	1

İlk çözümün simpleks tablosunda gösterilmesi

Simpleks metodunda bir ilk çözümün oluşturulması gerekmektedir. Boyut Ltd için en basit başlangıç çözümü hiçbir masa ve iskemle üretmeyip sıfır kazanç sağlamak olacaktır. Bu çözüm matematiksel olarak mümkündür, ancak finansal olarak pek te cazip bir çözüm gibi gözmemektedir.

$M = 0$

$İ = 0$

$Sm = 60 – 4.(0) – 2.(0) = 60$ saat kullanılmayan zaman

$Sc = 48 – 2.(0) –4.(0) = 48$ saat kullanılmayan zaman

Karı ise şöyle hesaplayabiliriz:

Kar = 8.M + 6.İ +0.Sm + 0.Sc

Kar = 8.(0) + 6.(0) + 0.(60) + 0.(48)

Kar = 0

Bu ilk geçerli çözüm simpleks tablosunda şöyle gösterilebilir:

Ürün Kombinasyonu	Miktar	M	İ	Sm	Sc
Sm	60	4	2	1	0
Sc	48	2	4	0	1

Bu çözüm grafik metodu ile karşılaştırılacak olursa aynı problemin çözüm grafiğindeki A noktasına karşılık gelmektedir. Bu da grafikteki konveks bölgenin köşelerinden birini oluşturmaktadır.

İkame oranlarının simpleks tablosunda gösterilmesi

Öncelikle aşağıdaki tablonun bileşenlerini inceleyerek her bir bileşenin ne anlama geldiğini görelim:

Cj			8$	6$	0$	0$
Ürün kombinasyonu		Miktar	M	İ	Sm	Sc
0$	Sm	60	4	2	1	0
0$	Sc	48	2	4	0	1

Cj sütunu Sm ve Sc değişkenleri için birim başına karı göstermektedir.

Son iki sütun eşitsizlikleri eşitliğe dönüştürmek için eklenen değişkenlerin katsayılarını göstermektedir.

Dört ve beşinci sütunlar imalatı yapılan gerçek ürünlerin katsayılarını göstermektedir. Örneğin M sütunundaki 4 sayısının anlamı, eğer bir masa üretecek olursak (yani 1 masayı çözüme dahil edecek olursak) montaj bölümündeki 4 saatten vazgeçmek zorunda kalmamızdır. Bu durumda dört ve beşinci sütunlardaki elemanların aslında ikame oranlarını gösterdiğini söyleyebiliriz.

Bu ikame oranlarını incelediğimizde iki tür işlem yaptığımızı görebiliyoruz:

1. Üretim programına, veya çözüme, M ve İ gerçek ürünlerini eklemek
2. Başka amaçlarla kullanmak üzere her iki bölümün kullanılmayan iş zamanından fedakarlık etmek.

Simpleks tablosuna iki satır daha eklemek

Şu aşamaya kadar simpleks tablosunu oluştururken herhangi bir işlem devreye girmedi. Yaptığımız sadece problemin denklemlerini farklı bir düzende ifade etmekti. Her bir çözüm için elde edilecek karı bulmak ve bu çözümün iyileştirilebilip iyileştirilemeyeceğine karar verebilmek için ilk oluşturduğumuz simpleks tablosuna iki satır daha eklememiz gerekir. Bunlar Zj ve Cj – Zj satırlarıdır.

Cj			8$	6$	0$	0$
Ürün kombinasyonu		Miktar	M	İ	Sm	Sc
0$	Sm	60	4	2	1	0
0$	Sc	48	2	4	0	1
	Zj	0$	0$	0$	0$	0$
	Cj – Zj		8$	6$	0$	0$

Zj satırında miktar sütununa karşılık gelen değer bu özel çözümden elde edilen toplam karı göstermektedir.(İlk çözümümüzde bu değer sıfır idi.)
Zj satırında diğer sütunlara karşılık gelen değerler ise M, İ, Sm, Sc değişkenlerinden bu kombinasyona bir birim eklenmesi halinde karın ne kadar azalacağını göstermektedir. Örneğin bir adet masa imal etmek istediğimiz takdirde M sütunundaki 4 ve 2 bize SM den 4 saat SC den de 2 saat azaltmak zorunda olduğumuzu söylemektedir. Ancak kullanılmayan zamanın değeri 0$ ettiğinden karda bir azalma olmamaktadır.
Cj birim başına kar olarak tanımlanmıştır. Örneğin masa için Cj 8$/masa dır.

Bu durumda Cj – Zj , üretim programına veya diğer bir deyişle çözüme bir birimlik bir değişken eklemenin getireceği net karı göstermektedir. Örneğin 1 birim M eklenmesi çözüme 8$ kar ekliyor ve 0$ zarara sebep oluyor. Yani M için Cj – Zj = 8$ olmaktadır.

Yorum; Cj – Zj satırının pozitif sayılardan oluşması eklenen her bir birim için fazladan o kadar karın artacağı anlamına gelmektedir. Aynı şekilde bu satırda negatif sayıların olması da eklenen her birim için karın o kadar azalacağını göstermektedir. O halde eğer bu satır da hiçbir pozitif değer kalmamışsa bu, daha fazla kar edilemeyeceği, yani optimum sonuca ulaşıldığı anlamına gelecektir.

İkinci Çözümün Oluşturulması;

Bir kez ilk simpleks tablosu oluşturulduktan sonra artık diğer aşama karı nasıl arttırabileceğimizi bulmaktan ibarettir. Bu kısımda verilecek hesaplama yöntemi gerek ikinci çözüm gerekse takip eden çözüm tablolarının oluşturulmasında kullanılacaktır. Bir tablodan diğerine geçme işlemi pivot alma olarak adlandırılır. Ve her bir iterasyon pivot adını alır. İlk pivot şu şekildedir:

1.Aşama;

Hangi değişkenin bir biriminin en fazla karı getireceğini saptarız. Bu bilgiyi bize Cj–Zj satırı verecektir. Daha önce de belirttiğimiz gibi bu satırda pozitif sayıların varlığı karın daha fazla arttırılabileceğini ifade eder. Bu satırdaki pozitif sayı ne kadar büyük olursa karı da o kadar çok arttırabileceğimiz anlamı çıkar. Tablomuza baktığımızda kara en fazla katkıda bulunan ürünün 8$ ile M olduğunu görüyoruz. Bu durumda M sütunu optimum sütun olarak alınacaktır. Kolaylıkla görülebileceği gibi optimum sütun Cj – Zj satırında en büyük pozitif değeri alan sütundur, veya bir başka deyişle kara en fazla katkıda bulunan ürüne ait sütundur. Bu durumda ürün kombinasyonunda var olan değişkenlerden biri yerine M değişkeninin koyulması gerekmektedir.

2.Aşama;

Bu aşamada M değişkeninin hangi değişkenin yerine geçeceğini saptamamız gerekmektedir. Bunu da şöyle gerçekleştiririz: Miktar sütunundaki 60 ve 48 sayılarını sırasıyla M sütunundaki 4 ve 2 sayılarına bölüp, elde edilen en küçük pozitif oranın bulunduğu satırı yer değiştirilecek satır olarak belirleriz.

Sm satırı (60 saat-kullanılabilir süre) / (4 saat-bir ürün için gereken süre) = 15 tane M
Sc satırı (48 saat-kullanılabilir süre) / (2 saat-bir ürün için gereken süre) = 24 tane M

Sm satırı daha küçük pozitif orana sahip olduğundan yer değiştirecek satır olacaktır. Bu arada Sm ve Sc satırlarıyla optimum sütunun kesişiminde yer alan elemanlara da arakesit elemanları adı verilir. Sm satırına ait arakesit elemanı 4, Sc satırına ait arakesit elemanı ise 2 dir.

3.Aşama;

Artık optimum sütunu ve yer değiştirecek satırı seçtiğimize göre ikinci simpleks çözümü veya diğer adıyla geliştirilmiş çözümü bulabiliriz. Yeni tablodaki ilk işlemimiz yer değiştirecek satırı kaldırıp yerine M satırını koymak olacaktır. Bu yeni satırı ise, yer değiştirecek satırın tüm elemanlarını, yine yer değiştirecek satırın arakesit elemanına bölerek elde ederiz. Yani 60/4 = 15, 4/4 = 1, 2/4 =1/2, 1/4 = 1/4, 0/4 = 0. Böylece yeni M satırının katsayıları (15, 1, ½, ¼, 0) olarak elde edilecektir.

Cj			8$	6$	0$	0$
Ürün kombinasyonu		Miktar	M	İ	Sm	Sc
8$	M	15	1	½	¼	0
0$	Sc					
	Zj					
	Cj – Zj					

Bu tabloda karşımıza ilk defa bir pozitif Cj değeri çıkmış oldu (8$). Ayrıca üretilen 15 adet masanın grafik çözümde karşılığı olan noktanın C noktası olduğunu gözlemlemekte de fayda vardır. Bu da kapalı konveks bölgenin bir başka köşe noktasını oluşturmaktadır.

4.Aşama;

İkinci tabloyu tamamlayabilmek için kalan satırların da yeni değerlerini hesaplamamız gerekmektedir. Bu hesaplama da şu formül ile kolayca yapılabilir:

(eski satır elemanı) – [(eski satırın arakesit elemanı) x (yeni satırdaki buna karşılık gelen eleman)] = (yeni satır)

Yukarıdaki formülü her bir elemana uygulayacak olursak tabloyu şu şekilde elde ederiz:

Cj			8$	6$	0$	0$
Ürün kombinasyonu		Miktar	M	İ	Sm	Sc
8$	M	15	1	½	¼	0
0$	Sc	18	0	3	- ½	1
	Zj	120$	8$	4$	2$	0$
	Cj – Zj		0$	2$	-2$	0$

Üçüncü Çözümün Oluşturulması;

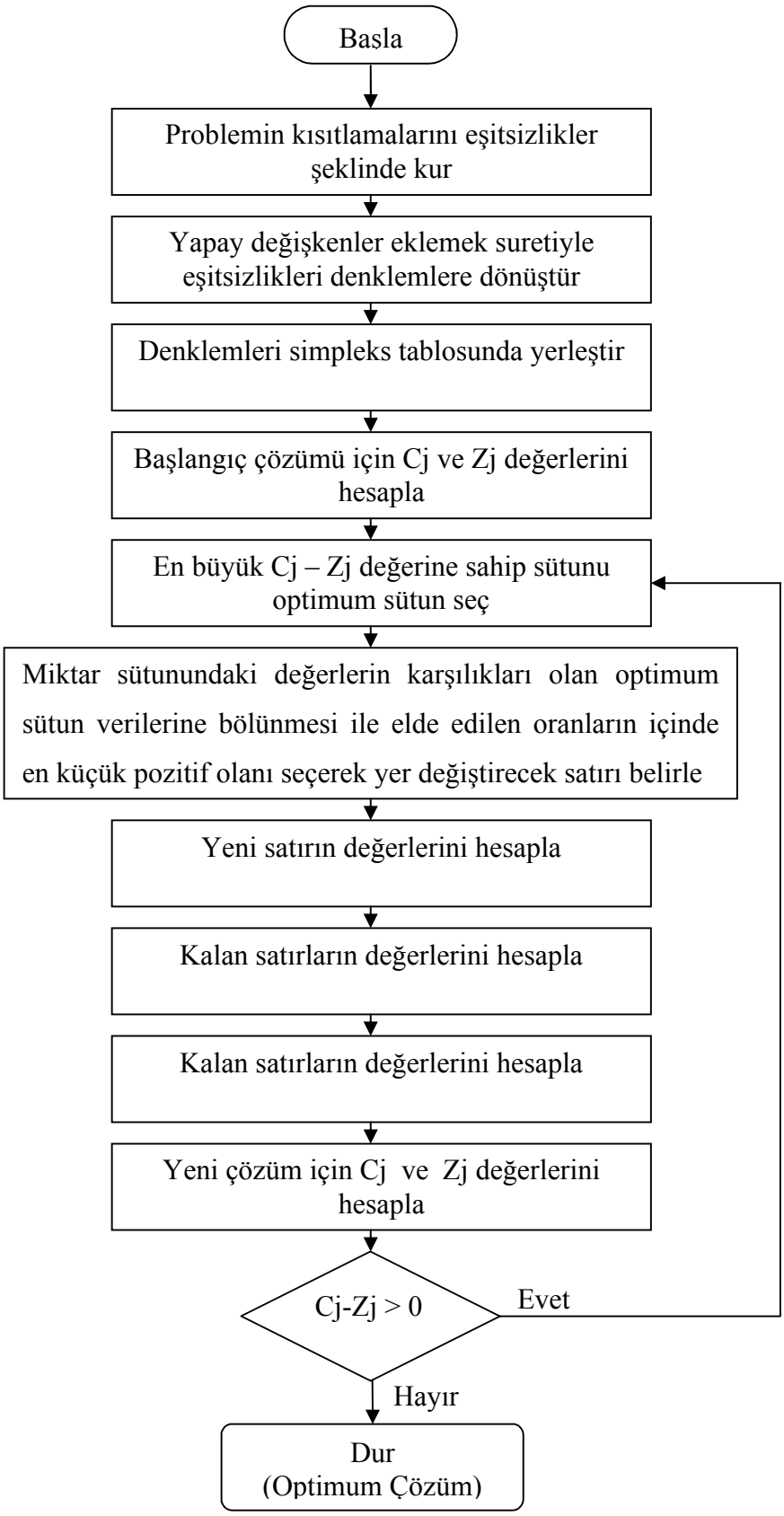
Cj – Zj satırının İ sütunundaki pozitif sayının varlığı çözümün bir aşama daha öteye götürülebileceğini göstermektedir. Bu durumda aynı aşamaları tekrar gerçekleştirerek üçüncü tabloyu elde ederiz:

Cj			8$	6$	0$	0$
Ürün kombinasyonu		Miktar	M	İ	Sm	Sc
8$	M	12	1	0	1/3	-1/6
6$	İ	6	0	1	- 1/6	1/3
	Zj	132$	8$	6$	5/3$	2/3$
	Cj – Zj		0$	0$	-5/3$	-2/3$

Kolayca görülebileceği gibi artık tablonun Cj – Zj satırında hiçbir pozitif değer kalmamıştır. Dolayısıyla karın bundan öteye arttırılması imkansızdır. Yani optimum sonuca ulaşılmıştır. Ayrıca bu çözümün de grafik çözümdeki D noktasına karşılık geldiğini gözlemleyebiliriz.

Simpleks Maximizasyon Metodundaki 11 Aşamanın Özeti;

1.Problemin kısıtlamalarını eşitsizlikler şeklinde kur.
2.Yapay değişkenler eklemek suretiyle eşitsizlikleri denklemlere dönüştür.
3.Denklemleri simpleks tablosunda yerleştir.
4.Başlangıç çözümü için Cj ve Zj değerlerini hesapla.
5.En büyük Cj – Zj değerine sahip sütunu optimum sütun seç.
6.Miktar sütunundaki değerlerin karşılıkları olan optimum sütun verilerine bölünmesi ile elde edilen oranların içinde en küçük pozitif olanı seçerek yer değiştirecek satırı belirle
7.Yeni satırın değerlerini hesapla
8.Kalan satırların değerlerini hesapla.
9.Yeni çözüm için Cj ve Zj değerlerini hesapla.
10.Pozitif bir Cj – Zj değeri bulursan 5'inci aşamaya dön.
11.Hiçbir pozitif Cj – Zj değeri kalmadıysa optimum çözüme ulaşıldı demektir.



Şekil 2.1: Simpleks maksimizasyon metodunun akış şeması

2.4. Tamsayılı Programlama

Lineer Programlama sorunlarının modeli kurulurken bazı karar değişkenlerinin kesinlikle tamsayı değerler alması gerektiğini görürüz fakat Lineer Programlamaya uygun olması için büyük değerler alabilecek karar değişkeninin kesirli kısmı yok sayılabilir mantığı ile kesirli değerler almalarına izin veririz. Bazı durumlarda bu varsayım kabul edilemez olabilir. Bulunacak sayısal sonucun tamsayı değerler alması gerekebilir. Bu tip durumlardaki sorunlara Tamsayılı Programlama (TP; Integer Programming – IP) denilir ve söz konusu programların çözümü konusu Tamsayılı Programlama (TP) olarak isimlendirilir.

Sonlu sayıda seçenek kümesindeki seçenekler ile ilgili birçok kararın yap/yapma, evet/hayır gibi kesikli olması yüzünden Tamsayılı Programlama sıklıkla kullanılır. Tüm değişkenlerin tamsayı olduğu bir programlama sorunu tamamen tamsayılı programlama sorunu olarak isimlendirilir. Eğer değişkenlerin bazıları tamsayı, bazıları kesirli ise söz konusu sorun Karma Tamsayılı Programlama sorunu adını alır. Bazı durumlarda tamsayı değişkenler sadece 0 veya 1 değerlerini alabilir. Bu tip sorunlar da tamamen (karma) 0-1 programlama sorunları veya tamamen (karma) ikili tamsayı programlama sorunları olarak isimlendirilir.

2.4.1. Tamsayılı programlamanın formülasyonu

Doğrusal Programlama modeli ile Tamsaysayılı Doğrusal Programlama arasındaki fark, Doğrusal Programlama modelinde karar değişkenlerinin sıfır ve sıfırdan büyük olma koşulu aranırken, Tam Sayılı Doğrusal Programlama da değişken değerlerinin sıfıra eşit ve sıfırdan büyük tam sayı olmaları şartının istenmesidir.

Genel olarak Tam Sayılı Doğrusal Programlama problemi sembolik olarak şöyle gösterilebilir;

$$\text{Maks. } Z = \sum_{j=1}^{n} c_j x_j$$

$$\text{Kısıtlayıcılar } \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i=1, 2, ...m \tag{2.23}$$

$$x_j = 0, 1, 2, ...\text{tam sayı } (J = 1, 2, ...n)$$

Tam Sayılı Doğrusal Programlama modeli, değişkenlerin alacağı tam sayı değerlerine göre iki kategoride incelenir.

Bunlar;

1. Karışık(karma) tamsayılı programlama; n tane karar değişkeninden k tanesi için tam sayı olma koşulu, n-k tanesi için pozitif olma koşulu vardır.

2. Saf(tüm-tamamen) tamsayılı programlama : Karar değişkenlerinin tamamının tam sayı değer alması durumudur.

Saf tamsayılı modeller de tam sayılı değişkenlerin alabilecekleri de_erler itibariyle ikiye ayrılırlar. Tam sayılı değişkenler, kısıtların izin verdiği ölçüde her pozitif tam sayı değeri alabiliyorsa pozitif modeller, sadece 0 ve 1 de_erlerini alabiliyorlarsa sıfır-bir modeller olarak adlandırılırlar.

Tam Sayılı Doğrusal Programlama modellerinin kullanım alanlarından bazıları;
Malzeme kullanımı, sabit maliyetlerin hesabı, yığın üretim sorunları, yap-yapma kararları, gezgin satıcı problemleri, işçileri makinelere atama problemleri, kritik yol problemleri vb. şeklinde sıralanabilir.

2.4.2. Tamsayılı programlama ile ilgili çeşitli örnekler

Örnek1;
Farz edelim ki $14,000 yatırım yapmak istiyoruz. Dört yatırım fırsatı belirledik. Yatırım 1 $5,000'lık bir yatırım gerektirmekte ve getirisinin şimdiki değeri $8,000

olmaktadır. Yatırım 2 $7,000'lık bir yatırım gerektirmekte ve getirisinin şimdiki değeri $11,000 olmaktadır. Yatırım 3 $4,000'lık bir yatırım gerektirmekte ve getirisinin şimdiki değeri $6,000 olmaktadır. Yatırım 4 ise $3,000'lık bir yatırım gerektirmekte ve getirisinin şimdiki değeri $4,000 olmaktadır.

Hangi yatırımlara para yatırarak toplam getirimizi enbüyükleyebiliriz?

Yanıt;
LP'de olduğu gibi ilk adım değişkenleri belirlemektir. Tamsayı sınırlamasının getirdiği çeşitli karışıklıklar ve numaralar yüzünden TP'de bu işlem daha zordur.
Örnekte her yatırım için bir 0-1 xj değişkeni kullanacağız

xj 1 ise *j*. yatırım yapılır
xj 0 ise *j*. yatırım yapılmaz

Bu durumda 0-1 programlama sorunu söz konusudur:

maks $\quad 8\,x1 + 11\,x2 + 6\,x3 + 4\,x4$
öyle ki $\quad 5\,x1 + 7\,x2 + 4\,x3 + 3\,x4 \leq 14$
$xj = 0$ veya 1 $j = 1, \ldots 4$

Tamsayı kısıtlarına önem verilmezse en iyi LP çözümü aşağıdaki gibidir:
$x1 = 1$, $x2 = 1$, $x3 = 0.5$ ve $x4 = 0$; toplam getiri değeri $22,000.

Ne yazık ki bu çözüm tamsayılı çözüm değildir. x3 0'a yuvarlanırsa $19,000.toplam getiri değeri olan olurlu bir çözüm elde edilir.

TP çözüm tekniği kullanılarak en iyi tamsayılı çözüm bulunursa $x1 = 0$, $x2 = x3 = x4 = 1$ ve toplam getiri değeri $21,000 olur.

Doğrudan yapılacak bir değerlendirmede (amaç fonksiyonu katsayılarının kısıt katsayılarına oranlanması) Yatırım 1'in en iyi seçenek olduğu önerilebilir.
Bu örnek yuvarlamanın her zaman en iyi değeri vermediğini gösterir.

Örnek2;

Önümüzdeki üç yıllık planlama dönemi için dört projenin değerlendirilmesi yapılacaktır. Her projeye ait beklenen getiriler ile yıllık harcamalar aşağıdaki tabloda gösterilmiştir. Üç yıl boyunca uygulamaya konulacak projeleri belirleyiniz.

		Harcamalar		
Proje	Getiri	Yıl1	Yıl2	Yıl3
1	0.2	0.5	0.3	0.2
2	0.3	1	0.5	0.2
3	0.5	1.5	1.5	0.3
4	0.1	0.1	0.4	0.1
Kullanılabilir sermaye		3.1	2.5	0.4

Yanıt;

xj = 1, j. proje uygulanırsa
xj = 0, aksi takdirde (j. proje uygulanmaz).

Kurulacak 0-1 programlama modeli:

Maks $0.2 x1 + 0.3 x2 + 0.5 x3 + 0.1 x4$

Öyle ki $0.5 x1 + 1 x2 + 1.5 x3 + 0.1 x4 \leq 3.1$

$0.3 x1 + 0.8 x2 + 1.5 x3 + 0.4 x4 \leq 2.5$

$0.2 x1 + 0.2 x2 + 0.3 x3 + 0.1 x4 \leq 0.4$

xj = 0 veya 1 j = 1, … 4

Örnek3;

Bazı yeni kısıtlarla sorunu genişletelim.
Örneğin, aşağıdaki önermeleri ve getirdiği kısıtları düşünelim:

Sadece iki yatırım yapabiliriz

$x1 + x2 + x3 + x4 \leq 2$

Üç veya dört adet yatırım yapılması durumunda $x1 + x2 + x3 + x4 > 3$ olacaktır.

Eğer Yatırım 2 yapılırsa,Yatırım 4 de yapılmalıdır

$x2 < x4$ veya $x2 – x4 \leq 0$

Eğer x2 1 ise, x4 de istediğimiz gibi 1 olur; eğer x2 0 ise, x4 için herhangi bir kısıtlama

olmaz (x4 0 veya 1 olur)

Eğer Yatırım 1 yapılırsa, Yatırım 3 yapılamaz

$x1 + x3 \leq 1$

Eğer x1 1 ise, x3 istediğimiz gibi 0 olur; eğer x1 0 ise, x3 için herhangi bir kısıtlama olmaz (x3 0 veya 1 olur)

Ya Yatırım 1 ya Yatırım 2 yapılabilir

$x1 + x2 = 1$

Eğer x1 1 ise, x2 0 olur (sadece Yatırım 1 yapılır); eğer x1 0 ise, x2 1 olur (sadece Yatırım 2 yapılır).

Örnek4;

Sadece bir kısıdı olan herhangi bir TP sırt çantası sorunu olarak isimlendirilir. Bir diğer özellik de kısıdın ve amaç fonksiyonunun katsayıları negatif olmayan sayılardır.

Örneğin aşağıdaki model sırt çantası sorunudur:

Maks $8 x1 + 11 x2 + 6 x3 + 4 x4$

Öyle ki $5 x1 + 7 x2 + 4 x3 + 3 x4 \leq 14$

$xj = 0$ veya 1 $j = 1, \ldots 4$

Geleneksel öykü, bir sırt çantasında (örnekte kapasitesi 14'dür) bazı eşyaları (dört çeşit eşya) taşımak ile ilgilidir. Her eşyanın bir büyüklüğü ve taşıma sonucu elde edilecek bir değeri vardır (eşya 2 için büyüklük 7, değer 11'dir).

Amaç, sırt çantasına sığabilecek eşyaların toplam değerini enbüyüklemektir.
Sırt çantası sorunları genellikle kolay çözülürler.

2.4.3. Tamsayılı programlamanın çözümü

Tamsayılı programlamanın çözümü üzerine dört farklı yöntemden bahsedilecektir.

2.4.3.1. Grafiksel çözüm yöntemi

Lineer Programlama problemlerinden de bilindiği gibi grafik çözüm yöntemi en fazla üç değişken ile ilgilenildiği durumda kullanılabilen bir yöntemdir. Ancak yaygın olarak iki değişken olması durumunda kullanılmaktadır, (Tekin, 2004). Lineer Programlama problemlerinde olduğu gibi kısıtlayıcı denklemler yardımıyla belirlenen uygun çözüm alanı içerisindeki tamsayılardan oluşan değerler, Tam Sayılı Doğrusal Programlama problemi için çözüm değerlerini oluştururlar. Bunlar içerisinden en uygun nokta, deneme yanılma yöntemi ile bulunur.

2.4.3.2. Gomory'nin kesme düzlemi yöntemi

Lineer Programlama problemlerinin tamsayılı çözümlerini sağlayacak hesaplama yöntemi 1959 yılında R.E. Gomory tarafından geliştirilmiştir. Gomory'nin geliştirdiği hesaplama yöntemine tamsayılı algoritma veya kesme düzlemi yöntemi adı verilmiştir. Bu yöntem tüm(saf) tamsayılı programlamayı ve karışık tamsayılı programlamayı içermektedir, (Öztürk, 2001).

Bu yöntemde takip edilecek aşamalar şunlardır; (Öztürk, 1986)

i. Bir Tam Sayılı Lineer Programlama probleminde ilk aşama, eğer gerekli ise, orijinal sınırlamaları tamsayılaştırmadır. Bu, katsayılar tam olsun diye, tüm sınırların değiştirilmesi anlamına gelir.

ii. Lineer Programlama probleminin optimal çözüm tablosu bulunur. Eğer optimal çözüm değerleri tamsayı ise, Tam Sayılı Lineer Programlama problemi için çözüm elde edilmiştir. Yoksa sonraki aşamaya geçilir.

iii. Bu aşamada kesme bulunur. Bu amaçla optimal çözüm tablosundan tamsayı olmayan değişkenlerin biri seçilir ve yeni bir kısıtlama elde edilir.

İki değişkenli problemlerde Gomory kısıtlamasının özellikleri şunlardır; (Aslan, 1978)

- Elde edilen bu kısıtlamalar bir önceki uygun alandan genellikle konveks bir alan keserler.
- Kesme düzeyi uygun olmasa bile en az bir kafes noktasından geçer.
- Her kesim, bütün uygun kafes noktalar_n_ kapsayacak daha küçük bir alana yaklaşır.

iv. Eşitlik sınırlaması (Gomory'nin kesme düzlemi) optimal çözüm tablosuna yeni bir sıra olarak eklenir. Eklenen sınırlamadaki katsayılar tüm tamsayıyı verecek şekildedir. Daha sonra yönteme Lineer Programlama çözüm yöntemleri uygulanarak optimal çözüm tablosu bulunur.

2.4.3.3. Dal sınır yöntemi

Tamsayılı Programlama problemlerinin değişkenlerinin bir kısmı veya hepsi bazen üst veya alt sınırla veya hem alt hem de üst sınırla ayrı ayrı şartlanmaktadır. Bu tipteki optimizasyon problemlerinin çözümü için çok genel bir yaklaşım dal-sınır tekniğidir. Bu yöntem A.H. Land ve A.G. Doig tarafından geliştirilmiştir, (Halaç, 1995). Bu yöntem ile minimizasyon tipli bir problem çözülürken temel mantık şöyledir: (Anderson ve diğ., 1991)

i. Tamsayılı programlama uygun çözüm alanı çok sayıda alt setlere ayrılır. Her bir alt set için sınır değeri hesaplanır. Alt ve üst sınırlardan, her bir alt set içinde çözüm değeri seçilir.
ii. Bir alt sınır (ilk alt setler arasından en küçüğü) ile alt set diğer bölümler için seçilir. Daha önce olduğu gibi alt ve üst sınırların her ikisi de hesaplanır. Bölünmeye optimal çözüm bulunana kadar devam edilir. Optimal çözüm herhangi bir alt set için alt sınırdan daha büyük olamaz.
Problem maksimizasyon tipli ise, çözüm yöntemi alt ve üst sınır seçimi dışında minimizasyon tipli problem gibidir.

Örnek;

Çok dönemli sermaye bütçeleme sorunu için TP modeli:

maks $0.2\ x1 + 0.3\ x2 + 0.5\ x3 + 0.1\ x4$

Öyle ki $0.5\ x1 + 1\ x2 + 1.5\ x3 + 0.1\ x4 \leq 3.1$

$0.3\ x1 + 0.8\ x2 + 1.5\ x3 + 0.4\ x4 \leq 2.5$

$0.2\ x1 + 0.2\ x2 + 0.3\ x3 + 0.1\ x4 \leq 0.4$

$xj = 0$ veya $1\ j = 1, \ldots 4$

Bu sorunun çözümünü güçleştiren değişkenlerin tamsayı değerler (sıfır veya bir) almaya zorlanmasıydı.

Eğer değişkenler kesirli değerler alabilseydi (sıfır ve bir arasındaki tüm değerler) sorunu LP olarak çözebilirdik. Sorunun bu şekilde bir LP gevşetmesini [xj = 0 veya 1 (j=1,...,4) yerine 0 <= xj <= 1 (j=1,...,4) kısıtlarını kullanalım] çözeceğimizi varsayalım.

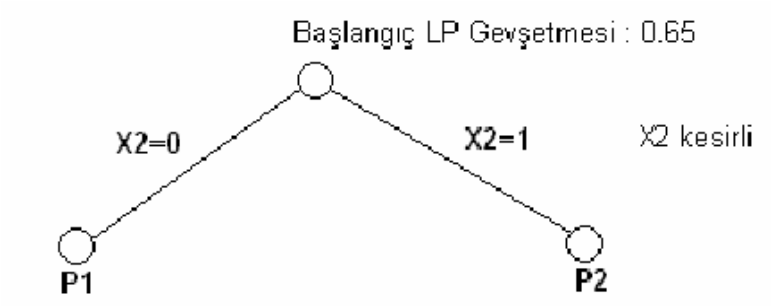
Herhangi bir LP çözüm paketi ile x2=0.5, x3=1, x1=x4=0 ve en iyi amaç fonksiyon değeri 0.65 olarak hesaplanır.
Bu durumda esas tamsayılı sorunumuzun en iyi amaç fonksiyon değeri hakkında fikir sahibi oluruz (<= 0.65). 0.65 en iyi tamsayılı çözüm için bir *üst sınır*dır (upper bound).

Söz konusu değerin üst sınır olmasının nedeni tamsayı zorunluluğunu kaldırınca elde edilen LP gevşetmesinin çözümünün değerinin en az tamsayılı çözümünki kadar (belki de daha iyi) olacağıdır.

Elde edilen LP gevşetmesi çözümünde x2 değişkeninin tamsayılı yerine kesirli bir değer aldığını görürüz. Tamsayılı değer alması için iki yeni sorunun çözümü ile ilgilenebiliriz:

P1: orijinal LP gevşetmesine ek olarak x2=0
P2: orijinal LP gevºetmesine ek olarak x2=1

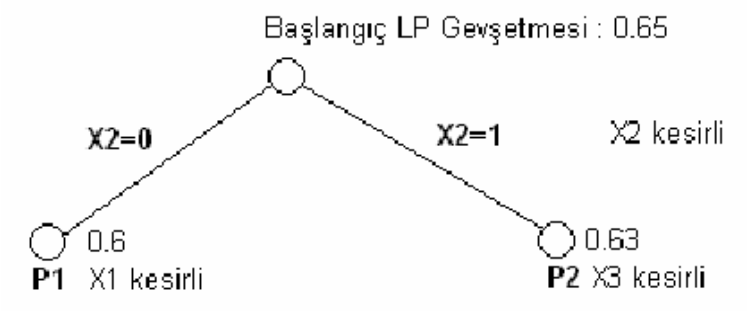
Bu şekilde sadece bir değişkenin kesirli değer yerine tamsayılı değer almaya zorlanarak yeni çözüm aranması işlemine *dal işlemi* (branching) denilir. Dal işlemi, aşağıdaki gibi bir ağaç üzerinde (ağaç arama isminin nasıl ortaya çıktığını gösterecek şekilde) şematize edilebilir.



Şekil 2.2: Tamsayılı Programlama Dal Sınır Yöntemi Adım 1

P1 ve P2 LP gevşetmelerini çözersek, aşağıdaki çözümleri elde ederiz:
P1'in çözümü x1=0.5, x3=1, x2=x4=0, amaç fn. değeri 0.6
P2'in çözümü x2=1, x3=0.67, x1=x4=0, amaç fn. değeri 0.63

Aşağıdaki şekilde de görüldüğü gibi yeni çözümlerdeki değişken değerleri de tamsayılı değildir:

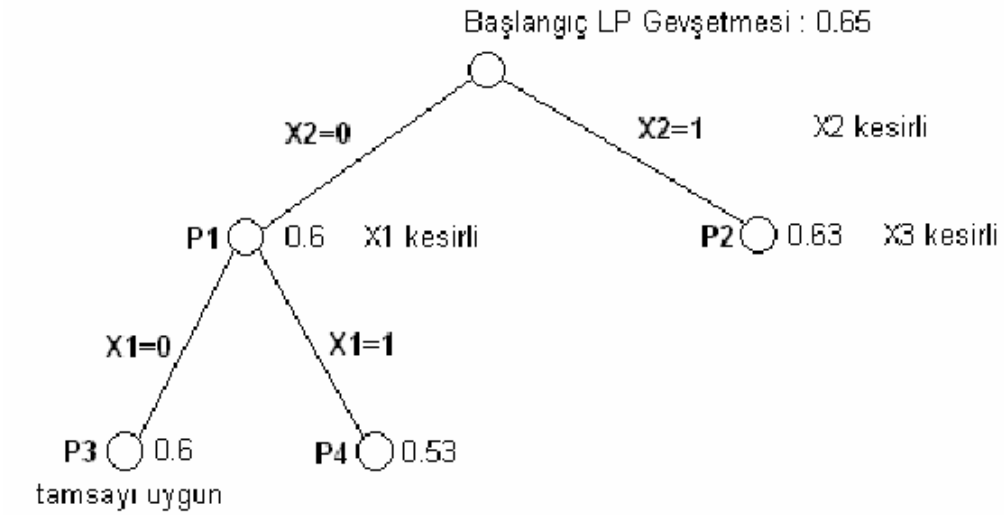


Şekil 2.3: Tamsayılı Programlama Dal Sınır Yöntemi Adım 2

En iyi tamsayılı çözümü bulmak için süreci tekrar ederiz, kesirli değer alan bir değişkeni tamsayılı olmaya zorlayarak iki yeni sorun üretiriz.

P1 sorunu üzerinde dal işlemi yapalım. Bu durumda elde edilecek LP gevşetmeleri ve çözümlerinin listesi:

P3 (P1'e ek olarak x1=0) çözümü x3=x4=1, x1=x2=0, amaç fn.: 0.6
P4 (P1'e ek olarak x =1) çözümü x1=1, x3=0.67, x2=x4=0, amaç fn. 0.53
P2 çözümü x2=1, x3=0.67, x1=x4=0, amaç fn: 0.63



Şekil 2.4: Tamsayılı Programlama Dal Sınır Yöntemi Adım 3

Görüldüğü gibi bu adımda P3 sorununun çözümü 0.6'dır ve tüm değişkenlerin değerleri tamsayılıdır. P3 üzerinde yeni bir dal işlemi yapmaya gerek kalmaz ve LP En iyi tamsayılı çözüm değeri hakkında yeni bir bilgi elde etmiş oluruz: amaç

fonksiyonunun en iyi çözüm değeri 0.6 ile 0.65 arasında (her iki değer dahil) olacaktır.

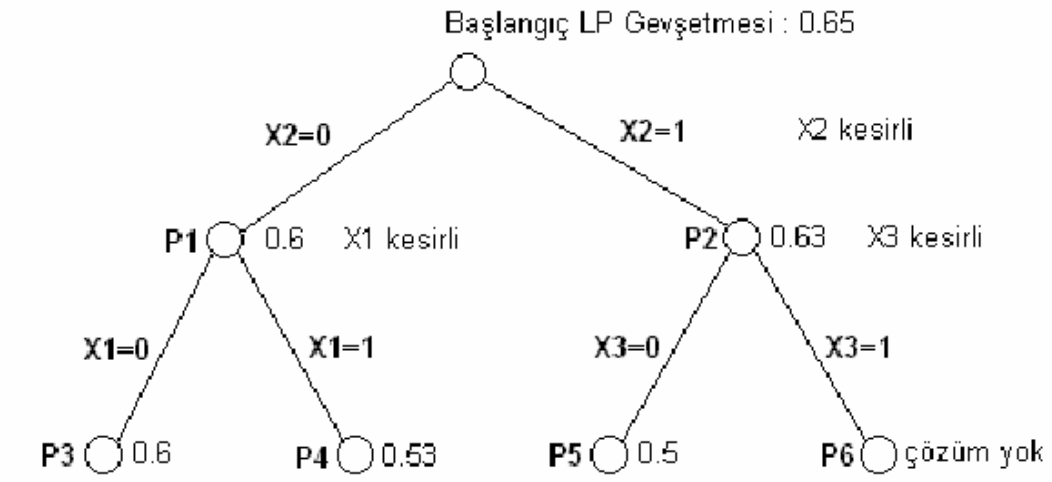
P4'ü düşünelim, şu anki değeri 0.53'dür ve x3 değişkeni kesirlidir. Yeni bir dal işlemi yapmamız durumunda amaç fonksiyon değeri daha iyi duruma gelemez. Zaten 0.6'lık bir tamsayılı çözümümüz olduğundan P4 de LP gevşetmeleri listesinden çıkarılır. Bu şekilde *alt sınır*dan (lower bound) daha iyi bir olurlu çözüm bulunamayacağı için eleme yapılmasına *sınır işlemi* (branching) denilir.

Geriye sadece P2 kalır;
P2 çözümü x2=1, x3=0.67, x1=x4=0, amaç fn: 0.63

P2 için dal işlemi yapılırsa
P5 (P2'ye ek olarak x3=0) çözümü x1=x2=1, x3=x4=0, amaç fn: 0.5
P6 (P2'ye ek olarak x3=1) çözümü olurlu değil

Bu durumda ne P5 ne de P6'dan dal işlemi yapmak gerekmez.
En iyi tamsayılı çözüm (esas sorunun çözümü) amaç fonksiyon değeri 0.6 ve x3=x4=1, x1=x2=0 şeklindedir.

En iyi çözümü elde etmek için kullanılan tüm süreç:



Şekil 2.5: Tamsayılı Programlama Dal Sınır Yöntemi Adım 4

2.4.3.4. Tamsayı olmayan çözümün yuvarlanması yöntemi

Bu yöntemde en iyi Lineer Programlama çözümünde değişkenlerin kesirli değerlerini bir yukarı veya bir aşağı tam sayıya yuvarlamak suretiyle tam sayılı çözüm elde edilir. Yuvarlama yöntemi ile iyi bir tam sayılı çözüm bulunabilir. Ancak bunun en iyi tam sayılı çözüm olduğunun garantisi yoktur. Bu yöntem Tam Sayılı Doğrusal Programlama problemi çözümünde elverişli bir yöntemdir. Elverişliliği, zaman ve masraftan tasarruf nedeni ile ortaya çıkar. Sakıncası ise ulaşılan sonucun optimal tam sayı çözümünden farklı olabilmesi ve bazen de uygun çözüm olmama durumunun ortaya çıkmasıdır, (Özgüven, 2003).

BÖLÜM 3. STOK KESME PROBLEMLERİ ve YENİ BİR ÇÖZÜM YAKLAŞIMI

3.1. Giriş

Stok kesme problemi endüstriyel alanlarda artıkların minimize edilmesi ve kaynakların verimli kullanılması amacıyla yaygın bir kullanım alanı bulmaktadır. Stok kesme problemleri, 1 boyutlu ve 2 boyutlu olmak üzere ikiye ayrılmaktadır.

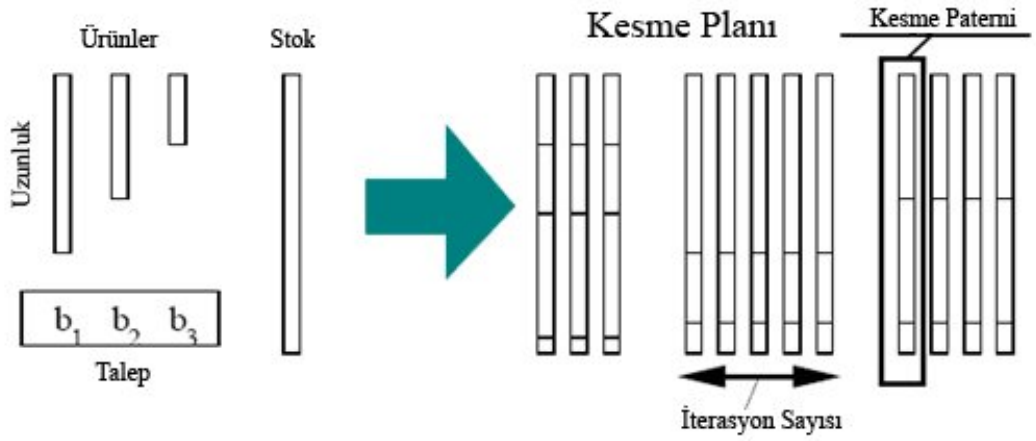
1 boyutlu stok kesme problemleri birbirinden farklı uzunluktaki nesneleri en optimum yerleştirmeyi sağlayarak, en az artık bırakacak şekilde uzun kaynak nesnelerden çıkarma işlemidir.

2 boyutlu stok kesme problemi, birbirinden farklı boyutlardaki nesneleri en optimum yerleştirmeyi sağlayarak, en az artık alan bırakacak şekilde büyük boyutlu nesnelerden çıkarma işlemidir. Bu problemin çözümü seri üretim yapan sektörlerde, maliyetleri düşürmek ve verimi arttırmak için çok değerlidir. Stok kesme probleminin hedefi müşteri taleplerine göre en optimum planı yaparak 2 boyutlu bir parçadan talebe göre malzeme çıkarmaktır.

Burada en uygun yerleştirme işlemi yaparken bazı parametreler vardır,

- Cisimler yatay veya dikey döndürülebiliyor mu,
- İçerisinden malzeme çıkarılacak parça üzerinde herhangi bir defo var mı,
- Kesilecek cisimler dikdörtgen mi yoksa n köşeli rasgele cisimler mi,
- Çıkarılacak malzemeler arasında öncelik sırası var mı,

3.2. Tek Boyutlu Stok Kesme Problemi



Şekil 3.1: Tek Boyutlu Stok Kesme Problemi

L : Stok Uzunluğu

l_i : kesilen parça uzunluğu

a_{ij} : Pattern içinde kullanılan kesilen parça adedi

x_i : Pattern adedi

Pattern Oluşturma Kuralı;

$$\sum_{i=1}^{m} l_i a_{ij} \le L$$
$$a_{ij} \le b_i, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n \tag{3.1}$$

Amaç Fonksiyonu;

$$z_{G\&G}^{1D-CSP} = \min \sum_{j=1}^{n} x_i$$
$$s.t. \quad \sum_{j=1}^{n} a_{ij} x_j \ge b_i, \quad i = 1,\ldots,m \tag{3.2}$$
$$x_j \in Z_+, \quad j = 1,\ldots,n$$

Örnek ;

Bu örnekte tipik bir stok kesme problemi formulize edilecek ve artık minimize edilecektir. Bir marangoz üzerinde çalıştığı bir iş için 3.5 metrelik 25 tahtaya, 7.5 metrelik 35 tahtaya ihtiyaç duymaktadır. Marangozun malzeme aldığı toptancı ise sadece 8 metrelik ve 12 metrelik tahtalardan satmaktadır. Bu durumda her bir tahtadan kaç tane satın almalı ve bunları nasıl kesmelidir?

Çözüm;

Gerekli optimizasyonu yapmak için öncelikle aday kümelerimizi belirlememiz gerekmektedir.

8 metrelik tahtadan;
x_1 1 adet 3.5 metrelik tahta, 4.5 metre artık
x_2 1 adet 7.5 metrelik tahta, 0.5 metre artık
x_3 2 adet 3.5 metrelik tahta, 1 metre artık

12 metrelik tahtadan;
x_4 1 adet 3.5 metre tahta, 8.5 metre artık
x_5 2 adet 3.5 metrelik tahta 5 metre artık
x_6 3 adet 3.5 metrelik tahta 1.5 metre artık
x_7 1 adet 7.5 metrelik tahta 4.5 metre artık
x_8 1 adet 3.5, 1 adet 4.5 metrelik tahta, 1 metre artık

Bu aday patternler ile toplam artık miktarını minimize etmemiz gerekmektedir.

$$F = 4.5x_1 + 0.5x_2 + x_3 + 8.5x_4 + 5x_5 + 1.5x_6 + 4.5x_7 + x_8$$

Sınırlamalarımız;

$$x_1 + 2x_3 + x_4 + 2x_5 + 3x_6 + x_8 \geq 25$$

$$x_2 + x_7 + x_8 \geq 35$$

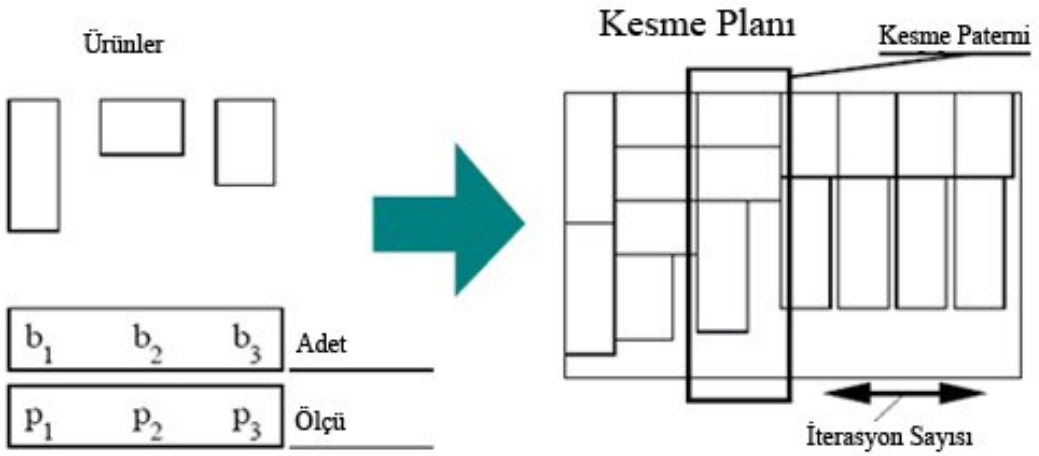
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \geq 0$$

Hazırlanan doğrusal model simpleks tablo ile çözülürse;

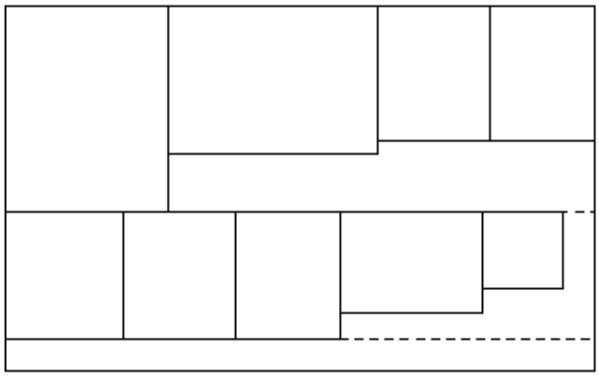
x_2 ‘den 35 Adet

x_6’ dan 8.333 Adet kesilmesi gerektiği sonucu çıkar.

3.3. İki Boyutlu Stok Kesme Problemi



Şekil 3.2: İki Boyutlu Stok Kesme Problemi



Şekil 3.3: İki Boyutlu Stok Kesme Problemi yerleşim örneği

Değişkenler : L,W, m, (l_i,w_i, p_i, b_i)

i = 1, . . . , m,

L ve W Stok malzemenin uzunluk ve genişliği

m : Her bir kesilen parça tipi adedi

l_i, w_i : Kesilen parça boyutları.

p_i : Kesilen parça adedi

b_i : Kesilecek parça adedi

a^j : Muhtemel Stripe'lar.

Muhtemel Stripe'ler

$$a^j = \left(a_{1j}, \ldots, a_{mj}\right)' \in Z_+^m, \quad j = 1, \ldots, n, \tag{3.3}$$

Aşağıdaki şartları sağlamak zorundadır,

$$\sum_{i=1}^{m} l_i a_{ij} \leq L, \quad a_{ij} \leq b_i, \quad i = 1, \ldots, m \tag{3.4}$$

$$\begin{aligned} & z^{2D-2CP} = \min \sum_{j=1}^{n} \left(\sum_{i=1}^{m} - p_i a_{ij} \right) x_j \\ s.t. \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \\ & \sum_{j=1}^{n} w\left(a^j\right) x_j \leq W \\ & x_j \in Z_+, \quad j = 1, \ldots, n \end{aligned} \tag{3.5}$$

Burada en uygun yerleştirme işlemi yaparken bazı parametreler vardır,

- Cisimlerin yatay veya dikey döndürülebilir olması.
- İçerisinden malzeme çıkarılacak parça üzerinde herhangi bir defo olması.
- Kesilecek cisimler dikdörtgen yada n köşeli rasgele cisimler olması.
- Çıkarılacak malzemeler arasında öncelik sırası olması.

3.4. Uygulama

Uygulamada kaynak stokların ve sipariş ürünlerin en boy ve adet bilgileri girilerek en az fire verecek biçimde yerleşiminin hesaplanarak ekrana çizdirilmesi üzerine çalışılmıştır.

Uyuglamada 2 boyutlu stok kesme problemi 2 boyutlu yerleşim problemi gibi değerlendirerek bilinen klasik yöntemleri kullanmadan yeni bir yaklaşımla çözüm aranmıştır.

Uygulama Borland Delphi 5 derleyicide, Windows Xp İşletim sisteminde Pentium Core 2 Duo işlemcili 2GB belleğe sahip olan bir bilgisaya üzerinde geliştirilmiştir. Geliştirilen yeni yaklaşımla hazırlanmış uygulama programı klasik yöntemlerle çözüm arayan farklı bir stok kesme programıyla ortak verilerle test edilerek hız ve yerleşim yüzdesi bakımından karşılaştırılmıştır.

3.5. Yeni Çözüm Yöntemi

Tanımlamalar;

Kaynak: İçerisinden Kesim Yapılacak Büyük Levha.
Sipariş: Kaynak İçerisine En Az Fire Verecek Şekilde Yerleştirilecek Parça.
Ken: Kaynak Eni
KBoy: Kaynak Boyu
KMiktar: Kaynak Stok Miktarı
SEn: Sipariş Eni
SBoy: Sipariş Boyu
SMiktar: Sipariş Stok Miktarı
n: Sipariş Denklem Katsayısı

ArtıkEn: Yerleşim Sonucu Kaynaktan Artan Dikdörtgenin Eni
ArtıkBoy: Yerleşim Sonucu Kaynaktan Artan Dikdörtgenin Boyu

Çözüm aşamaları;

Siparişler ve Kaynaklar yerleşim yönüne göre büyükten küçüğe doğru sıralanır.

$SEn[0] \geq SEn[1] \geq SEn[2] \geq .. \geq SEn[i]$

i = Sipariş ürün indisi

KEn[0]≥KEn[1] ≥KEn[2] ≥..≥KEn[j]
j = Kaynak ürün indisi

for j=0 to KaynakSayısı
SEn[0]*n[0]+SEn[1]*n[1]+..+SEn[i]*n[i]<=Ken[j]
SBoy[i]≤KBoy[j]
n[i] ≤SMiktar[i]
Denklemi n>=0 ve Tamsayı olmak şartıyla çözülür ve yerleştirilen n[i]'er Smiktar[i]'den çıkartılır.

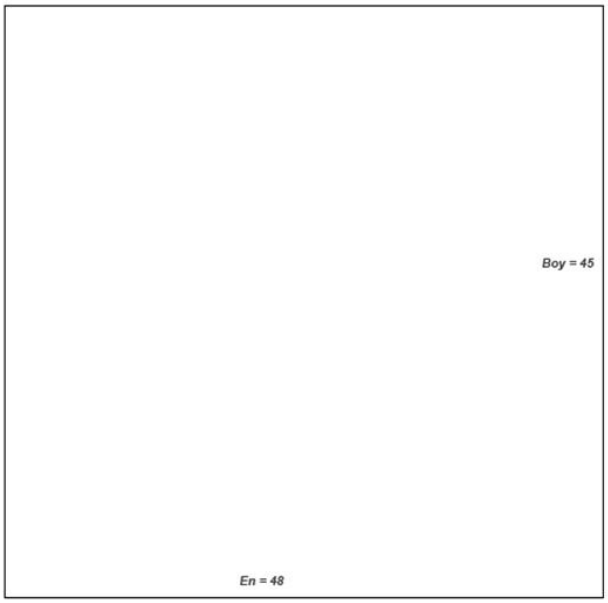
Bu denklem için birden çok çözüm çıkabileceğinden bu çözümlerden SEn elemanı en büyük ve en büyük n sayısına sahip olan çözüm seçilir. (En az fire veren)

Daha sonra çözüm dizisi katsayılarıyla birlikte Kaynak üzerine yerleştirilir. Yerleşim sonucunda kalan boşlukların her biri bir dikdörtgen olacak şekilde bölünür.
Yeni oluşturulan tüm artık dikdörtgenleri birer yeni kaynakmış gibi tekrardan sipariş yerleştirme işlemine sokulur. Bu işlem recursive olarak tüm siparişler bitene kadar veya tam yerleşim yapılana kadar veya kaynağa yeni bir siparişin sığmayacağı anlaşılana kadar tekrarlanır.

Örnek Problem;

Eni 48cm, boyu 45cm olacak şekilde bir kaynağımız olduğunu varsayalım.

Kaynak;



Şekil 3.4: Üzerinde kesme işlemi yapılacak kaynak stok

Siparişler;

Tablo 3.1: Kesme işleminin yapılacağı sipariş ürünlerin listesi

Açıklama	**En**	**Boy**	**Miktar**
0	13	11	5
1	12	17	14
2	9	7	9
3	7	7	57
4	5	4	37
5	4	3	45

İlk Yerleşim (Kaynak = 48*45);

Yerleşim denklemi çözümü;

13*n[0]+12* n[1]+9* n[2]+7* n[3]+5* n[4]+4* n[5]<=48

Kısıtlar;

n[i]; 0 ve pozitif tamsayılar olmalı.

n[0]<=5,
n[1]<=14,
n[2]<=9,
n[3]<=57,
n[4]<=37,
n[5]<=45 olmalı.

Sn[i]*n[i] değişkenleri kısıtlar dahilinde hesap matrisine yerleştirilir.

Tablo 3.2: Yerleşim taramasının yapılacağı hedef uzay matrisi

Adet	**Sn[0]*n[0]**	**Sn[1]*n[1]**	**Sn[2]*n[2]**	**Sn[3]*n[3]**	**Sn[4]*n[4]**	**Sn[5]*n[5]**
0	0	0	0	0	0	0
1	13	12	9	7	5	4
2	26	24	18	14	10	8
3	39	36	27	21	15	12
4	(52>48) İpt	48	36	28	20	16
5			45	35	25	20
6				42	30	24
7					35	28
8					40	32
9					45	36
10						40
11						44
12						48

Matris soldan sağa ve aşağıdan yukarıya taranarak çözüm aranır.

39+48>48
39+36>48

39+24>48

39+12>48

39+0<48 fire = 9

39+0+45>48

39+0+36>48

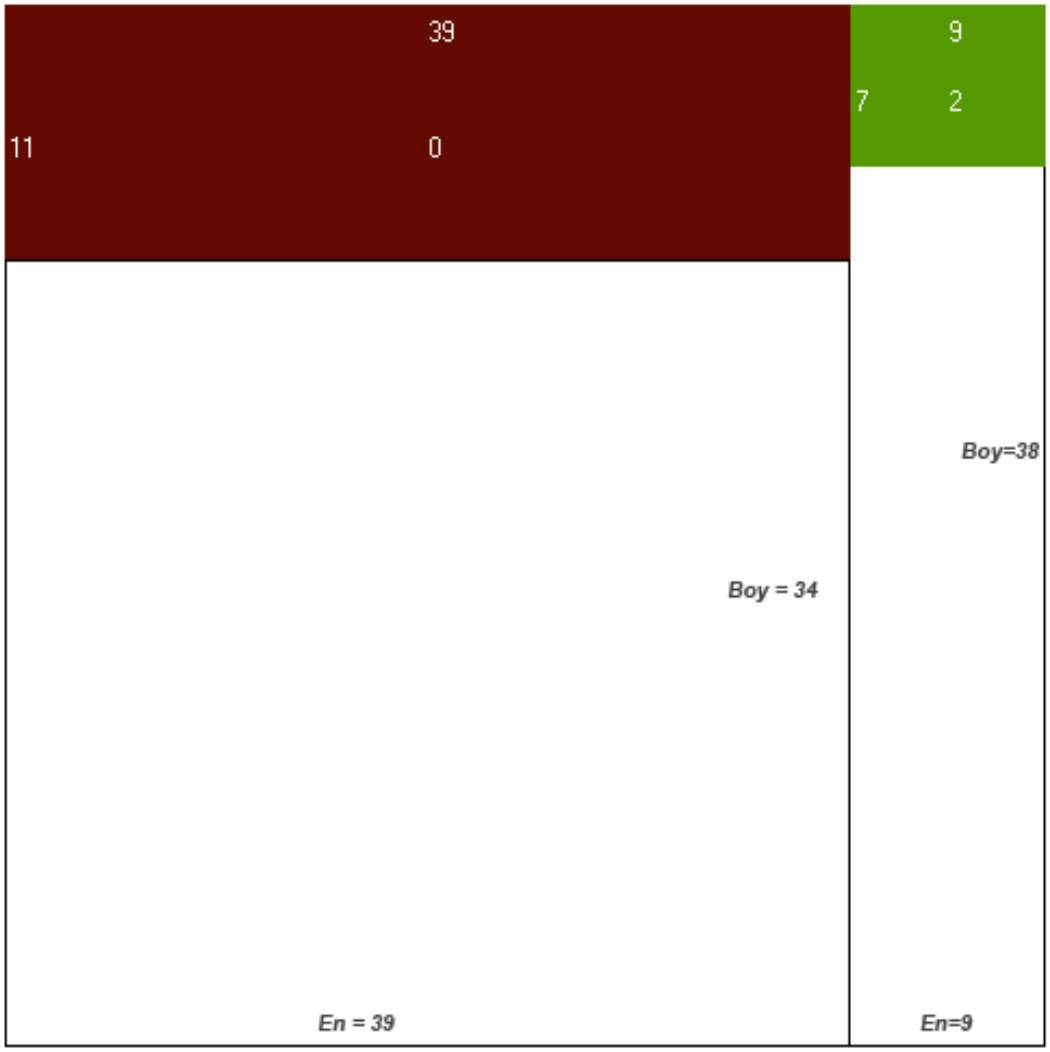
39+0+27>48

39+0+18>48

39+0+9<=48 fire=0 yerleşim sonucu.

13*3+9*1=48 çözümünden;

SEn[0]*3+SEn[2]*1<=48 Şartı Sağlanmaktadır.



Şekil 3.5: Kaynak stoğa ilk adımda yapılan yerleşim.

Denklem çözümü yukarıda gösterildiği gibi kaynak üzerine yerleştirilmiştir. Yerleşim sonucunda 39x34 lük ve 9x38 lik iki adet firemiz artmıştır. Bu artmış olan dikdörtgenler yeniden sanki birer kaynakmış gibi yerleşim işlemine sokularak ilerlemeye devam edilir.

İkinci Yerleşim (Kaynak = 39*34);

Yerleşim denklemi çözümü;
13*n[0]+12* n[1]+9* n[2]+7* n[3]+5* n[4]+4* n[5]<=39

Kısıtlar;
n[i]; 0 ve pozitif tamsayılar olmalı.
n[0]<=2 (3 Adet Kullanıldı)
n[1]<=14,
n[2]<=8, (1 Adet Kullanıldı)
n[3]<=57,
n[4]<=37,
n[5]<=45 olmalı.

Sn[i]*n[i] değişkenleri kısıtlar dahilinde hesap matrisine yerleştirilir.

Tablo 3.3: Adım 2 için yerleşim taramasının yapılacağı hedef uzay matrisi

Adet	**Sn[0]*n[0]**	**Sn[1]*n[1]**	**Sn[2]*n[2]**	**Sn[3]*n[3]**	**Sn[4]*n[4]**	**Sn[5]*n[5]**
0	0	0	0	0	0	0
1	13	12	9	7	5	4
2	26	24	18	14	10	8
3		36	27	21	15	12
4		48	36	28	20	16
5			45	35	25	20
6				42	30	24
7					35	28
8					40	32
9					45	36
10						40
11						44
12						48

Matris soldan sağa ve aşağıdan yukarıya taranarak çözüm aranır.

26<=39 fire=13
26+48>39
26+36>39
26+24>39
*26+12<=39 fire=1
26+12+45>39
..
26+12+9>39
26+12+0<=39 fire=1
26+12+0+42>39
..
26+12+0+7>39
26+12+0+0<=39 fire=1
26+12+0+0+45>39
..
26+12+0+0+5>39
26+12+0+0+0<=39 fire=1
26+12+0+0+0+48>39
..
26+12+0+0+4>39
26+12+0+0+0<=39 fire=1 (tarama bitti fakat dikey adıma devam edilir (* noktasının üzerine)

26+0<=39 fire=13
26+0+45>39
..
26+0+9<=39 fire=4
26+0+9+42>39
..
26+0+9+7>39
26+0+9+0<=39 fire=4

26+0+9+0+45>39

..

26+0+9+0+5>39

26+0+9+0+0<=39 fire=4

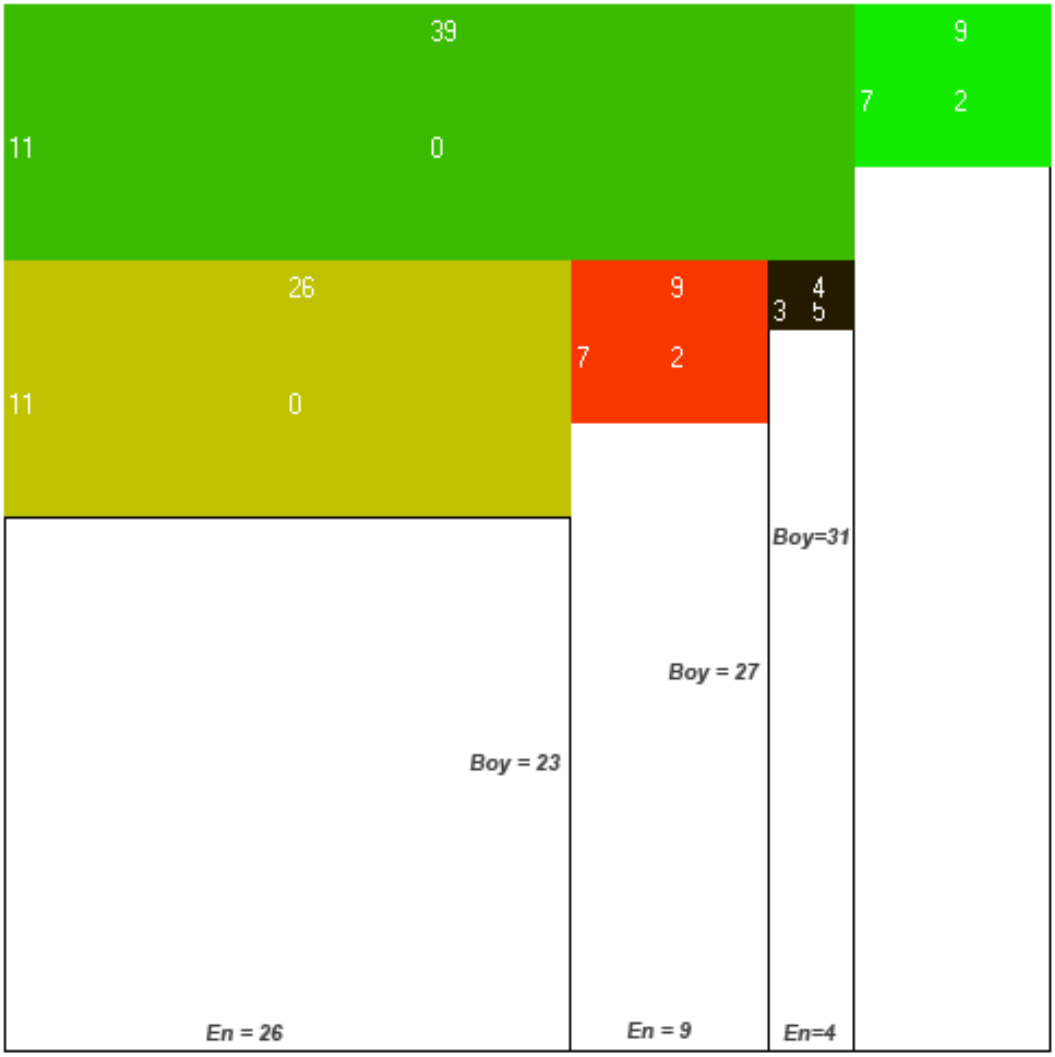
26+0+9+0+0+48>39

..

26+0+9+0+0+4<=39 fire=0 yerleşim sonucu.

13*2+9*1+4*1=39 çözümünden;

SEn[0]*2+ SEn[2]*1+ SEn[5]*1 <=39 Şartı Sağlanmaktadır.



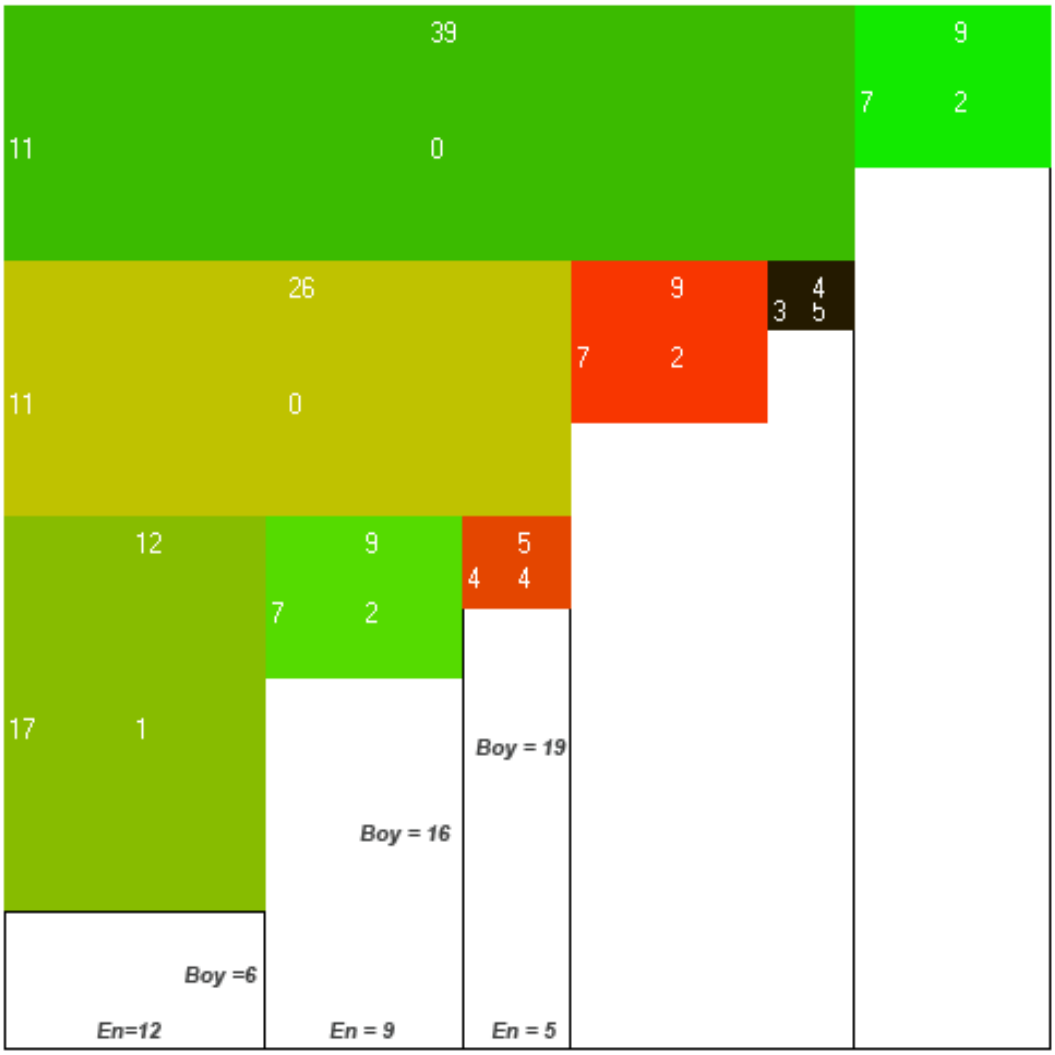
Şekil 3.6: Adım 2 sonucunda kaynak stoğa yapılan yerleşim.

Denklem çözümü yukarıda gösterildiği gibi kaynak üzerine yerleştirilmiştir. Yerleşim sonucunda 26x23 lük ve 9x27 ve 4*31 lik üç adet firemiz artmıştır.

Üçüncü Yerleşim (Kaynak = 26*23);

12*1+9*1+5*1=26 çözümünden;

SEn[1]*2+ SEn[2]*1+ SEn[4]*1 <=26 Şartı Sağlanmaktadır.

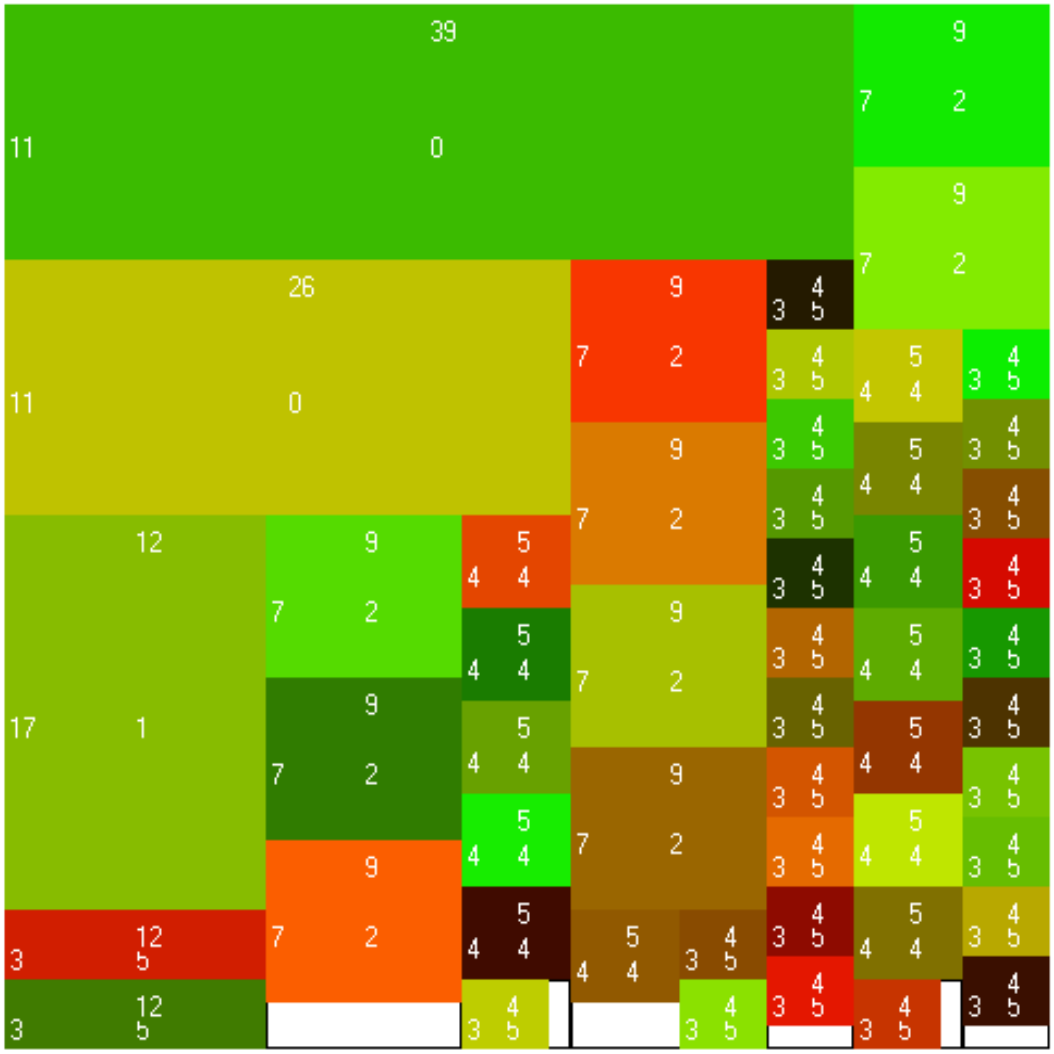


Şekil 3.7: Adım 3 sonucunda kaynak stoğa yapılan yerleşim.

Denklem çözümü yukarıda gösterildiği gibi kaynak üzerine yerleştirilmiştir. Yerleşim sonucunda 12x6 ve 9x16 ve 5*19 luk üç adet firemiz artmıştır.

Algoritma recursive olarak yerleşim olmama, siparişin tamamlanması ya da kaynağın tükenmesi sonuçları doğana kadar doldurma işlemine devam etmektedir.

Yerleşim Sonucunda;



Şekil 3.8: Tüm adımlar tamamlandığında kaynak stoğa yapılan yerleşim.

elde edilir.

BÖLÜM 4. PERFORMANS DEĞERLENDİRMESİ

4.1. Yeni Çözüm Yaklaşımının İşlem Külfeti

Bölüm 3'de yeni çözüm yaklaşımının anlatıldığı örnek problemin;

Kaynak;
En: 48cm, boy: 45cm

Siparişler;

Tablo 4.1: Kesme işleminin yapılacağı sipariş ürünlerin listesi

Açıklama	**En**	**Boy**	**Miktar**
0	13	11	5
1	12	17	14
2	9	7	9
3	7	7	57
4	5	4	37
5	4	3	45

Çözümüne 14461 adet matematiksel operasyon (4 işlem, mantıksal kontrol, taşıma işlemleri) sonucunda, 109 ms 'de ulaşılmıştır.

Aynı problem, aynı şartlarda klasik yöntem ile çözüldüğünde çözüme 2485 ms'de ulaşılmıştır. Bu problem özelinde zaman üzerinden genelleme yaparsak, klasik yöntem ile çözüme yaklaşık 329684 adet matematiksel operasyon sonucunda ulaşılmıştır.

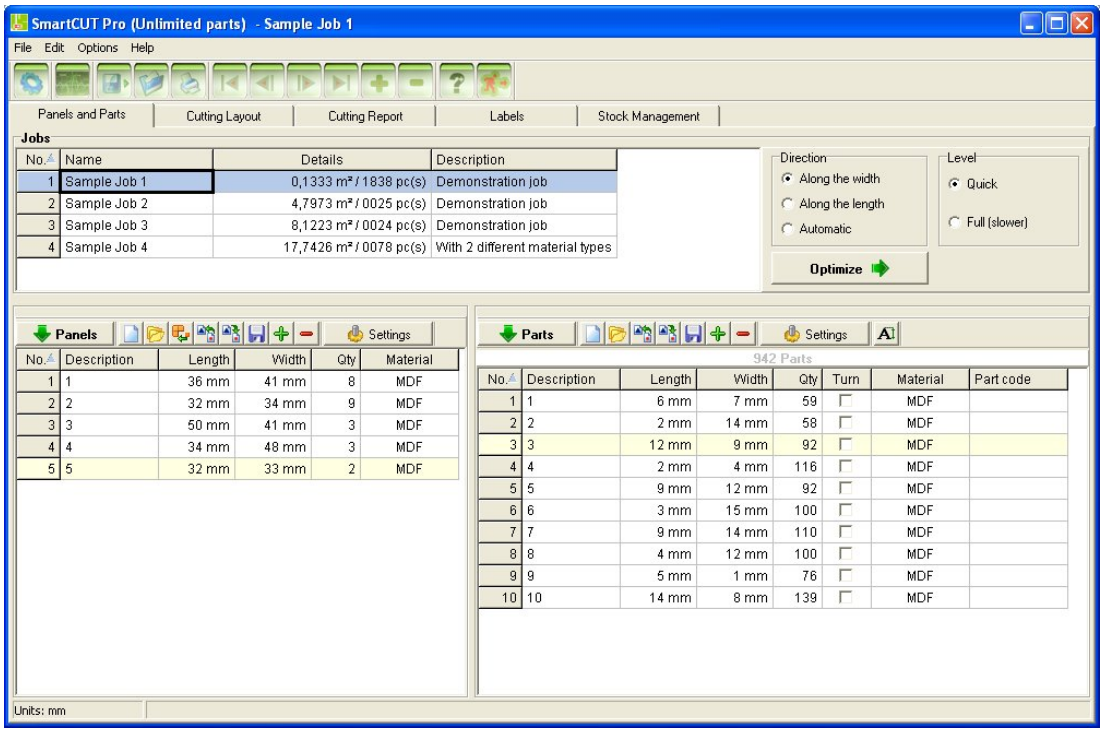
4.2. SmartCutPro Yazılımı İle Karşılaştırılması

SmartCutPro yazılımı stok kesme problemleri çözümünde piyasada yaygın olarak tercih edilen ve çözüme en hızlı ulaştığını vurgulayan bir optimizasyon yazılımıdır. Karşılaştırma işlemi aynı ölçülerde 5 çeşit, 25 adet kaynak, 10 çeşit, 942 parça sipariş üzerinde yapılmıştır.

SmartCutPro yazılımının;

- Yatay yönde hızlı
- Dikey yönde hızlı
- Yatay + dikey yönde hızlı
- Yatay yönde yavaş
- Dikey yönde yavaş
- Yatay + dikey yönde yavaş

Tarama modları ile yeni çözüm yaklaşımının yerleşim yüzdesi ve çözüm süreleri bakımından karşılaştırması yapılmıştır.

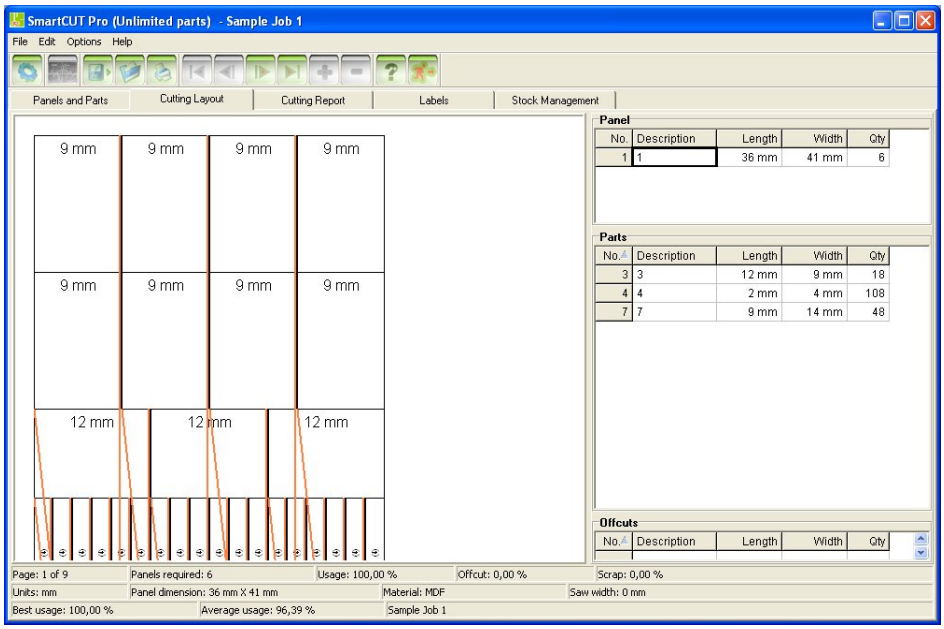


Şekil 4.1: SmartCutPro kaynak ve sipariş ürünlerin tanımlama ekranı

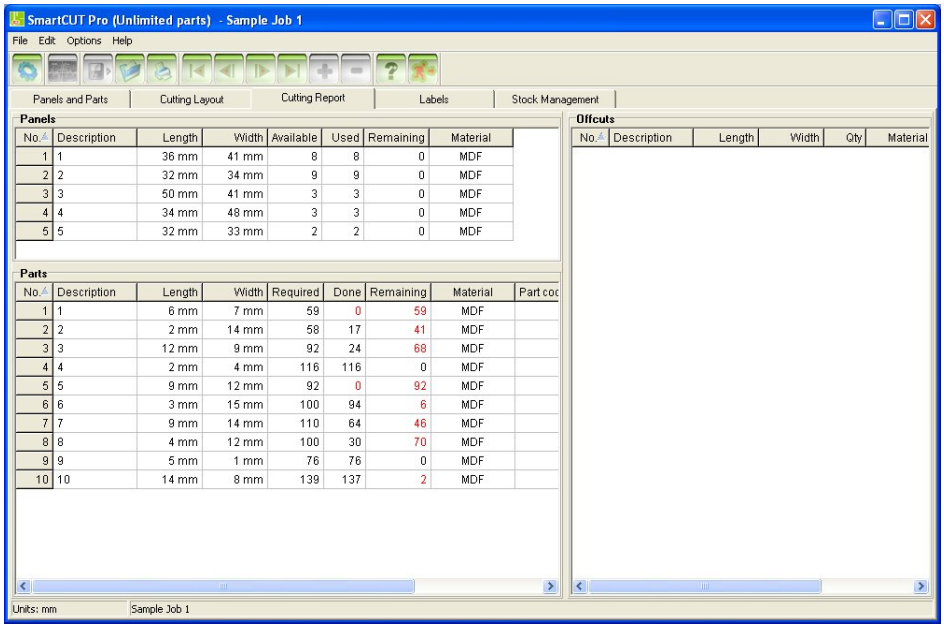
SmartCutPro ile yatay yönde hızlı modda tarama;

Tarama Süresi : 4sn

Yerleştirme Yüzdesi : %96,39



Şekil 4.2: SmartCutPro ile yatay yönde hızlı modda tarama yerleşim sonucu ekranı

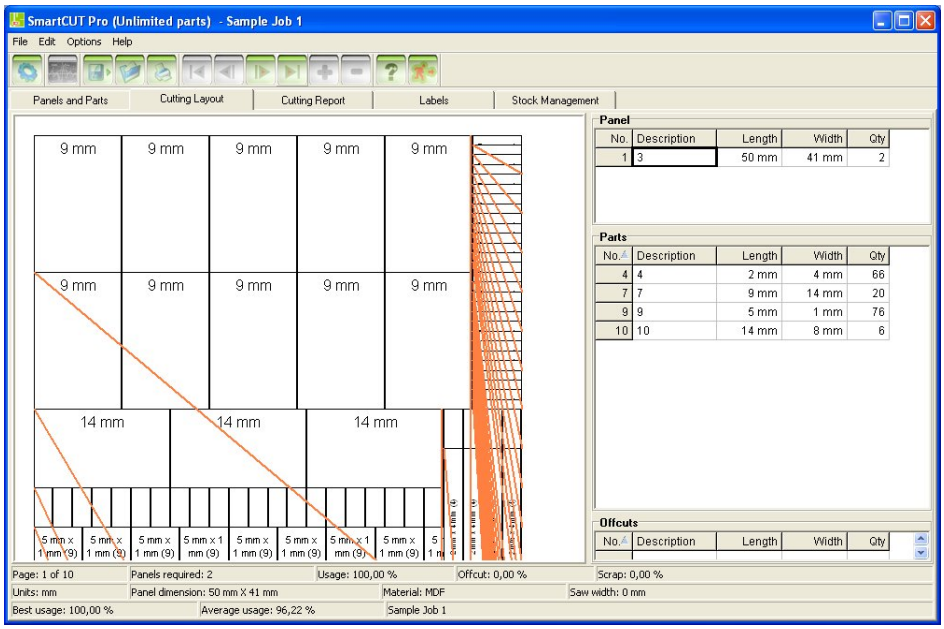


Şekil 4.3: SmartCutPro ile yatay yönde hızlı modda tarama kalan stok durumu ekranı

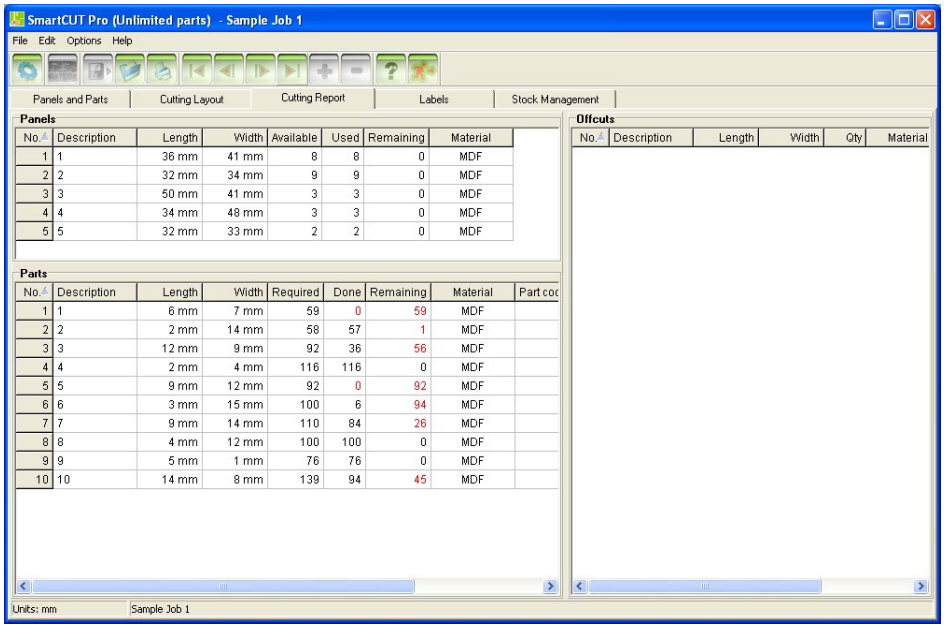
SmartCutPro ile dikey yönde hızlı modda tarama;

Tarama Süresi : 3,5sn

Yerleştirme Yüzdesi : %96,22



Şekil 4.4: SmartCutPro ile dikey yönde hızlı modda tarama yerleşim sonucu ekranı

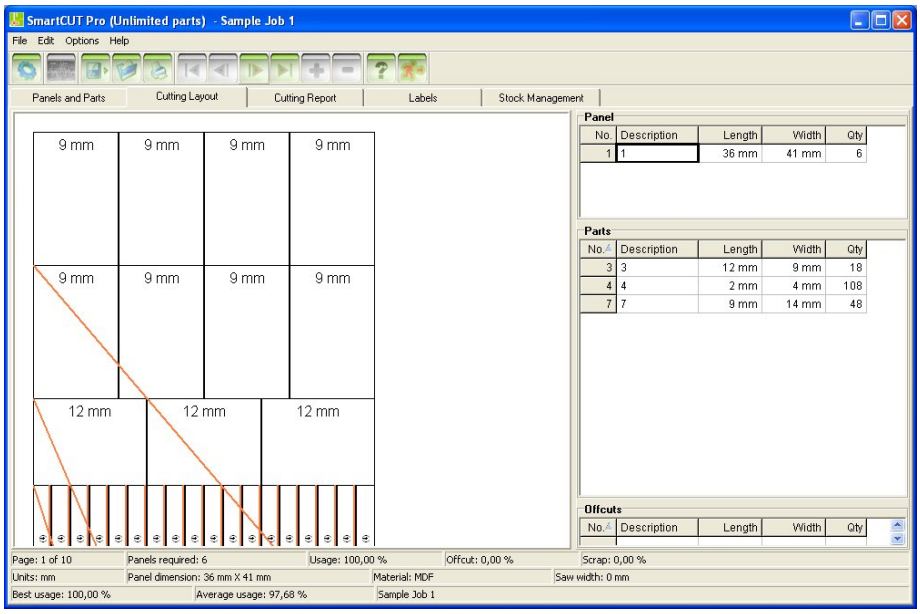


Şekil 4.5: SmartCutPro ile dikey yönde hızlı modda tarama kalan stok durumu ekranı

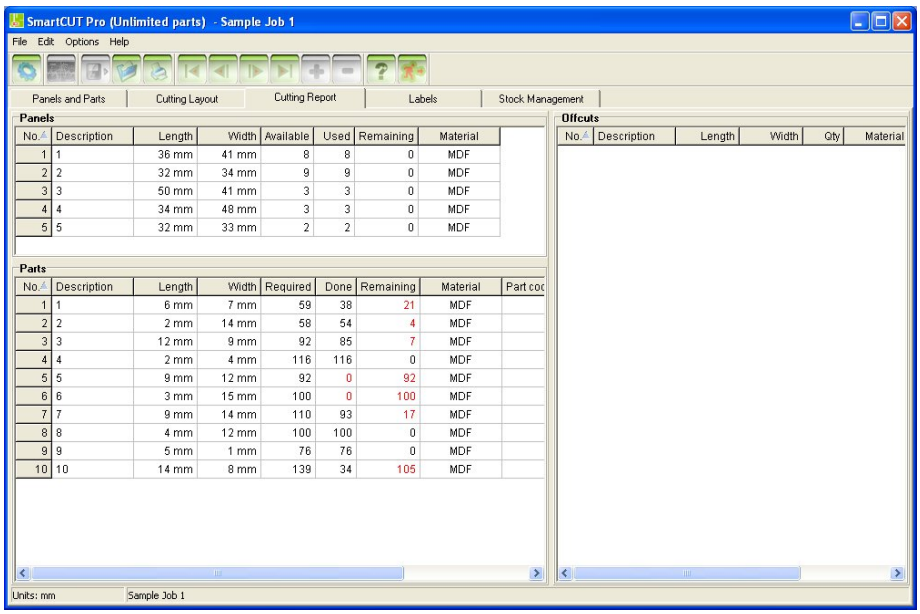
SmartCutPro ile yatay+dikey yönde hızlı modda tarama;

Tarama Süresi : 4sn

Yerleştirme Yüzdesi : %97,68



Şekil 4.6: SmartCutPro ile yatay+dikey yönde hızlı modda tarama yerleşim sonucu ekranı

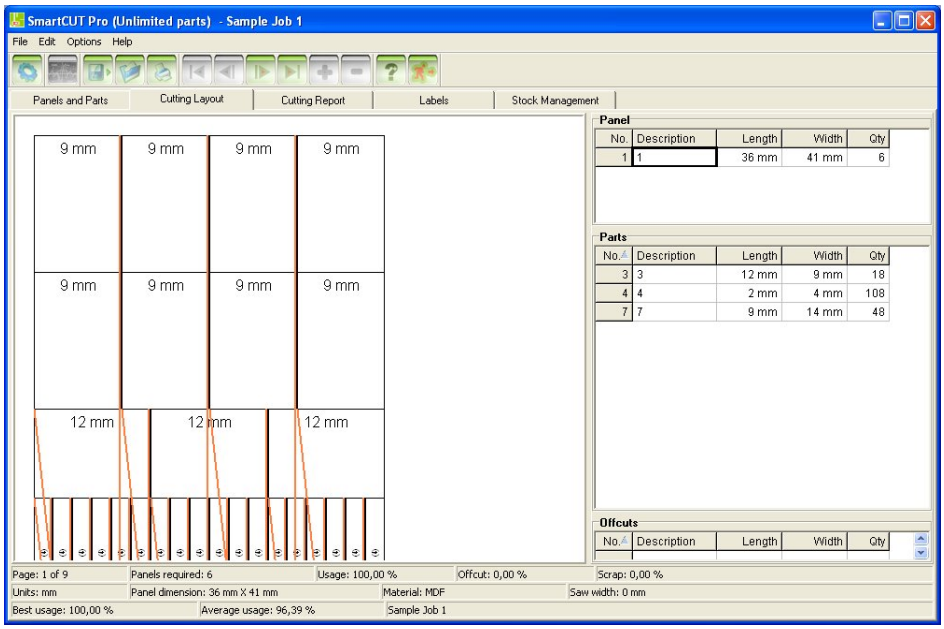


Şekil 4.7: SmartCutPro ile yatay+dikey yönde hızlı modda tarama kalan stok durumu ekranı

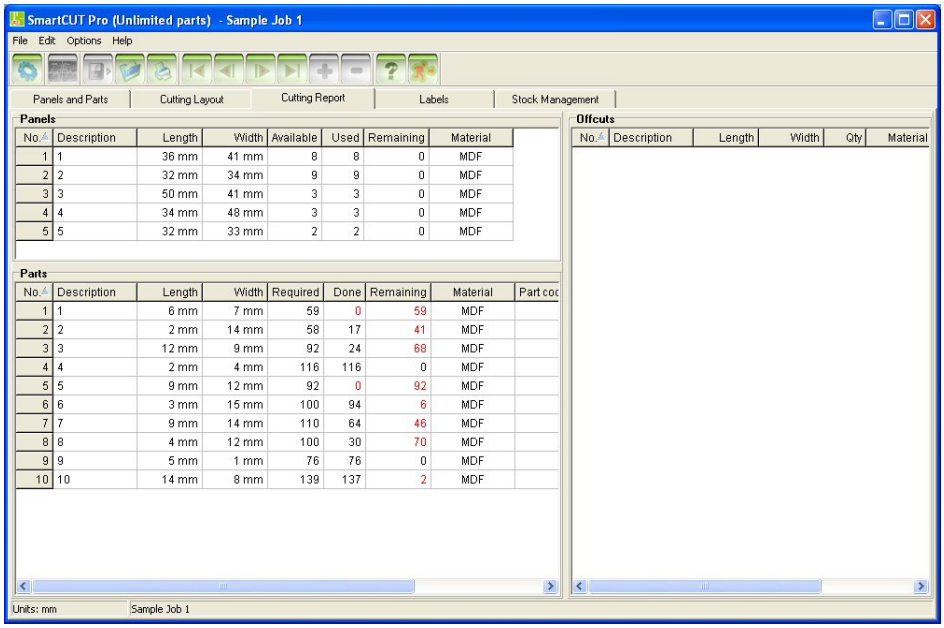
SmartCutPro ile yatay yönde yavaş modda tarama;

Tarama Süresi : 76sn

Yerleştirme Yüzdesi : %96,39



Şekil 4.8: SmartCutPro ile yatay yönde yavaş modda tarama yerleşim sonucu ekranı

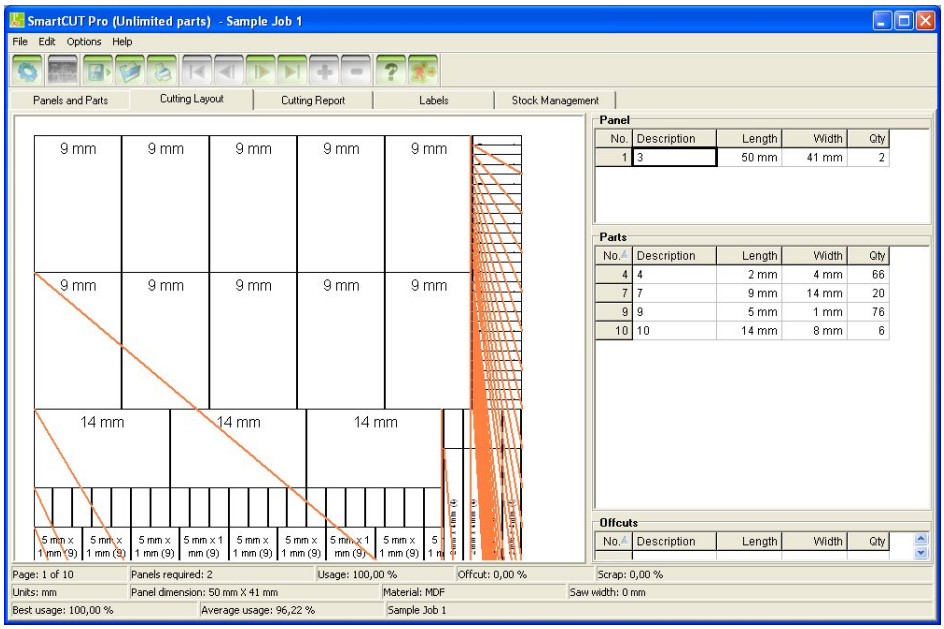


Şekil 4.9: SmartCutPro ile yatay yönde yavaş modda tarama kalan stok durumu ekranı

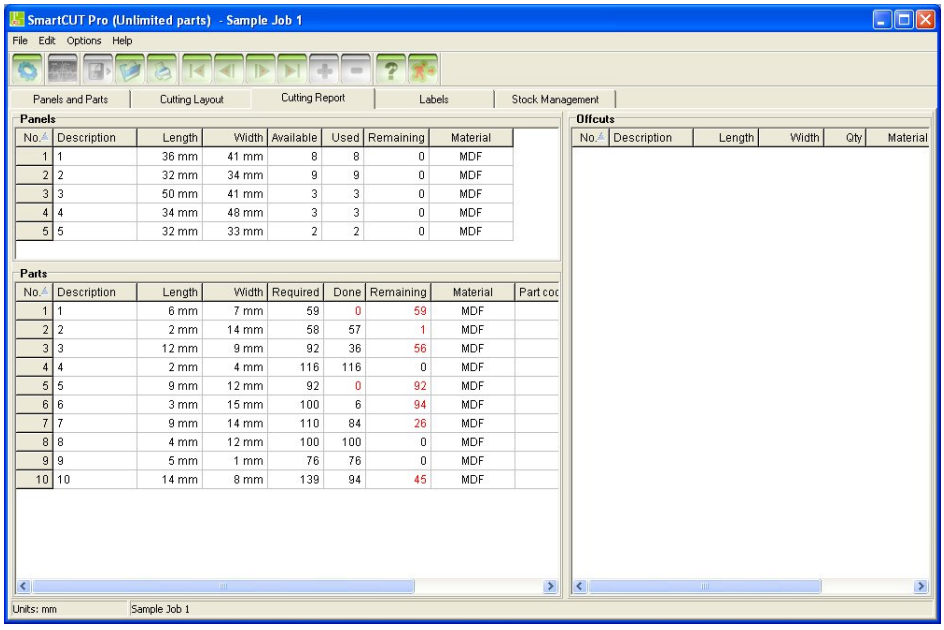
SmartCutPro ile dikey yönde yavaş modda tarama;

Tarama Süresi : 23sn

Yerleştirme Yüzdesi : %96,22



Şekil 4.10: SmartCutPro ile dikey yönde yavaş modda tarama yerleşim sonucu ekranı

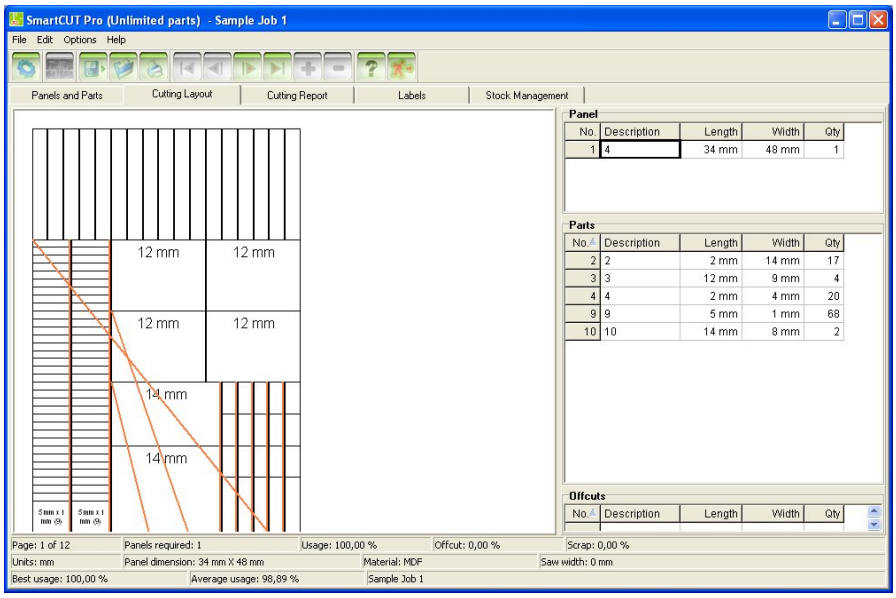


Şekil 4.11: SmartCutPro ile dikey yönde yavaş modda tarama kalan stok durumu ekranı

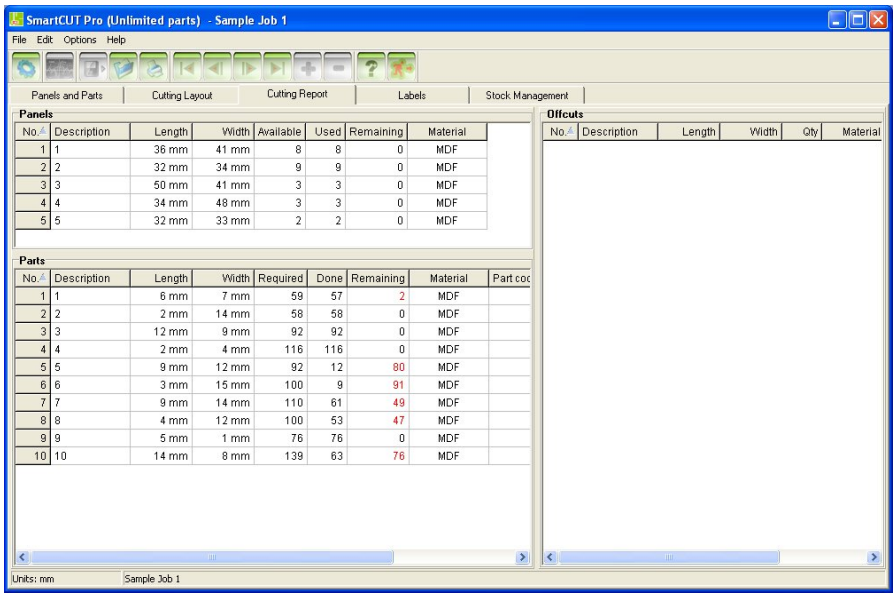
SmartCutPro ile yatay+dikey yönde yavaş modda tarama;

Tarama Süresi : 38sn

Yerleştirme Yüzdesi : %98,89



Şekil 4.12: SmartCutPro ile yatay+dikey yönde yavaş modda tarama yerleşim sonucu ekranı

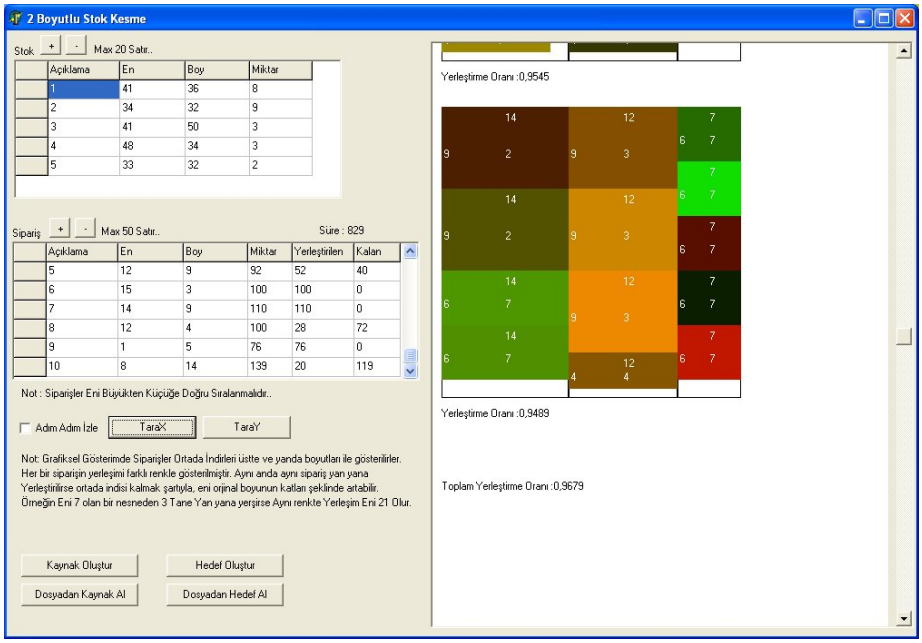


Şekil 4.13: SmartCutPro ile yatay+dikey yönde yavaş modda tarama kalan stok durumu ekranı

Geliştirilen yeni yöntem ile yatay yönde tarama;

Tarama Süresi : 0,83sn

Yerleştirme Yüzdesi : %96,76

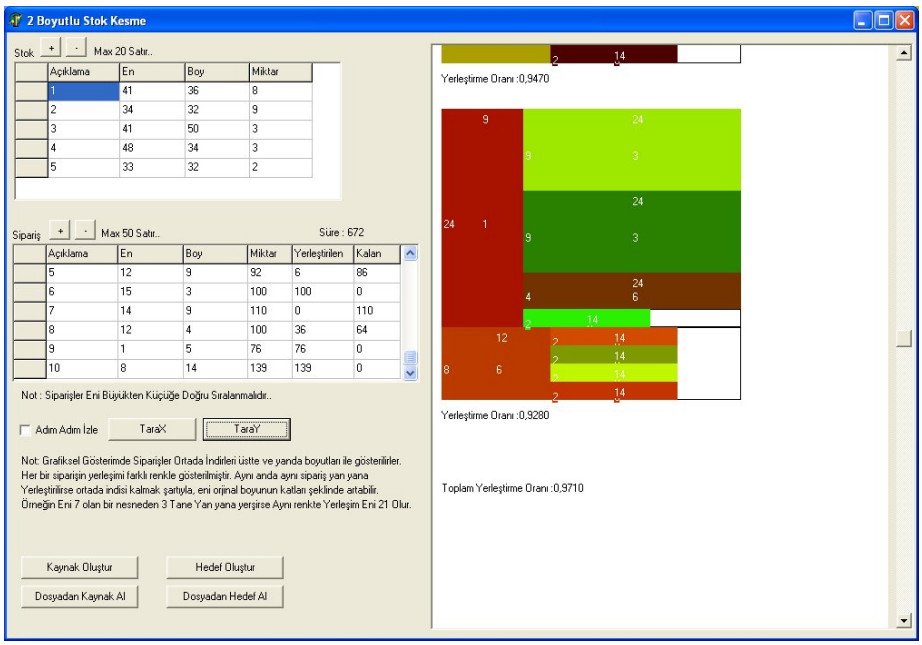


Şekil 4.14: Geliştirilen yeni yöntem ile yatay yönde tarama yerleşim sonucu ekranı

Geliştirilen yeni yöntem ile dikey yönde tarama;

Tarama Süresi : 0,67sn

Yerleştirme Yüzdesi : %97,10



Şekil 4.15: Geliştirilen yeni yöntem ile dikey yönde tarama yerleşim sonucu ekranı

SmartCutPro ile Hızlı Mod Ortalama Sonuçları;

Ortalama Tarama Süresi : 8,83sn

Ortalama Yerleştirme Yüzdesi : 96,76

SmartCutPro ile Yavaş Mod Ortalama Sonuçları;

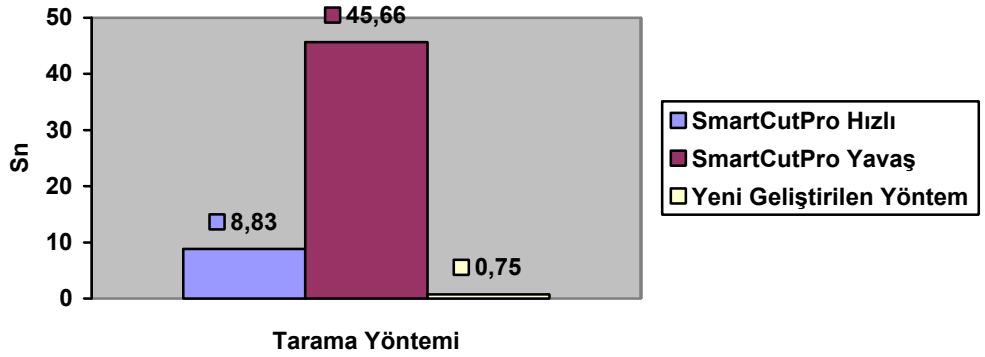
Ortalama Tarama Süresi : 45,66sn

Ortalama Yerleştirme Yüzdesi : 97,16

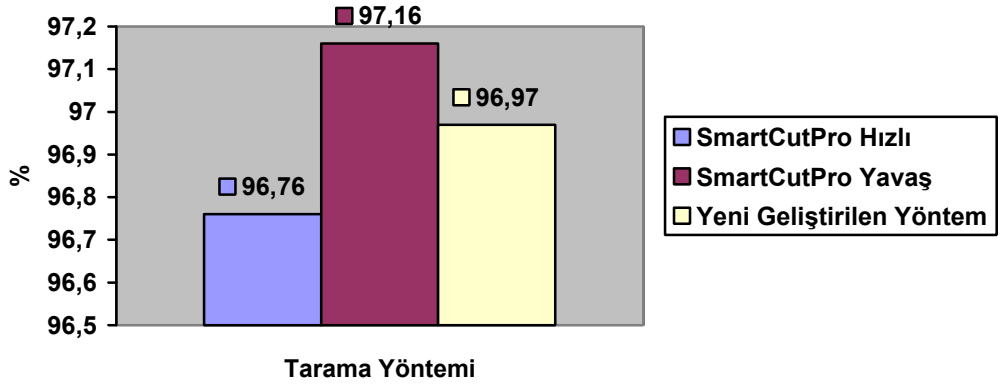
Geliştirilen Yöntemin Ortalama Sonuçları;

Ortalama Tarama Süresi : 0,75sn

Ortalama Yerleştirme Yüzdesi : 96,93



Şekil 4.16: Tarama zaman grafiği



Şekil 4.17: Tarama yerleşim yüzdesi grafiği

4.3. Yeni Çözüm Yaklaşımı ve Simpleks Yöntemi ile Stok Kesme Problemi Çözümüne Monte Carlo Simülasyonunun Uygulanması

Uygulama Borland Delphi 5 derleyicide, stok kesme problemine klasik yöntemlerle çözüm arayan simpleks yöntemini kullanan bileşen ile, Windows Xp İşletim sisteminde Pentium Core 2 Duo işlemcili 2GB belleğe sahip olan bir bilgisaya üzerinde geliştirilmiştir.

Simülasyonda düzgün dağılım ile rasgele üretilmiş kaynaklardan ve hedeflerden oluşan 100 adet problemin yeni çözüm yaklaşımıyla ve Simpleks bileşeniyle çözülüp performansının ölçülmesi amaçlanmıştır.

Kaynak Sınırları;

Simülasyonda düzgün dağılım ile rasgele üretilen kaynaklar 5 farklı boyutta üründen oluşmaktadır. Her bir kaynak ürünün eni ve boyu 30 – 55 cm arasında olacak şekilde, miktarı ise 1 - 10 adet arasında olacak şekilde sınırlandırılmıştır.

Dolayısıyla simülasyonun her adımında en az 5, en çok 50 adet kaynak ürün içerisinden kesme işlemi yapılabilmektedir. Toplam kaynak alanı 4500 – 151250 cm^2 arasında olmaktadır.

Hedef Sınırları;

Simülasyonda düzgün dağılım ile rasgele üretilen hedefler 20 farklı boyutta üründen oluşmaktadır. Her bir hedef ürünün eni ve boyu 1 – 15 cm arasında olacak şekilde, miktarı ise 50 - 150 adet arasında olacak şekilde sınırlandırılmıştır.

Dolayısıyla simülasyonun her adımında en az 1000, en çok 3000 adet hedef ürün ile kesme işlemi yapılabilmektedir. Toplam hedef alanı 1000 – 675000 cm^2 arasında olmaktadır.

Tablo 4.2: Monte Carlo simülasyonu sonuçları

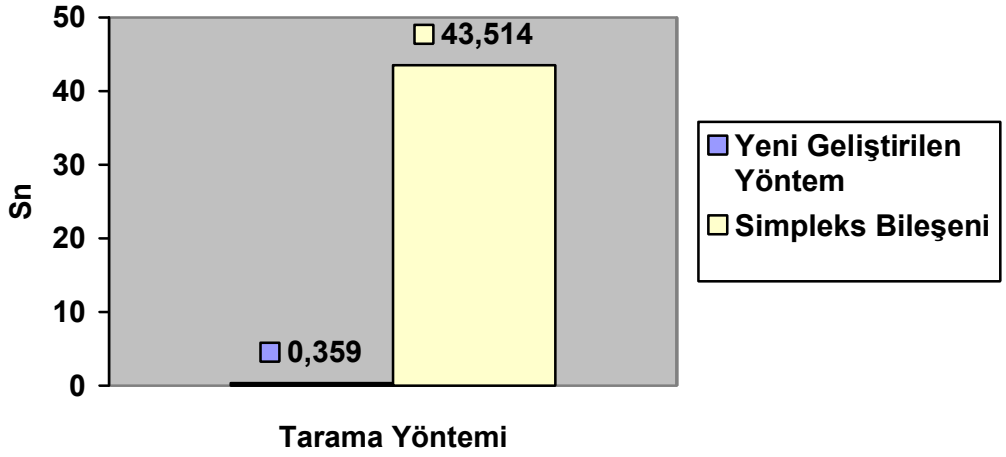
Problem No	Yeni Çözüm Yaklaşımı Süre (sn)	Yeni Çözüm Yaklaşımı Yerleşim Oranı (%)	Simpleks Süre (sn)	Simpleks Yerleşim Oranı (%)
1	0,360	99,5893	29,328	96,7382
2	0,625	99,0529	46,688	99,6358
3	0,266	97,6770	32,500	93,7585
4	0,344	97,2314	52,921	97,6692
5	0,344	99,6575	50,078	96,8799
6	0,266	99,3824	26,250	96,5871
7	0,391	99,7350	48,359	97,8360
8	0,421	99,5117	64,485	98,7471
9	0,265	99,9212	38,610	99,4110
10	0,265	99,8615	27,969	99,0716
11	0,359	99,2326	34,969	93,3651
12	0,344	97,3003	27,359	97,6474
13	0,219	99,4952	20,687	95,2267
14	0,125	97,6220	25,000	95,8498
15	0,328	98,7463	47,422	96,9130
16	0,265	99,0344	37,250	95,2720
17	0,266	96,5325	40,828	98,2127
18	0,281	99,6016	32,532	95,9127
19	0,234	97,0172	55,312	98,9342
20	0,219	97,6662	38,719	92,4258
21	0,391	99,6636	23,750	97,3827
22	0,375	98,6484	26,016	99,1452
23	0,438	99,8469	33,656	98,3615
24	0,312	98,4399	53,391	99,2837
25	0,328	99,1046	47,594	94,8545
26	0,375	99,6518	32,781	99,5858
27	0,219	98,3436	41,141	97,8711
28	0,219	99,5458	55,610	97,8905
29	0,266	96,6502	43,172	99,4828
30	0,203	95,1832	34,750	96,5568
31	0,656	98,4992	53,765	98,8529
32	0,453	98,4920	43,110	98,5211
33	0,203	94,5994	53,218	98,3214
34	0,594	99,1196	50,047	98,9282
35	0,187	99,0434	35,422	98,7997
36	0,203	97,3061	29,563	97,8031
37	0,250	96,5995	43,657	99,3406
38	0,156	98,8292	19,344	96,7429
39	0,328	98,4368	51,031	97,9859
40	0,407	98,5015	46,406	98,2656
41	0,547	98,2552	45,781	95,6042
42	0,468	98,2978	41,563	97,1780
43	0,282	99,4405	37,359	91,9876
44	0,375	98,6649	49,500	98,4003
45	0,391	99,4180	43,797	97,7273
46	0,296	99,0504	38,907	96,3409
47	0,313	98,0824	46,969	97,6561
48	0,468	98,2340	54,282	94,7265
49	0,297	98,2517	39,531	97,4407
50	0,422	99,5573	42,156	97,3491

51	0,406	98,7514	45,719	96,8818
52	0,485	98,6958	60,922	97,1262
53	0,281	98,2683	36,484	99,0817
54	0,609	99,3269	40,625	98,2943
55	0,203	97,3259	37,078	98,4053
56	0,360	98,4609	50,844	98,8247
57	0,360	98,5311	44,500	96,1703
58	0,343	97,0867	45,735	99,5170
59	0,234	99,1585	43,953	95,1134
60	0,328	99,1090	51,563	97,4765
61	0,438	99,0519	39,703	97,7246
62	0,406	98,5323	34,485	90,9327
63	0,328	98,9279	44,719	98,3963
64	0,156	95,7210	38,719	98,2196
65	0,359	99,9516	42,719	98,8182
66	0,375	99,2312	45,516	98,4932
67	0,234	99,2496	47,532	99,2955
68	0,218	99,7759	25,375	98,6530
69	0,172	99,7377	30,156	97,4047
70	0,484	99,4830	45,625	99,7409
71	0,500	98,6969	43,234	97,8960
72	0,594	99,0434	51,906	97,5701
73	0,375	99,8271	36,641	99,9188
74	0,125	95,8816	22,672	95,2584
75	0,578	98,1356	64,844	96,4857
76	0,156	97,0594	29,610	95,9648
77	0,781	98,6169	60,625	99,2705
78	0,343	99,6497	47,407	97,1809
79	0,125	99,7073	35,531	99,6676
80	0,578	99,6066	43,609	95,7566
81	0,938	97,7402	84,187	95,3875
82	0,828	98,8018	89,453	96,9829
83	0,469	95,4367	79,687	99,3147
84	0,625	99,6039	66,906	96,3585
85	0,329	97,6396	47,250	91,2904
86	0,141	99,0504	26,547	92,2841
87	0,329	99,8341	35,453	95,2488
88	0,547	99,1794	67,953	86,4140
89	0,375	99,5947	44,735	98,8761
90	0,297	99,0408	36,766	98,4779
91	0,453	99,2589	33,360	99,0972
92	0,516	98,6214	45,375	99,3647
93	0,328	99,6526	39,531	98,7787
94	0,234	98,9251	34,484	98,2090
95	0,453	99,0682	64,984	99,1272
96	0,375	99,8315	52,016	99,6134
97	0,313	98,5090	54,672	98,3281
98	0,140	95,4650	24,454	99,3672
99	0,312	97,8811	56,125	96,5627
100	0,313	98,4729	44,906	95,5736

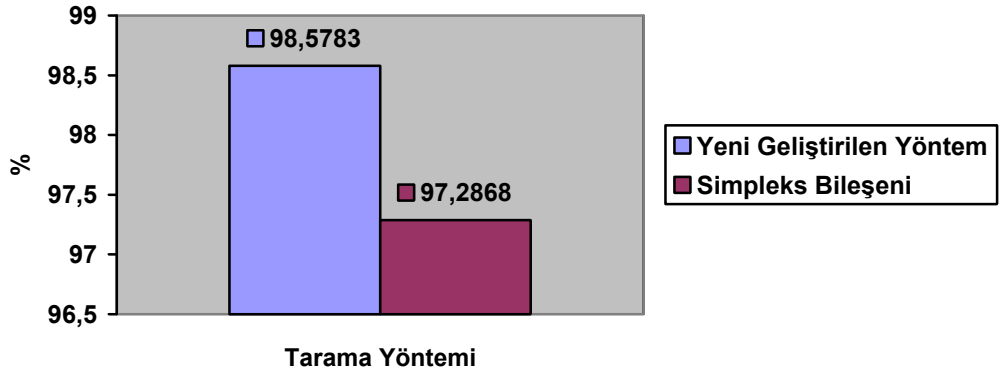
100 Adet problem çözümü sonucunda;

Yeni çözüm yaklaşımının toplam problem çözüm süresi : 35,858 sn
Yeni çözüm yaklaşımının ortalama problem çözüm süresi : 0,359 sn
Yeni çözüm yaklaşımının ortalama yerleşim yüzdesi : % 98,5783

Simpleks bileşeninin toplam problem çözüm süresi : 4351,430 sn
Simpleks bileşeninin ortalama problem çözüm süresi : 43,514 sn
Simpleks bileşeninin ortalama yerleşim yüzdesi : % 97,2868



Şekil 4.18: Monte Carlo simülasyonu tarama zaman grafiği



Şekil 4.19: Monte Carlo simülasyonu tarama yerleşim yüzdesi grafiği

Yeni geliştirilen yöntem ile aynı kaynak ve sipariş bilgilerine sahip girdilerle klasik yöntem kullanarak optimizasyon yapan SmartCutPro yazılımı ile zaman ve yerleşim oranı bakımından karşılaştırmalar yapılmıştır. Bu karşılaştırmalar sonucunda SmartCutPro yazılımının hızlı moduna göre yaklaşık 11,7 kat daha kısa sürede problem çözülmüş, %0,21 daha iyi yerleşim oranı elde edilmiştir. SmartCutPro yazılımının yavaş moduna göre yaklaşık 60,8 kat daha kısa sürede problem çözülmüş, %0,19 daha kötü yerleşim oranı elde edilmiştir.

Yeni geliştirilen yöntem ile aynı kaynak ve sipariş bilgilerine sahip girdilerle klasik yöntem kullanarak optimizasyon yapan simpleks bileşeni ile zaman ve yerleşim oranı bakımından Monte Carlo simülastyonu yapılarak karşılaştırmalar yapılmıştır. Bu karşılaştırmalar sonucunda Simpleks bileşenininden yaklaşık 121 kat daha kısa sürede problem çözülmüş, %1,29 daha iyi yerleşim oranı elde edilmiştir.

SONUÇLAR ve ÖNERİLER

Yapılan bu çalışmada ölçüleri kullanıcı tarafından belirlenebilen farklı adetlerde kaynaklardan, yine ölçüleri kullanıcı tarafından belirlenebilen farklı adetlerde sipariş stokların en az fire verecek şekilde yerleşimini hesaplayan bir uygulama geliştirilmiştir. Uygulamada klasik stok kesme yöntemleri yerine yeni bir yaklaşım geliştirilerek en kısa sürede en iyi yerleşimi yapmak amaçlanmıştır.

Sonuç olarak tamsayılı programlamaya alternatif olarak stok kesme problemi üzerine geliştirilen bu yeni yöntem, stok kesme problemleri özelinde tamsayılı programlamaya göre çok daha kısa sürede yaklaşık sonuçlar elde edebilmiştir. Çok yüksek çeşitlilikte kaynak ve sipariş ürünlerin olduğu problemlerde işlem yapılacak uzay çok büyüdüğünden tamsayılı programlama ile çözüm elde etmek zorlaşmaktadır, aynı şekilde benzer yapıda olan üç boyutlu yerleşim problemlerinin çözümünde yada dikdörtgen dışında düzgün yapıda olmayan nesnelerin yerleştirilmesi ve kesilmesi problemlerinde de problemin çözümü için gerekli olan zamanın kısalması sağlanabilir. Sonraki yapılacak çalışmalarda yeni yaklaşımın sadece kesme ve yerleşim problemleri özelinden çıkarılarak tamsayılı programlamanın uygulanabildiği farklı problemlerde de sonuç vermesi üzerine çalışmalar yapılabilir.

KAYNAKLAR

Aslan, D., "Yöneylem Araştırmaları, İşletme Fakültesi Araştırma Enstitüsü Ders Notları", ***Atatürk Üniversitesi İşletme Fakültesi***, Erzurum, No:41, (1978).

Anderson, D., R., Sweeney, D., C., Williams, T., A., "An Introduction to management Science – Quantitative Approaches to Decision Making", Sixth Edition, ***West Publishing Company***, New York, (1991).

Barnhart, C., Hane, C., A., Vance, P., H., "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems", ***Oper. Res. 48***, 318-326, (2000).

Belov, G., Scheithauer, G., "A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths", ***European Journal of Operational Research 141***, 274–294, (2002).

Belov, G., Scheithauer, G., "Setup and open stacks minimization in onedimensional stock cutting", Technical report, ***Dresden University***, (2003).

Bodily, S., E., "Quantitative Business Analysis: Text and Cases", ***Irwin/ McGraw-Hill***, USA, (1998).

Bronson, R., "Theory and Problems of Operations Research, Schaum's Outline Series", ***McGraw-Hill***, USA, (1982).

Carvalho J., M., V., "LP models for bin-packing and cutting stock problems", ***European Journal of Operational Research 141***, 253-273, (2002).

Clapham, C., "Coincise Dictionary of Mathematics", 2 nd Edition, ***Oxford University Press***, New York, (1996).

Cooke, R., "The History of Mathematics: A Brief Course", ***John Wiley & Sons Inc.***, Canada, (1997).

Çetin, E., "Stokastik Programlama Yöntemiyle Bir Portföy Optimizasyonu Modelinin Geliştirilmesi", Doktora Tezi, ***İ.Ü. Sosyal Bilimler Enstitüsü Sayısal Yöntemler Bilim Dalı***, İstanbul, (2004).

Degraeve, Z., Peeters, M., "Optimal integer solutions to industrial cutting stock problems: Part 2 benchmark results", ***INFORMS Journal on Computing 15***, 58-81, (2003).

Dosan, İ., "Yöneylem Araştırması, Teknikleri ve İşletme Uygulamaları", 2. Baskı, ***Bilim Teknik Yayınevi***, İstanbul, (1995).

Gass, S., I., "Great Moments in History", ***OR/MS Today V29***, 31, (2002).

Gılmore, P., C., Gomory, R., E., "A linear programming approach to the cuttingstock problem", ***Operations Research 9***, 849-859, (1961).

Gottfried, B., S., Weisman, J., "Introduction to Optimization Theory", ***Prentice Hall Inc.***, New Jersey, (1973).

Gürdoğan, N., "Üretim Planlamasında Doğrusal Programlama ve Demir Çelik Endüstrisinde Bir Uygulama", ***Ankara Üniversitesi S.B.Fakültesi Yayını***, No:473, Ankara, (1981).

Johnston, R., E., Sadinlija, E., "A new model for complete solutions to nedimensional stock problems", ***European Journal of Operational Research 153***, 176-183, (2004).

Junger, M., Thienel, S., "The Abacus system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization", ***Software: Practice and Experience 30 (11)***,1325-1352, (2000).

Kobu, B., "İşletme Matematiği", 6. Baskı, ***Avcıol Basım Yayın***, İstanbul, (1997).

Linderoth, J., T., Savelsbergh, P., "A computational study of strategies for mixed integer programming", ***Informs Journal on Computing 11***, 173-187, (1999).

Martello, S., Toth, P., "Knapsack Problems – Algorithms and Computer Implementations", ***John Wiley and Sons***, (1990).

Martin, A., "General mixed-integer programming: computational issues for branch-and-cut algorithms", ***Computat. Comb. Optimization, LNCS***, Vol. 2241, pp. 1–25, (2002).

Nitsche, A., Scheithauer, G., Terno, J., "Tighter relaxations for the cutting stock problem", ***European Journal of Operational Research 112***, 654-663, (1999).

Peeters, M., "One-dimensional cutting and packing: new problems and algorithms", Ph.D. thesis, ***Katholieke Universiteit Leuven***, Belgium, (2002).

Özgüven, C., "Doğrusal Programlama ve Uzantıları", ***Detay Yayıncılık***, Ankara, (2003).

Öztürk, A., "Leontief Modeli ve Doğrusal Programlama", Genişletilmiş 2. Baskı, ***Örnek Kitabevi***, Bursa, (1986).

Öztürk, A., "Yöneylem Araştırması", 7. Baskı, ***Ekin Kitabevi Yayınları***, Bursa, (2001).

Scheithauer, G., Terno, J., Muller, A., Belov, G., "Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm", ***Journal of the Operational Research Society 52***, 1390-1401, (2001).

Scheithauer, G., Terno, J., "The modified integer round-up property of the onedimensional cutting stock problem", ***European Journal of Operational Research***, 562-571, (1995).

Schoenfield, J., E., "Fast, exact solution of open bin packing problems without linear programming", draft, ***US Army Space & Missile Defense Command***, Huntsville, Alabama, USA, 30, (2002).

Taha, H., A., Çeviren ve Uyarlayanlar: Alp BARAY- Şakir ESNAF, "Yöneylem Araştırması", 6. Basımdan Çeviri, ***Literatür Yayıncılık***, İstanbul, (2000).

Tulunay, Y., "İşletme Matematiği", ***İ.Ü. İşletme Fakültesi Yayın No: 233***, ***İşletme İktisadı Enstitüsü Yayın No: 126***, İstanbul, (1990).

Tütek, H., Gümüşoğlu, Ş., "Sayısal Yöntemler", 3. Baskı, ***Beta Basım Yayın Dağıtım***, İstanbul, (2000).

Ullmann, J., E., "Schaum's Outline of Theory and Problems of Quantitative Methods in Management", ***McGraw-Hill***, New York, (1976).

Ulucan, A., "Yöneylem Araştırması (İşletmecilik Uygulamalı Bilgisayar Destekli Modelleme)", ***Siyasal Kitabevi***, Ankara, (2004).

Ünsal, F., M., Rüzgar, B., Rüzgar, N., "Sayısal Yöntemler", Cilt 1, ***Türkmen Kitabevi***, İstanbul, (2000).

Vance, P., H., Barnhart, C., Johnson, E., L., Nemhauser, G., "Solving binary cutting stock problems by column generation and branch-and-bound", ***Comput. Opt. Appl. 3***, 111-130, (1994).

Vanderbeck, F., "Computational study of a column generation algorithm for bin packing and cutting stock problems", ***Mathematical Programming***, 565-594, (1999).

Vanderbeck, F., "Exact algorithm for minimizing the number of setups in the nedimensional cutting stock problem", ***Operations Research 48 (6)***, 915-926, (2000).

Wolsey, L., A., "Integer Programming and Wiley Interscience Series in Discrete Mathematics and Optimization", ***John Wiley and Sons***, Chichester, (1998).

ÖZGEÇMİŞ

1983 yılında Kayseri'de doğdu. Lise öğrenimini Kabataş Erkek Lisesi'nde tamamladı. 2001 yılında girdiği Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nden 2005 yılında mezun oldu. 2005 yılından itibaren İstanbul'da bulunan Okyanus Yazılım Bilgisayar Eğitim ve Danışmanlık firmasında Bilgisayar Mühendisi olarak görev yapmaktadır.