YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OPTİMİZASYON TEKNİKLERİ YÖNTEMİ İLE OLUKLU ÜRETİM HATTI PROBLEMİ ÇÖZÜMÜ

Matematik Müh. M. Zahid GÜRBÜZ

FBE Matematik Müh. Anabilim Dalında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı: Yrd. Doç. Dr. İbrahim EMİROĞLU

İSTANBUL, 2007

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

a_i	Safihanın eni
x_i	Safihanın bir kombinasyonda kaç sütunda yer alacağı
b_i	Safihanın boyu
y_i	Safihanın kombinasyon sonrası üretilmesi planlanan miktar
m_i	Safihanın kombinasyon öncesi istenen üretim miktarı
n	Üretilecek safiha sayısı
K	Bir kombinasyondaki maksimum Sütün sayısı
K_i	Safiha
L	Rulo halindeki oluklu mukavvanın boyu
R	Rulo halindeki oluklu mukavvanın eni
S	Kombine üretimde izin verilen maksimum fire
s	Kombine üretimde oluşan fire

KISALTMA LİSTESİ

IEEE	Institute of Electrical and Electronics Engineers
LP	Lineer Programlama

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Optimizasyon, günümüzde en çok ihtiyaç duyulan konulardan biridir. Çünkü optimizasyon, ekonominin temel ilkesi olan "kıt kaynaklarla en fazla faydayı sağlamak" ilkesinin en önemli aracıdır. Oluklu mukavva sektörünün üretim safhalarındaki en büyük sorunu olan "kombine üretim" aşaması, gerek modellemesiyle gerekse uygulaması ile tam bir mühendislik problemidir. Böyle bir çalışmanın gerçek hayatta çok faydalı bir yere sahip olduğunu görmek gerek çalışma süresinde gerekse sonuç aşamasında oldukça faydalı ve zevkli bir süreç geçirmeme vesile olmuştur.

Bu zevkli çalışmamda, analiz, modelleme ve uygulama kısmında benden desteğini esirgemeyen, sektörden Gn. Md. Güray MOLLAOĞLU, Gn. Md. Halil AYDIN ve Analist Hakan CIBIR ile Yük. Müh. Birol ASLANYÜREK ve Mat. Müh. Ayfer AKEMİR'e ve en önemlisi tez danışmanım saygıdeğer hocam Yrd. Doç. Dr. İbrahim EMİROĞLU'na teşekkürlerimi sunarım.

Saygılarımla,

M.Zahid GÜRBÜZ

ÖZET

Bu tezin amacı, oluklu üretim süreçlerinden en önemlisi olan kombine üretim safhasında maliyet, fire ve bekleme süresi konularında optimum derecede üretimi gerçekleştirecek matematiksel modelin kurulması ve gerçek hayatta uygulanmasıdır. Literatürde, benzer üretim aşamalarının optimizasyonu konusunda birçok çalışma yapılmış olmasına karşın oluklu üretimi konusunda özel bir çalışmaya rastlanmamıştır. Kombine üretim, şimdiye kadar usta diye tabir edilen kişilerce göz kararı ile seçilen ürünler bir arada üretilerek "kombine üretim" işlemi tamamlanıyordu.

Tez çerçevesinde ilk önce oluklu üretiminin ne olduğundan ve hangi aşamalardan geçtiğinden kısaca bahsedilecektir. Ardından asıl konumuz olan kombine üretim aşaması üzerinde durulacaktır. Bunun yanı sıra benzer konulardaki çalışmalardan, örneğin, kumaş kesme veya cam kesme işlemlerinden bahsedilecek ve Oluklu üretimin bunlardan faklı olan tarafları ortaya koyulacaktır. Kombine üretiminde optimizasyon yapılacak kısımlar yani problemin amacı ve koşulları ortaya konarak matematiksel modeli kurulacaktır. Bu Tez içerisinde, kombine üretiminin diğer bir adıyla oluklu kesme probleminin, hem lineer programlama problemine indirgenebileceği hem de uygun bir algoritma ile çözülebileceği öne sürülmüştür. Ardından da bu çözümün gerçekten de optimumluğu sağladığı gösterilecektir.

Tezin son bölümünde ise sonuçların sayısal verileri yer almaktadır. Elde edilen sonuçlardan yola çıkarak yapılan optimizasyonun çok ciddi bir şekilde faydası ortaya çıkmıştır. Üretim maliyetlerinin düşmesine hatırı sayılır bir şekilde katkılar sağladığı görülmüştür. Lineer programlamanın kullanılması ile daha önce kullanılan deneme yanılma yöntemine göre sadece sonuca yaklaşan ihtimaller denendiği için daha hızlı sonuç elde edilmiştir. Aynı şekilde önerilen algoritma da mümkün olmayacak ihtimalleri atladığı için çok hızlı sonuç vermiştir.

Anahtar Kelimeler: Oluklu Mukavva, Kombinasyonel Optimizasyon, Kombine Üretim, Oluklu Üretim, Kesme Optimizasyonu.

ABSTRACT

The aim of this thesis is to design the mathematical modeling to implement production in optimum level about cost, outage and waiting period in the phase of combinational manufacturing which is the most important process of the corrugated cardboard production processes and to carry out it. Although there is a huge number of works for solving similar problems, there is no special work about corrugated production optimization. The combinational manufacturing process has been implementing by the craftsmen by gathering products which are chosen roughly for now on.

In this study, first of all, what the corrugated production is and processes of it are introduced basically. Then the combinational manufacturing which is the main purpose of this study is focused. In addition to this, similar works like cutting stage problem or glass cutting problem are mentioned and the differences of corrugated production from the others are pointed. After that, according to the main aim of this study, the optimization conditions are introduced and the mathematical model of the problem is introduced. In this study, the problem of combinational manufacturing in the corrugated cardboard production is suggested that it can be solved by both linear programming and a suitable algorithm. Then, it is seen that the suggested solution really provides its optimum.

In final section of this work, the numerical results of the solutions are presented. According to this data, the suggested optimization is turned out to be an advantage. This study helps on considerably the decrease in production costs. With the usage of linear programming, the solution is gotten rapidly. In the same way, because of the fact that the suggested algorithm passes the probabilities which are impossible, it also gets a rapid solution.

Keywords: Corrugated Sheet, Combinational Optimization, Combinatorial Production, Corrugated production, Cutting optimization.

1. GİRİŞ

Üretim hattında optimizasyon, üretici firmaların en büyük sorunlarından biridir. Piyasada tutunma çabalarında fiyatları arttıramadıkları için maliyeti düşürme yollarına gidecekler ve bunun için üretim aşamalarında oluşan fireleri minimize etme yoluna gideceklerdir.

1.1 Konu ve Önemi

Oluklu üretimi, basitçe anlatmak gerekirse, büyük bir parça kâğıttan küçük parçalar üretme işidir. Bu kağıtlar şekil olarak dikdörtgen biçimindedir. Oluklu Üretimi optimizasyonu tabiriyle de bu küçük parçaları üretirken en az fireyi vermek dolayısıyla oluşacak zararları minimize etme işidir.

Bu konu, literatürde 2 boyutlu kombinasyonel optimizasyon problemleri, 2 boyutlu kombinasyonel üretim problemleri yada 2 boyutlu kesme optimizasyonu problemleri arasında yer almaktadır. Ancak, oluklu üretiminde kombine üretim safhası diğer 2 boyutlu optimizasyon problemlerinden farkları vardır. Bu farklar gerek siparişlerin farklılığı gerekse üretimi yapan makinenin çalışma şekline bağlı olarak ortaya çıkmaktadır. Oluklu üretimi, üretim şekli olarak, tekstilde kumaş kesme problemi (Trubin ve Grad, 1977), buna benzer olarak Cam Kesme optimizasyonu (Puchinger, Raidl ve Koller, 2004), matematiksel bir oyun olarak karşımıza çıkan sekiz vezir problemi* (Watkins, 2004) gibi problemlerle benzerlik gösterir. Diğer bir bakış açısıyla 2 boyutlu yerleştirme problemleri de aslında benzer bir problemdir.

Trubin ve Grad kumaş kesme probleminde istenen boyuttaki küçük kumaşları (üretilmek istenen kumaşları) üretirken bütün siparişin üretilmesi gerektiği varsayımıyla yola çıkmıştır. İlk önce 1 kumaşı üretmiş. Hemen sonra uygun olan 2 kumaşı üretirken bir önceki üretimden kaldığı yerden devam etmiştir. Bu şekilde bütün kumaşları üreterek sonuca ulaşmıştır. Kombine üretimde kâğıdı kesen makinenin bıçak yerleri üretim işi tamamlanmadan değiştirilemiyor. Yani makine bir kez ayarlanıyor ve işlem tamamlanana kadar o şekilde kalıyor. Aynı zamanda kombine üretim optimizasyonunda bir üretim hamlesiyle bütün parçaların üretilmesi şartı yoktur. Yani uygun kombinasyon sağlanamıyorsa üretim gerçekleşmeyebilir.

---

* Çözüm yolu olarak benzerlik gösteriyor. Farkı ise elde edilecek sonucun belirli olması yani çözüm bulunduktan sonra daha iyisini arama yoluna gidilmiyor.

Puchinger, Raidl ve Koller'in Cam kesme optimizasyonu, camları hem dikey hem de yatay olarak parçalara ayırdığı için kombine üretime daha çok benzerlik göstermektedir. Bu yöntemin dezavantajı ise çözüm yolundadır. Puchinger, Raidl ve Koller, sezgisel(heuristic) algoritmalar kullandıkları için tam olarak optimumluktan söz edilemez.

Yukarıda bahsedilen her iki yöntemde de, kombine üretimde bulunan, dikey olarak kesilebilecek ürün sayısı hakkında hiçbir kısıt yoktur. Kombine üretimde, üretimi yapan makinenin büyüklüğüne göre 4,5 veya 6 adet dikey kesim bıçağı bulunmaktadır. Bazı çok gelişmiş makinelerde de bu sayı 7 ye çıkmaktadır.

1.2 Tezin Önemi ve Kapsamı

Bu tezin amacı, kombinasyonel bir problem olan oluklu mukavva üretimindeki kombine üretim safhasında minimum fire ve maksimum uzunluk ile üretilebilecek kombinasyonu bulabilmektir. Tez çerçevesinde kullanılacak olan büyük kâğıt (üretimde kullanılacak kağıt)'ın eni sınırlı ve belirli kabul edilecek ve boyu sınırsız olarak kabul edilecektir. Gerçek hayatta sınırsız bir boy olamaz. Bunu böyle kabul etmemizdeki sebep; üretimde kullanılacak büyük kâğıt bittiğinde hemen ardına yenisi takılarak üretim aksamadan devam edebiliyor olmasıdır.

Tez çerçevesinde iki tür çözüm yolu anlatılacaktır. Birincisi kombinasyonların hepsini deneyerek en uygun olanı seçmek, diğeri ise tez kapsamında ispatlamaya çalışacağım lineer programlama modelini kurup bu yöntemle çözmek olacaktır. İki yol da optimumluğu sağlıyor. Çünkü her iki yöntem de kesin bir optimumluk değerini gösteren yöntemlerdir.

Tez içerisinde ilk önce problem, detaylı bir şekilde anlatılmış, matematiksel modeli kurulmuş ardından da çözüm yollarından bahsedilmiştir. Bir sonraki bölümde öncelikle lineer kombinezonları deneme yolu ve algoritması anlatıldı ve çözüm yolu açıklandı. Sonraki bölümde de lineer programlama modeli kurulup çözüm MS Excel programıyla elde edilmiştir.

2. PROBLEMİN TANIMI

Bu bölümde kombine üretimin nasıl gerçekleştiği, üretimi yapan makinenin özellikleri ve ne gibi kısıtlara sahip olduğu, üretim için gelen müşteri isteklerinin dağılımı hakkında detaylı bilgiler ve gelen kutu siparişlerin göre kesilecek safiha*ların boyutları hakkında bilgi yer almaktadır.

2.1 Kombine Üretim

Oluklu Mukavva (ambalaj) sektöründe kombine üretim, üretimin en önemli ve üzerinde durulması gereken kısımdır. Çünkü üretimin temel maddesi olan mukavva bu bölümde parçalara ayrılır ve fireler de en çok bu bölümde ortaya çıkar. Bu nedenle bu aşamadaki fireleri minimize etmek firma açısından çok büyük avantaj sağlayacaktır.

Kombine üretim aşamasında kullanılan oluklu makineleri yaklaşık 100 metre uzunlukta bir alana yerleştirilmiştir. Makinenin eni, cinsine göre değişmekle birlikte ortalama 1.5 ila 2.5 metre arasında değişmekle birlikte gelişmiş ülkelerde hemen hemen bütün oluklu makinelerinin genişliği 2 metreden fazladır. Yüksekliği de yine ortalama 3 metredir. Makinenin eni bu problemin temel kısıtını oluşturmaktadır.

Oluklu makinesi, bir, iki veya üç kütükten üç, beş veya yedi tabaka kağıdı üretip bir araya getirebilen bir dizi makinedir. Çalışması kesintisiz bir prosestir. Oluklu makinesine istenen kağıt bobinleri yerleştirilir. Sonra makinenin ilk evresinde bu kâğıtlar oluşturulmak istenen safihanın cinsine göre oluk oluşturulur ve üst üste yapıştırılarak oluklu mukavva elde edilir. Bu yapıştırma işlemi yaklaşık 250 C derece sıcaklıklarda yapılır. Sonra hızla soğutularak mukavva şeklini alır. Bu oluşan mukavva bütün halindedir. Bir sonraki aşamada bu bütün olan mukavvayı siparişe uygun parçalara ayırmak gerekir. İşte asıl firenin çıktığı kısım burasıdır. Çünkü burada mukavvanın eninden verilen %3'lik bir fire üretilen mukavvanın boyunun çok uzun olduğu gerçeği karşısında büyük bir kayıp ve maliyete ağır bir yük getirmesi anlamına gelir. Bu nedenle bu firenin %1 civarlarında olmasını sağlamak için optimizasyon yapmamız isteniyor.

---

* Safiha, olukları oluşturulmuş mukavvanın kesilmiş, küçük parçalara ayrılmış haline denir. Dikdörtgen şeklindeki düz yassı levhaya verilen addır.

2.2 Siparişlerin dağılımı

Oluklu sektöründe üretilmesi için gelen siparişler çok farklılık göstermektedir. Aşağıdaki Tablo 2.1 de bir firmanın 1 günlük gelen siparişleri gösterilmiştir. Bu siparişlere bakılarak gelen siparişlerde üretilmesi gereken safihaların en, boy ve miktar değerleri oldukta değişkendir. Her müşteri kendi siparişlerinde bile farklı boyutlarda safihalar talep etmektedir. Bu durumda gelen siparişlere göre belirli bir genelleme yapılıp üretimi o şekilde yapmak imkansız hale gelmektedir.

Tablo 2-1 Bir oluklu firmasına ait bir günlük safiha siparişlerinin listesi.

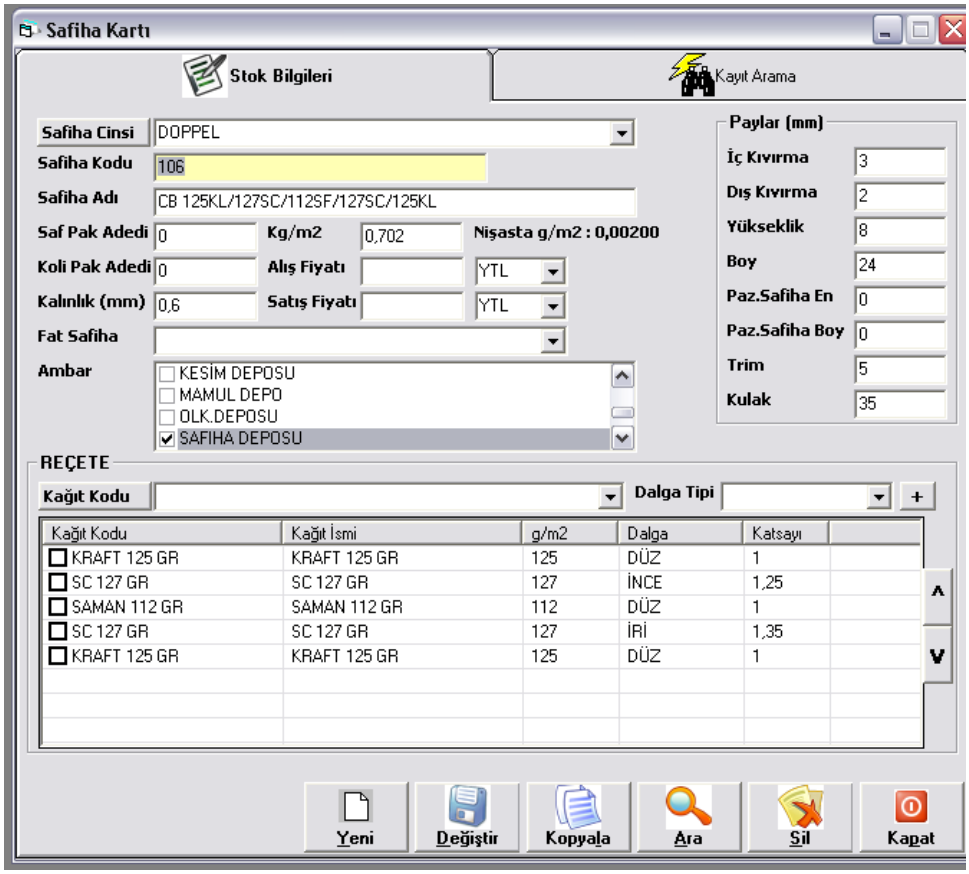
MusteriKodu	SiparisID	SafihaKodu	sEn	sBoy	Miktar	Termin*
YEŞİM	396	418	584	1940	1900	May 15 200
ROTAP	413	420	719	1100	600	May 4 200
MAJÖR	426	420	808	2124	1000	May 8 200
ROTAP	409	432	790	1990	500	May 4 200
ROTAP	410	432	690	1990	500	May 4 200
SAİDE	401	497	704	2035	603	May 3 200
SAİDE	399	498	704	1835	1055	May 4 200
YEŞİM	397	513	602	1415	175	May 5 200
ROTAP	411	513	680	1195	250	May 4 200
ROTAP	414	513	520	1390	250	May 4 200
ROTAP	415	513	570	1390	200	May 4 200
ROTAP	416	513	620	1390	400	May 4 200
ROTAP	417	513	640	1190	1000	May 4 200
ROTAP	412	514	754	1325	2000	May 3 200
SAİDE	400	514	250	1000	1055	May 4 200
HAYAT	402	535	491	1162	5000	May 3 200
HAYAT	403	535	469	1057	5000	May 3 200
HAYAT	404	535	469	1057	5000	May 4 200
HAYAT	405	535	426	1322	2500	May 4 200
HAYAT	406	535	564	1627	2500	May 8 200
HAYAT	407	535	590	1267	1500	May 4 200
HAYAT	408	535	530	1477	4000	May 4 200
BOLPA	419	561	481	1307	4200	May 8 200
BOLPA	420	561	481	1307	3600	May 8 200
BOLPA	421	561	491	1307	4200	May 8 200
BOLPA	422	561	451	1307	4800	May 8 200
BOLPA	423	561	471	1307	2400	May 8 200
BOLPA	424	561	481	1307	2400	May 8 200
BOLPA	425	561	481	1307	1800	May 8 200
BOLPA	418	590	611	1307	2400	May 8 200
YEŞİM	398	601	375	811	30000	May 4 200
YEŞİM	395	651	365	811	8500	May 3 200

* Siparişin en son ne zamana kadar üretileceği tarihtir. (Deadline)

2.3 Safiha Kavramı

Safiha, kelime anlamı olarak düz, yassı yüz ve madeni levha anlamlarına gelmektedir. Oluklu sektöründe de safiha, üretilecek küçük mukavva parçalarına verilen addır. Safihalar birden fazla kâğıdın düz veya oluk biçiminde 250 C nin üzerinde bir sıcaklıkta üst üste yapıştırılması ile elde edilmiş levhalardır. Şekil 2.1 de görüldüğü gibi 3 düz kağıt arasına 2 adet farklı oluk dereceleriyle (ince ve iri olarak) yerleştirilmiş iki kağıdın oluşturduğu bir safiha görülmektedir. Şekil 2.2 de ise Şekil 2.1. de gösterilen safihanın stok kaydı girişini göstermektedir.

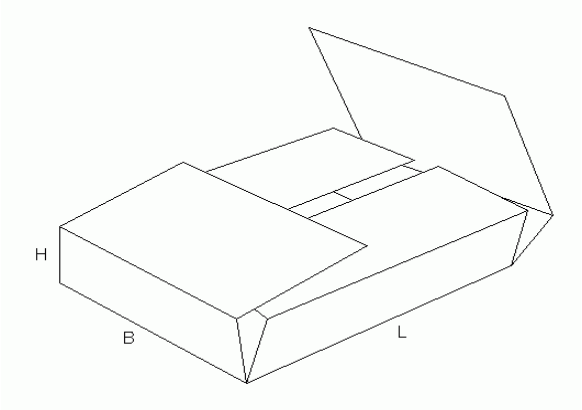
Şekil 2-1 Oluk cinsi doppel olan bir safiha kesiti.



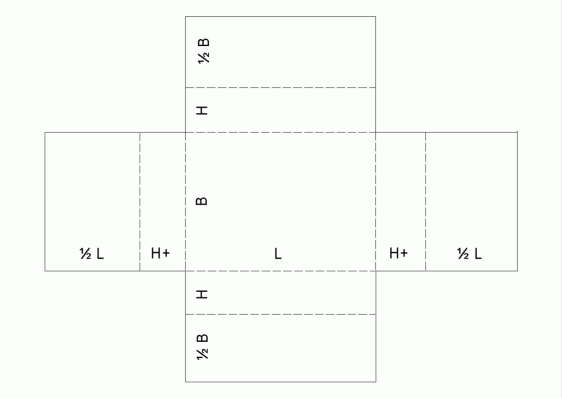
Şekil 2-2 Safiha'yı oluşturan bileşenler.

Son olarak da aşağıdaki Şekil 2.3 ve şekil 2.4 de bir kutunun açık ve kapalı halleri görülmektedir. Müşteriler kapalı kutunun boyutlarını sipariş olarak giriyor ve üretici firma o

kutu için gerekli olan şekil 2.4 deki safihanın enini ve boyunu hesaplayarak imalata yolluyor. Ancak şunu belirtmeliyim ki şekil 2.4 de görülen safiha dikdörtgen değildir. İmalat aşamasında bunun dış yüzeylerini çevreleyecek şekilde dikdörtgen olarak kesilmesi kombine üretim aşamasında yapılmaktadır. Son haline ulaşması ise sloter denen başka makinelerle yapılmaktadır. Ancak bu konuya girilmeyecektir.



Şekil 2-3 Kutunun kapalı hali.



Şekil 2-4 Kutunun açık hali.

3. PROBLEMİN FORMÜLASYONU

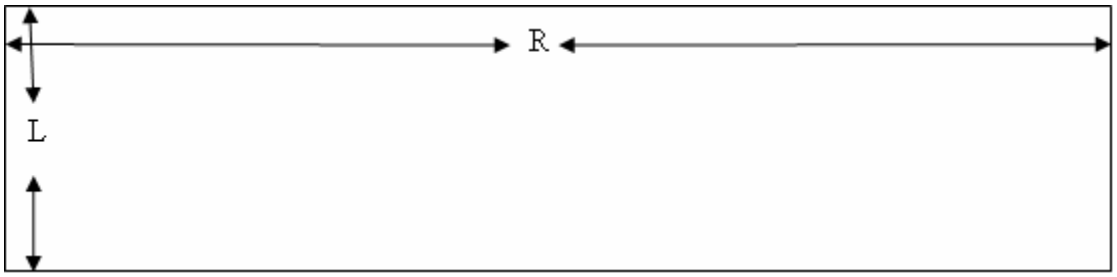
Kombine üretim problemi, minimize edilmesi gereken bir amacı olan ve gerek malzemeden kaynaklanan gerek makinenin fiziksel yapısından kaynaklanan gerekse yönetim kararlarından kaynaklanan belirli kısıtlamalara sahip bir problemdir.

3.1 Kısıtlamalar

Kısıtlamaları, kullanılan mukavvaya, makine yapısına ve yönetim kararlarına bağlı olmak üzere olarak 3 bölümde incelenebilir.

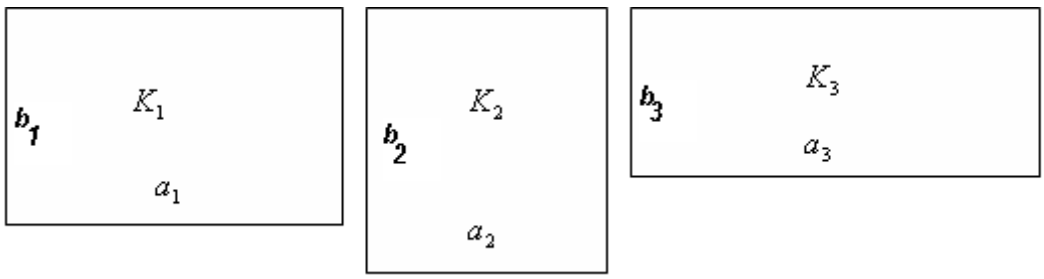
3.1.1 Kullanılan Mukavvaya Bağlı Kısıtlamalar

Rulo eni R olan bir mukavva rulosundan n adet safiha üretilmek isteniyor. Bu durumda üretilmek istenen safihaların enleri ile kaç sütunda kullanıldıklarının çarpımının toplamı R rulo enini geçemeyecektir.



Şekil 3-1 Mukavva rulosunun açık hali.

Bir mukavva rulosu şekil 3.1 de gösterilmiştir. Bu rulodan safihalar kesmek istiyoruz. Keseceğimiz her safihanın bir eni ve boyu vardır. Bunları a_i (i=1,2,…n) olarak ifade edeceğiz. Aşağıdaki şekil 3.2 de görüldüğü gibi safihanın eni a ile boyu b ile gösterilmiştir.

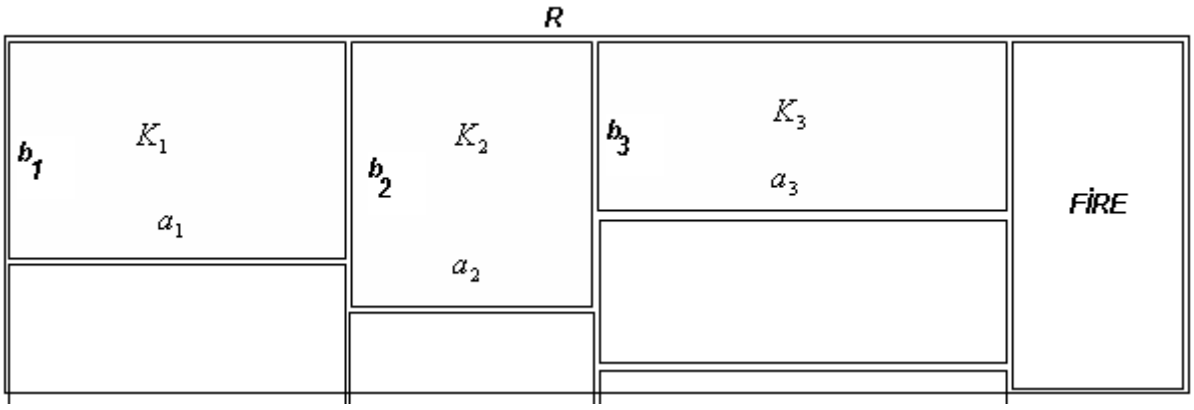


Şekil 3-2 Safihaların gösterimi..

Bu bahsettiklerimizden yola çıkarak safihaların enlerinin toplamının R rulo eninden küçük olması gerektiği koşulu aşağıdaki gibi yazılabilir.

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + ... + a_n x_n \leq R \quad (3.1)$$

Burada a_i (i=1,2,…n) değerleri safihaların enini x_i değerleri de i. safihadan kaç kez kullanılacağını belirtir.



Şekil 3-3 Safihaların rulo üzerindeki dizilişi.

Şekil 3.3 de görüldüğü üzere K1, K2 ve K3 safihaları 1 er kez kullanılmış ve en sağda fire olarak bir kısım kullanılamamıştır. Bu şekle göre x1=1, x2=1 ve x3=1'dir. Bu kombinasyon ile üretim yapıldığında bir miktar fire verilmiştir. Diğer kombinasyonlarda, örneğin, x1=1;x2=3 ve x3=0 olduğu kombinasyonda belki de daha az fire ile üretim gerçekleşecekti. Firenin minimum olması için biz 2. ihtimal olan daha az fireli kombinasyon ile üretimi gerçekleştirmeliyiz.

3.1.2 Yönetim Kararlarına Bağlı Kısıtlamalar

Üretici firma yönetimi, S milimetre fireden daha fazla fire veren ürünleri üretmek istememektedir. Bunun nedeni S'den daha fazla miktarda fire ile ürün ürettiğinde kaybedecekleri atık mukavva miktarı çok büyüyecektir. Bu S miktarı firmanın ürettiği minimum ene sahip safihaların ortalama değeri olarak alınabilir. Örnek vermek gerekirse, eğer firma 100 mm lik siparişleri vasat bir frekansla üretiyorsa bir kombinasyonda 100 mm den fazla fire vermek, kullanabileceği halde kullanmamak yani israf etmek anlamına gelecektir. Bu diğer kullanılamaz olan firelerin aksine firmanın üst yönetimi tarafından kolaylıkla kabullenilebilecek bir durum değildir. Bu nedenle ikinci kısıt olarak;

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n \geq R - S \qquad (3.2)$$

kısıtını ekleriz. Bu sayede gereksiz fireyi de ortadan kaldırmış oluruz.

3.1.3 Makine Yapısına Bağlı Kısıtlamalar

Yukarıda anlatılan kısıtlamalara ek olarak bir de makinenin sahip olduğu bıçak sayısından kaynaklan bir kısıt daha vardır. Genel olarak oluklu makineleri 5 adet bıçağa sahip olurlar ve aynı anda paralel olarak en fazla 6 ürün üretebilirler. Yani en fazla $\sum_{i=1}^{n} x_i = 6$ olabilir. Bunu da bir kısıt olarak eklememiz gerekiyor.

$$\sum_{i=1}^{n} x_i \leq K \qquad (3.3)$$

3.1.4 Kullanılan Mukavvanın Boyu

Bu kadar açıklamalardan sonra dikkat çeken bir konu da kullanılan mukavvanın boyu (L) ile ilgili hiçbir kısıtlamanın olmamasıdır. Burada hemen akla üretilecek safihaların boyu ile miktarlarının çarpımının büyük kâğıdın boyu olan L den küçük olması gerektiğidir.

$$b_i . y_i \leq L \qquad (i=1,2,\ldots n) \qquad (3.4)$$

Ancak, üretimde kullanılan safihaların boyu ve üretim adedi çarpımı mukavva boyunu geçtiği durumda hemen ardına yeni mukavva ekleyerek üretimin durmadan devam ediyor olması bize bu kısıtlamanın ortadan kalktığını gösteriyor. Yani burada $L \to \infty$ olarak ele alınabilir. Bu durumda;

$L \to \infty$ iken $b_i . y_i \leq \infty$

olur. Bu koşul her b_i ve y_i değeri için sağlanacağından dolayı, bu bir kısıtlama olarak görülmüyor ve problemin matematik modeline dâhil edilmiyor.

3.1.5 Toplam üretimin Boyu

Toplam üretimin boyu ile anlatılmak istenen bir kombinasyon tamamlandığında ne kadar uzunlukta mukavva kullanıldığıdır. Bu optimizasyondaki amaç minimum fire olarak ele alındığına göre akla hemen boyda verilecek fire gelmektedir. Esasında burada boydan fire

veriliyor olmasına karşın bunu ihmal edeceğiz.

Boydaki fire bir ürün tamamlandığında diğer ürünler tamamlanmaz ise tamamlanan üründen fazla miktarda üretim yapılacağı için o fire olarak kabul edilebilir. Ancak oluklu optimizasyonunda bir ürün tamamlandığı anda o kombinasyon için üretim bitmiş sayıldığı için orada kalan fire tamamlanmayan safihaların kendin boylarının 1 katının uzunluğunu geçemez. Metrelerce uzunluktaki bir kombine üretimde fire olarak ortaya çıkan bu bir boy safiha önemsenmeyecek kadar azdır. Bu nedenle rahatlıkla boy firesini ihmal edebiliriz.

4. PROBLEMİN LİNEER PROGRAMLAMA PROBLEMİNE İNDİRGENMESİ

Problemin matematiksel modelinden yola çıkarak, bunu bir Lineer programlama Problemine indirgeyebiliriz. Burada 3 adet kısıt ve bir amaç fonksiyonu görüyoruz.

4.1 LP Modelinin Kurulması

4.1.1 Kısıt Kümesi

Kısıtlar, yukarıda anlatılan modellemede geçen (3.1), (3.2) ve (3.3) denklemleri LP modeli için kullanılabilir. Bunlar, bir araya getirerek aşağıdaki kısıtlar kümesini oluşturur.

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n \leq R \tag{3.1}$$

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n \geq R - S \tag{3.2}$$

$$\sum_{i=1}^{n} x_i \leq K \tag{3.3}$$

Bu denklemleri 0 dan küçük veya eşit hale dönüştürerek LP problemini daha basit hale getirebiliriz. O halde problem;

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n - R \leq 0 \tag{4.1}$$

$$R - S - (a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n) \leq 0 \tag{4.2}$$

$$\sum_{i=1}^{n} x_i \leq K \tag{4.3}$$

şeklini alır. Burada, a_i (i=1,2…n), üretilecek safihanın eni; n, üretilecek safiha sayisi; R, Büyük mukavvanın eni; S, beklenen maksimum fire; K, Kombinasyondaki maksimum eleman sayısı olup hepsi bilinen birer sabitlerdir. x_i (i=1,2…n), safihanın kombinasyonda kaç kez kullanıldığını göstermekte olup bulunması gereken değişkendir.

4.1.2 Pozitiflik Koşulu

LP modelinde bulunması gereken değişkenler $x_1, x_2, \ldots x_n$ değişkenleridir. Bunlar, Bir safihanın söz konusu kombinasyonda kaç kez kullanıldığını gösterirler. Dolayısı ile $x_1, x_2, \ldots x_n$ değişkenleri sıfırdan büyük veya eşit olması gerekmektedir. O halde;

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0 \tag{4.4}$$

diyebiliriz. Böylece LP probleminin pozitiflik koşulları da sağlanmış oldu.

4.1.3 Amaç Fonksiyonu

LP modelinin bir diğer temel kavramı da amaç fonksiyonudur. Kombine üretim probleminde amaç oluşacak fireyi minimize etmektir. Dolayısıyla oluşturulacak kombinasyondaki fireyi minimum yapan değer optimum temel çözüm olarak kabul edilecektir. Bu durumda;

$$\min\ Z = R - (a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n) \tag{4.5}$$

Olarak amaç fonksiyonunu elde ederiz.

4.1.4 Tamsayı Koşulu

Bahsedildiği üzere $x_1, x_2, \ldots x_n$ değişkenleri safihanın söz konusu kombinasyonda kaç kez kullanıldığını gösterirlerdi. Buradan anlaşılıyor ki, kullanım sayılar pozitif olması gerektiği gibi tamsayı olmaz zorundadır. Çünkü, bir safiha kombinasyonda yarım kez kullanılır diye bir şey söz konusu değildir. Ya kulanılmaz ya da 1 veya daha fazla kez kullanılır. O halde $x_i \in Z^+$ (i=1,2…n) diyebiliriz.

4.2 LP Probleminin MS Excel ile çözümü

Doğrusal programlama problemlerinin çözümünde Excel çözücüsü, hem Excel'in çok yaygın olarak kullanılması hem de çözümün kolay ve anlaşılır olması nedeniyle kullanıcılar için pek çok avantaj sağlar (Alan ve Yeşilyurt, 2004). Şimdi LP modelinin Excel ile çözümünün nasıl olduğunu görelim. Tekniği daha iyi kavramak için bir örnek üzerinden gidelim.

4.2.1 Örnek Problem

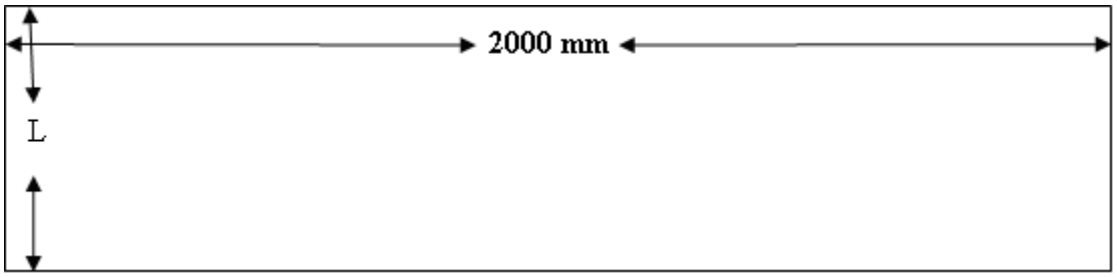
Tablo 4-1 LP Probleminin Excel ile çözümü için örnek sipariş kayıtları.

MusteriKodu	SiparisID	SafihaKodu	sEn	sBoy	Miktar	Termin
YEŞİM	100	420	111	1140	250	May 15 200
ROTAP	101	420	360	680	1000	May 4 200
MAJÖR	102	420	900	1700	200	May 8 200

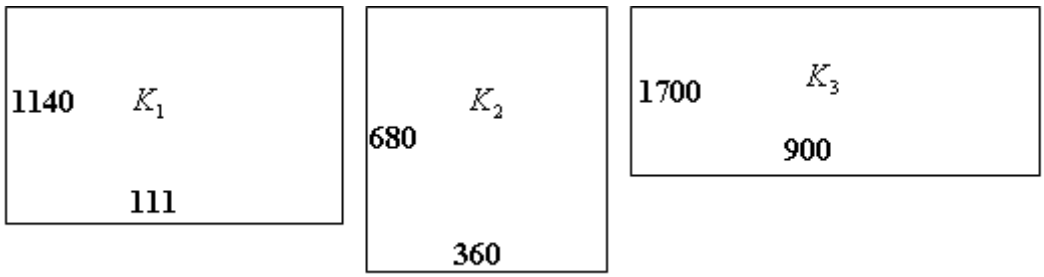
Yukarıdaki tabloda 3 adet safiha siparişi görülmektedir. Üretici firmanın sahip olduğu makine en fazla 6 adet ürünü aynı kombinasyonda üretme kapasitesine sahiptir. Üretimde

kullanacağımız mukavvanın eni 2000 mm olduğuna göre ve yönetim tarafından istenen maksimum fire miktarı 50 mm olduğuna göre; bu 3 üründen üretilebilecek en uygun kombinasyonu bulunuz.

ÇÖZÜM:



Şekil 4-1 Örnek rulo.



Şekil 4-2 Örnek safihalar.

3 adet sipariş kaydı olduğu için 3 adet bilinmeyene ihtiyaç vardır. Bunlar, x_1, x_2 ve x_3 olsun.

Problemin LP Modeli şu şekildedir:

1.Amaç:

$$\min Z = 2000 - (111 .x_1 + 360 .x_2 + 900 .x_3)$$

2.Kısıtlar:

$$111 .x_1 + 360 .x_2 + 900 .x_3 - 2000 \leq 0$$

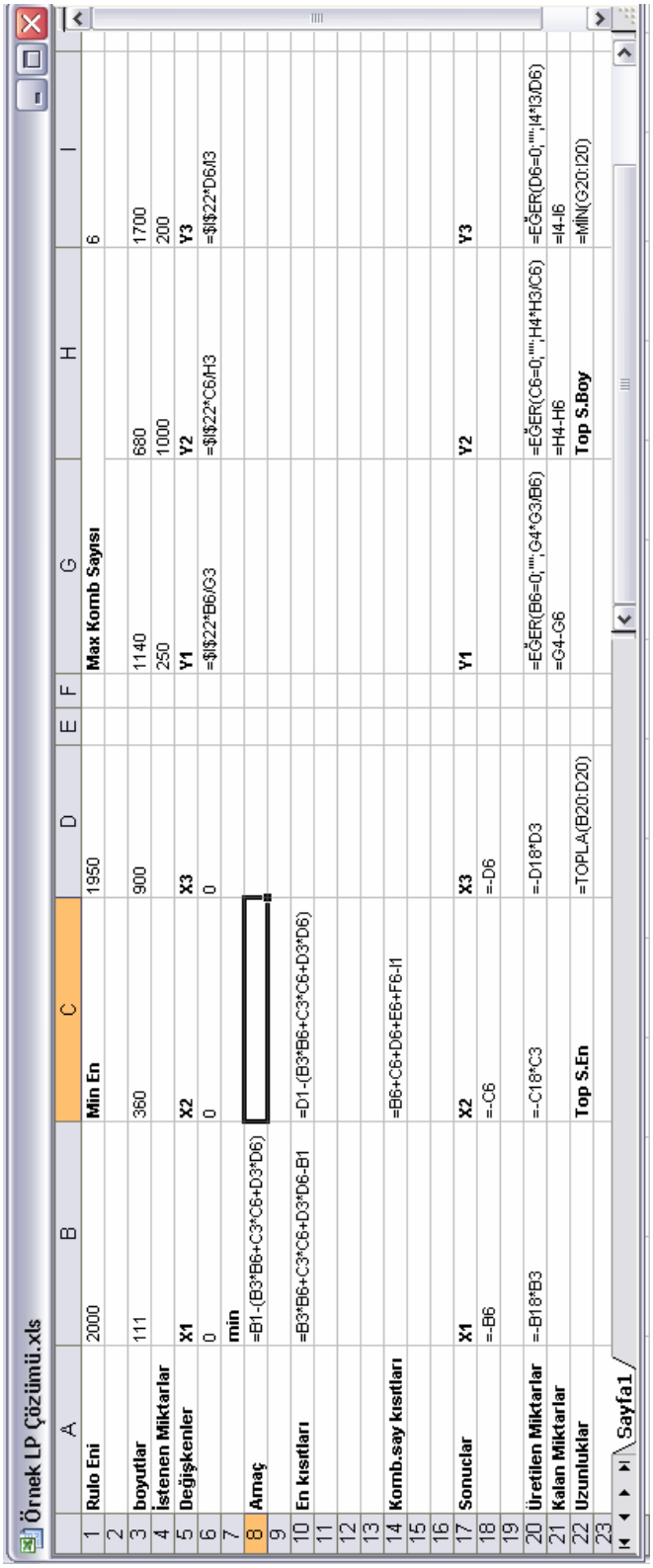
$$2000 - 50 - (111 .x_1 + 360 .x_2 + 900 .x_3) \geq 0$$

$$x_1 + x_2 + x_3 \leq 6$$

3. Pozitiflik Şartı:

$x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$ ve aynı zamanda tamsayıdır.

Buna göre Excel tasarımını şu şekilde yapalım.



	A	B	C	D	E	F	G	H	I
1	**Rulo Eni**	2000	**Min En**	1950			**Max Komb Sayısı**		6
2									
3	**boyutlar**	111	360	900			1140	680	1700
4	**İstenen Miktarlar**						250	1000	200
5	**Değişkenler**	**X1**	**X2**	**X3**			**Y1**	**Y2**	**Y3**
6		0	0	0			=$I$22*B6/G3	=$I$22*C6/H3	=$I$22*D6/I3
7		**min**							
8	**Amaç**	=B1-(B3*B6+C3*C6+D3*D6)							
9									
10	**En kısıtları**	=B3*B6+C3*C6+D3*D6-B1	=D1-(B3*B6+C3*C6+D3*D6)						
11									
12									
13									
14	**Komb.say kısıtları**		=B6+C6+D6+E6+F6-I1						
15									
16									
17	**Sonuclar**	**X1**	**X2**	**X3**			**Y1**	**Y2**	**Y3**
18		=-B6	=-C6	=-D6					
19									
20	**Üretilen Miktarlar**	=-B18*B3	=-C18*C3	=-D18*D3			=EĞER(B6=0;"";G4*G3/B6)	=EĞER(C6=0;"";H4*H3/C6)	=EĞER(D6=0;"";I4*I3/D6)
21	**Kalan Miktarlar**						=G4-G6	=H4-H6	=I4-I6
22	**Uzunluklar**		**Top S.En**	=TOPLA(B20:D20)				**Top S.Boy**	=MİN(G20:I20)
23									

Şekil 4-3 Örnek LP probleminin Excel tasarımı.

Excel LP Modelinin tasarımını yaparken dikkat etmemiz gereken noktalar şunlardır:

- Bulunması gereken değişkenlerin, yani x1, x2 ve x3 değişkenlerinin, değeri 0 olarak yazılır.
- Amaç fonksiyonu "=" (eşittir) ile başlayarak x e bağlı formülleri yazılır.
- Kaç adet kısıt var ise "=" (eşittir) ile başlayarak formüller ile yazılır.
- Rulo eni, maksimum fire, maksimum kombine sayısı gibi değerler bir hücreye yazılarak diğer formüllerde kullanılmak üzere kolaylık sağlanır. Bunlar değiştirmek gerektiğinde sadece bu alan değiştirilerek hem zamandan tasarruf edilir hem de hata olasılığını minimize ederiz.
- Şunu da belirtmek gerekirse, hangi kıstın hangi hücrede olduğunun bir önemi yoktur. Görsel olarak anlaşılır bir şekilde dizilimi yapılırsa okuyana kolaylık sağlar.
- Toplam safiha boyu (Top. S.Boy) alanı y değerlerini hesaplamak için gerekecektir. y değerlerinin formülü bu alana bağlı olarak yazılır.
- Üretilen miktarlar, kalan miktarlar ve toplam safiha eni (Top. S. En) değerleri de sonucu görmek için ekrana çıkmasını istediğimiz değerlerdir. Yazılması zorunlu değildir. Sonucu değiştirmez. Sonuçta bulmak istediğimiz şey x değerleri ve y değerleridir.

Kullanılan Formüller;

Amaç Fonksiyonu; minimize etmek istenilen değerler girilir

B8 = B1-(B3*B6+C3*C6+D3*D6)

Kısıtlar; model oluştururken hazırlanan kısıtlar ayrı ayrı hücrelere girilir.

B10 = B3*B6+C3*C6+D3*D6-B1

C10 = D1-(B3*B6+C3*C6+D3*D6)

C14 = B6+C6+D6-I1

Bu bilgilerin girilmesi programın çözümü için yeterlidir. Ancak, sonuçta elde edilen verilerden daha fazla bilgi edinebilmek amacı ile diğer birtakım hesaplamaları da Excel'e yaptırmak mümkündür. Örneğin; program sonucunda elde etmemiz gereken bilgilerden bir tanesi y değerleridir. y, değerleri safihadan alt alta kaç adet üretileceğini gösterir.

Toplan Safiha Boyu; üretilecek miktar ile safiha boyu b ile çarpılır ve x değerine bölünür. Bütün safihalar için bu bölüm hesaplanır ve bunların minimum değeri toplam safiha boyu olarak seçilir. Çünkü aynı anda ürünler üretilirken bir safiha üretimi tamamlandığında otomatik olarak diğerleri de tamamlanmış kabul edilir. Aksi takdirde, üretimi biten safiha için fazladan üretilme söz konusu olacaktır. Bu sebepten dolayı, uzunluk minimum hangi safihada oluyor ise kombinenin boyu olarak o boy seçilir ve diğerlerinin ne kadar üretileceği de ona göre hesaplanır.

G20 = EĞER(B6=0;"";G4*G3/B6)

H20 = EĞER(C6=0;"";H4*H3/C6)

I20 = EĞER(D6=0;"";I4*I3/D6)

Yukarıdaki formüller ile her bir safihanın boyu ile istenen miktarının çarpımını bulundu. Bu formüller, G20, H20 ve I20 hücrelerine yazıldı. Bu formüllerdeki EĞER fonksiyonu dikkat çekebilir. Bu eğer fonksiyonunu kullanmamızın sebebi; Excel, MİN fonksiyonunu çalıştırırken içerisinde boş olan hücreleri MIN fonksiyonuna dâhil etmez. Bu problemde de eğer bir safiha kombinasyonda bulunmuyor ise yani değeri 0 ise o safiha MIN içerisine dâhil edilmeyecektir.

I22 = MİN(G20:I20)

Bu bilgiler ışığında, toplam safiha boyu hesaplandığına göre, artık, her bir safihadan kaç tane üretebileceğimizi buluruz. Bunun için toplam safiha boyunun, safihanın boyuna (b) oranı ile o safihanın kullanım sayısının (x) çarpımına eşittir.

G6 = $I$22*B6/G3

H6 = $I$22*C6/H3

I6 = $I$22*D6/I3

Bir de bunlara ek olarak kombinasyon yapılıp üretildikten sonra her bir safihadan ne kadar kaldığını öğrenebilmek ister isek onları da rahatlıkla hesaplayabiliriz. Zira bu değer siparişten gelen miktar ile üretimde ortaya çıkan miktarın farkıdır.

G21 = G4-G6

H21 = H4-H6

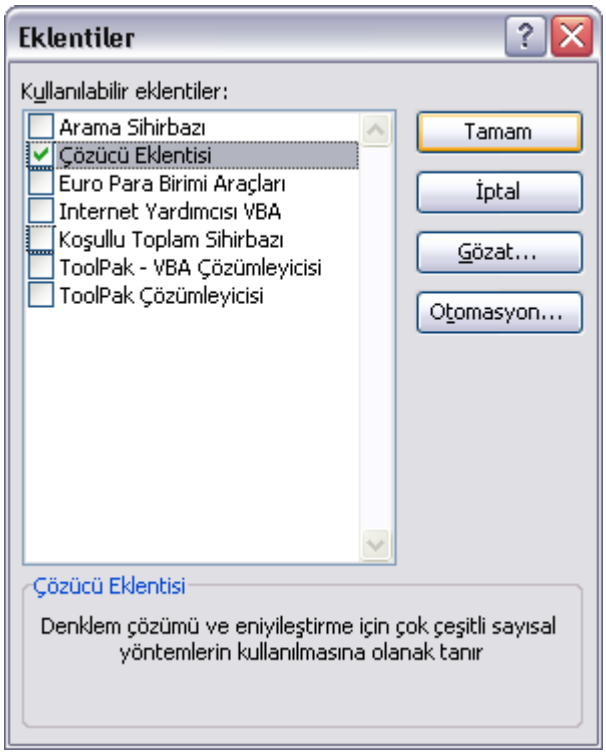
I21 = I4-I6

Çözücü (Solver)

LP problemlerini çözebilmek için Excel'in "Çözücü" eklentisi kullanılır. Araçlar menüsünde bulunur. (Bkz. Şekil 4.4)

Şekil 4-4 Excel'de çözücü (Araçlar > Çözücü).

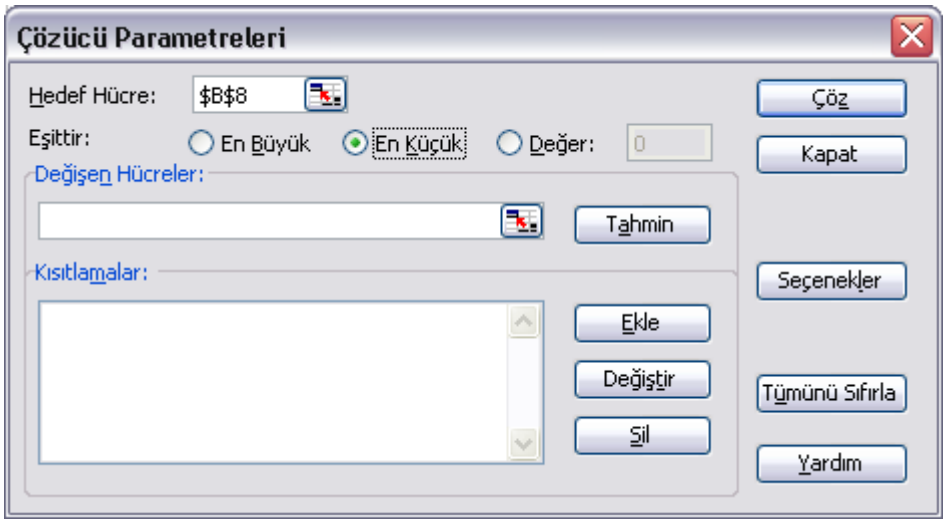
Genellikle, özellikle de çözücü hiç kullanılmamış ise, Araçlar menüsünde çözücü görünmez. Bunun sebebi, eklentinin yüklenmemiş olmasıdır. Eklentiyi yüklemek için yine Araçlar menüsünde görülen "Eklentiler…" seçeneğini seçerek eklentiler diyalog kutusu açılır.



Şekil 4-5 Excel'de eklentiler diyalog kutusu.

Şekil 4.5 de görülen diyalog kutusundan Çözücü eklentisi seçilir ve Tamam tıklanır (yada Enter tuşuna basılır). Bu işlemin ardından artık Araçlar menüsünde Çözücü artık görülür. Şimdi artık Excel'e girilen veriler çözücüye tanıtılabilir.

Çözücünün Kullanımı:

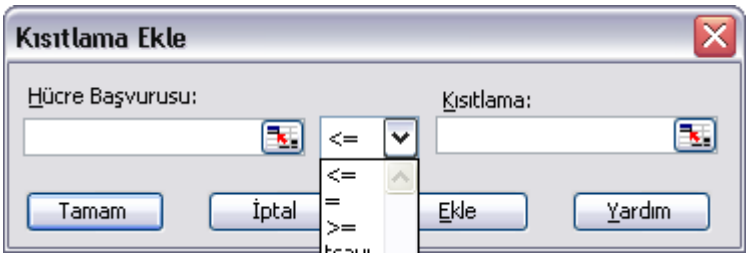


Şekil 4-6 Excel'in çözücüsü.

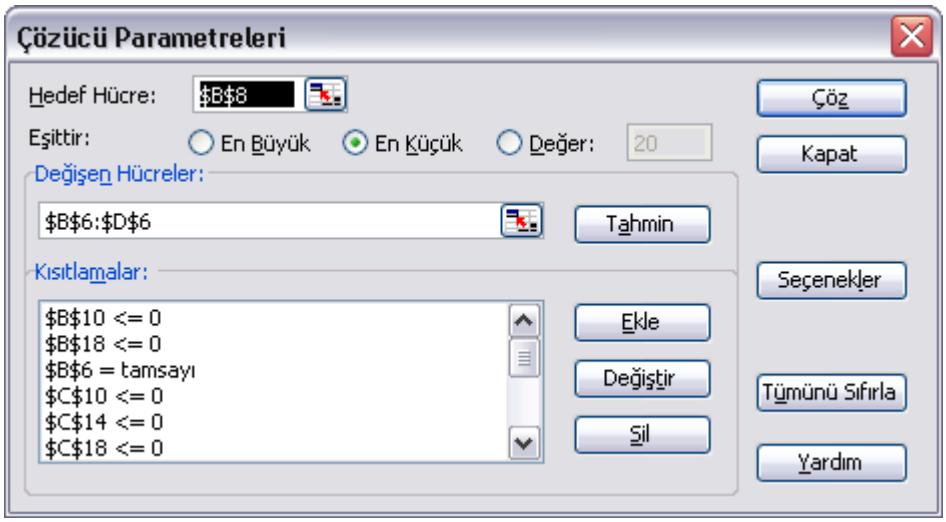
Excel'de çözücü açıldığında yukarıda görülen şekil 4.6'daki diyalog penceresi ile karşılaşılır.

Burada görülen terimleri kısaca açıklamak gerekirse;

- Hedef Hücre: Problemdeki amaç fonksiyonunu ifade eder.
- Eşittir: Bu ise amaç fonksiyonunun minimum, maksimum ya da bir değere ulaşma seçeneklerini ifade eder.
- Değişen Hücreler: Bulunması gereken bilinmeyenler olan x değişkenlerini ifade eder.
- Kısıtlamalar: Problemin bütün kısıtları buraya girilir. Ekle butonuna basarak Şekil 4.7'deki diyalog kutusundan her bir kısıt girilip sırayla eklenir. Orada girilen kısıtlar şekil 4.6'daki listeye dolar. Burada girmemiz gereken kısıtlar problemin kısıt denklemlerinin yanı sıra pozitiflik ve tamsayılılık kısıtlarını da ifade eder.

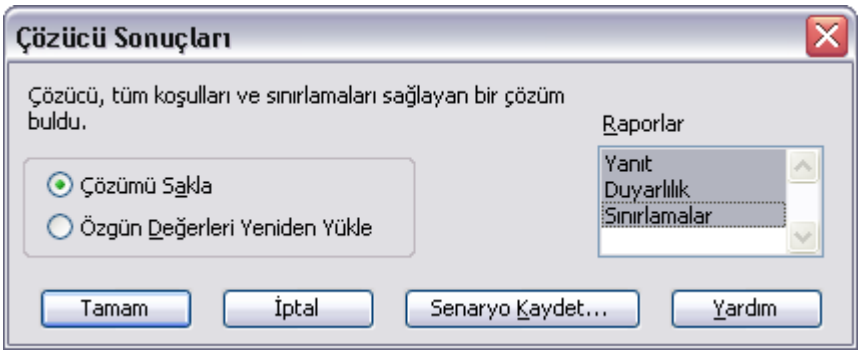


Şekil 4-7 Excel'de çözücüye kısıt eklenmesi.



Şekil 4-8 Excel'de çözücünün doldurulması.

Bu bilgilere dayanarak çözücü ayarları yapılabilir. Sonuçta Şekil 4.8 deki gibi bir görünüm elde ederiz. Bütün kısıtlar çözücüye girildi ise artık çöz butonuyla çözümü elde ederiz.



Şekil 4-9 Excel'de çözücünün sonuca ulaşması ve raporlama seçenekleri.

Excel, sonuca ulaşırsa yukarıdaki gibi bir mesaj kutusuyla kullanıcıya bilgi verir. Bu kutuda raporlama seçenekleri bulunur. 3 adet rapor vardır. Üç raporu da seçerek Tamam dediğimizde bize aşağıdaki tabloyu verir(Tablo 4.2). Böylece sonuç, duyarlılık analizi ile birlikte alınmış olur.

Sonuç olarak, hem oluşturulan LP modelinin doğruluğu tespit edildi hem de Excel ile bir LP probleminin nasıl çözüldüğü görüldü. Bir sonraki bölümde problemin kombinasyonel çözümü ve sonuçları üzerinde durulacaktır.

Tablo 4-2 Excel çözücüsünün sonuç raporu (duyarlılık analizi dahil).

Microsoft Excel 11.0 Yanıt Raporu						
Çalışma Sayfası: [Örnek LP Çözümü.xls]Sayfa1						
Rapor Oluşturuldu: 23.05.2007 20:00:50						
Hedef Hücre (En Küçük)						
	Hücre	**Ad**	**İlk Değer**	**Son Değer**		
	$B$8	Amaç min	20	20		
Ayarlanabilir Hücreler						
	Hücre	**Ad**	**İlk Değer**	**Son Değer**		
	$B$6	X1	0	0		
	$C$6	X2	3	3		
	$D$6	X3	1	1		
Sınırlamalar						
	Hücre	**Ad**	**Hücre Değeri**	**formül**	**Durum**	**Serbestlik**
	$B$10	En kısıtları min	-20	$B$10<=0	Farklı	20
	$C$10	En kısıtları X2	-30	$C$10<=0	Farklı	30
	$B$18	X1	0	$B$18<=0	Aynı	0
	$C$18	X2	-3	$C$18<=0	Farklı	3
	$D$18	X3	-1	$D$18<=0	Farklı	1
	$E$18		0	$E$18<=0	Aynı	0
	$F$18		0	$F$18<=0	Aynı	0
	$E$6		0	$E$6=tamsayı	Aynı	0
	$F$6		0	$F$6=tamsayı	Aynı	0
	$C$14	Komb.say kısıtları X2	-2	$C$14<=0	Farklı	2
	$B$6	X1	0	$B$6=tamsayı	Aynı	0
	$C$6	X2	3	$C$6=tamsayı	Aynı	0
	$D$6	X3	1	$D$6=tamsayı	Aynı	0

5. PROBLEMİN KOMBİNASYONEL ÇÖZÜMÜ

Kombine üretim problemi, bütün kombinasyonların tek tek karşılaştırılması ile bulunabilir. Şimdiki bölümde bu söyleneni gerçekleştiren algoritmalar oluşturulup incelenecektir. Ardından da ne gibi geliştirmeler yapılabilir onlardan bahsedilecektir. İlk önce kombinasyonlar nasıl elde edilir onu görelim.

5.1 Kombinasyonları Listeleme Algoritması

Basitçe anlatmak gerekirse n elemanlı dizi ile oluşturulabilecek kombinasyon sayısı;

$$\frac{(2.n)!}{n!.n!} - 1 \tag{5.1}$$

olarak ifade edilir. Örnek olarak;

2 elemanlı bir dizi olan a[2]={1,2} ile oluşturulabilecek kombinasyonlar;

Tablo 5-1 İki elemanlı kombinasyonların listesi.

#NO	Elemenlar	
1	1	
2	1	1
3	1	2
4	2	
5	2	2

Tablo 5-2 Üç elemanlı kombinasyonları listesi.

#NO	Elemenlar		
1	1		
2	1	1	
3	1	1	1
4	1	1	2
5	1	1	3
6	1	2	
7	1	2	2
8	1	2	3
9	1	3	
10	1	3	3
11	2		
12	2	2	
13	2	2	2
14	2	2	3
15	2	3	
16	2	3	3
17	3		
18	3	3	
19	3	3	3

Burada 2-1 olan kombinasyon görülmemektedir. Bunu sebebi, 1-2 olan kombinasyonla aynı şeyi ifade ettiği için tekrardan onu yazmaya gerek yok. Bu şekilde düşündüğümüz zaman (5.1) denkleminde verilen önermenin doğru olduğu görülür. Teyit için bir örnek daha vermekte fayda vardır. Tablo 5.2 ye bakılırsa aynı olayın orda da gerçekleştiği görülür. Burada 3 elemanlı kombinasyon sayısı 19'dur. (5.1) denkleminde n yerine 3 koyarak bu sonuca ulaşabiliriz.

$$\frac{6!}{3!.3!} - 1 = \frac{6!}{3!.3!} - 1 = \frac{6.5.4.3!}{3.2.1.3!} - 1 = 5.4 - 1 = 19$$

Benzer şekilde 4 elemanlı kombinasyonların sayısı 69, 5 elemanlı kombinasyon sayısı 251 dir.

Algoritması şu şekilde yapılmıştır.

1) F(int dizi[],int yer,int iyer)
2) Eğer yer >= Kombinasyonda bulunacak eleman sayısı ise çık
3) i ye kombinasyonun hangi elemanından itibarek ekleme yapıldığını ata (i=iyer)
4) kombinasyonun o anki pozisyonuna (yer) dizinin i. Elemanını ata
5) kombinasyonun pozisyonundan sonraki yerleri sıfırla
6) kombinasyonu ekrana bas
7) F(dizi,yer+1,iyer) fonksiyonu çağır
8) i yi artır
9) i < N ise 4.adıma git değilse çık

C dilinde yazılmış hali de şu şekildedir.

```
void F(int dizi[],int yer,int iyer)
{
    int i;
    if (yer >= M)
        return;

    for(i=iyer;i<N;i++)
    {
        dizi[yer]=a[i];
        sifirla(dizi,yer);
        printdizi(dizi);
        F(dizi,yer+1,i);
    }
}

void main()
{
    int ar[N]={0};
    F(ar,0,0);
}
```

Bu algoritma, tarafımdan yazılmıştır. Bu algoritmayı kullanarak kombinasyonları deneyecek

bir program ile tezi gerçekleştirip görelim.

5.2 Programın Arayüzü

Program temel birkaç işlemin yanı sıra kombinasyon işlemini yapmak üzere tasarlanmıştır. Şekil 5.1 de görüldüğü gibi programa sipariş girişleri kullanıcı tarafından (1-Yeni Giriş) yada dosyadan okutmak (2- Dosyadan Oku) suretiyle yapılabilmektedir. Bu girişleri listelemek için "3-Listele" komutu verilir. Okunan sipariş ve bobin bilgileri ekrana bastırılır. "4-Ayarlar" seçeneği ile kombine ayarları ekrana yazdırılır. Ve bütün hazırlıklar tamam ise 5- Kombineyi Başlat menüsü ile kombine işlemi başlar ve en uygun kombineyi ekrana liste halinde bastırır. Kullanıcı işi bittiğinde "0- Çıkış" ile programa son verir. Şimdi bu menülerin içeriklerini daha detaylı görelim.

Şekil 5-1 Kombine programının arayüzü.

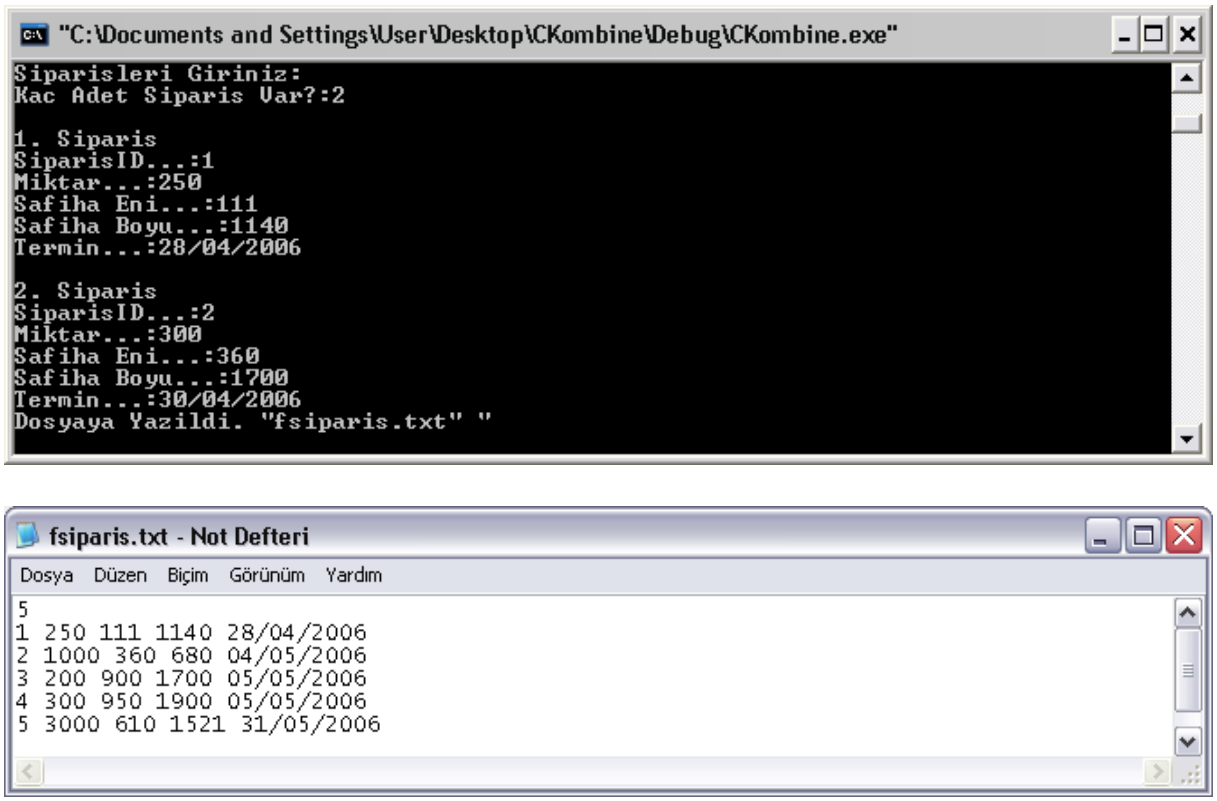
5.3 Bilgilerin Programa Girilmesi

1-Yeni Giriş seçeneği ile menüye girilir. Öncelikler Bobinlerin girilmesi istenecektir. Daha sonra da siparişlerin bilgileri kullanıcıya sorulacaktır.

Bobin Girişlerinde öncelikle kaç adet bobin girişi yapılacağını öğrenip daha sonra bu kadar adet bobin eni kullanıcıdan istenir. Kullanıcı bobinleri girdikten sonra Bu girilen bilgiler daha sonra ulaşılabilmek amacıyla "fbobin.txt" dosyasına kaydedilir. Kayıt işleminin tamamlanması ile sipariş girişi kısmına geçilir.

Şekil 5-2 Bobin girişleri.

Sipariş Girişlerinde de bobinlerde olduğu gibi önce kaç adet sipariş girileceği öğrenilir. Sonra da sırayla Sipariş ID, Miktar, Safiha Eni, Safiha Boyu, ve Termin bilgileri girilir. Bilgi girişleri tamamlandıktan sonra daha sonra ulaşılabilmek amacıyla "fsiparis.txt" dosyasına kaydedilir.



Şekil 5-3 Sipariş girişleri.

5.4 Bilgilerin Dosyadan Okunması

Menüde 1. seçenek seçildiğinde bilgi girişlerinin ardından girilen kayıtlar dosyaya yazılmıştı. Bu bilgileri dosyadan okumak için şu yapı kullanılır.

İlk enter karakterine kadar okunan kayıt dosyadaki satır sayısını verir. Bu sayı "bobinadedi" olarak isimlendirildi. "bobinadedi" kadar satır dosyadan okunur ve bobineni olarak hafızaya alınır.

```
struct _Bobin{
    int BobinEni;
};
```

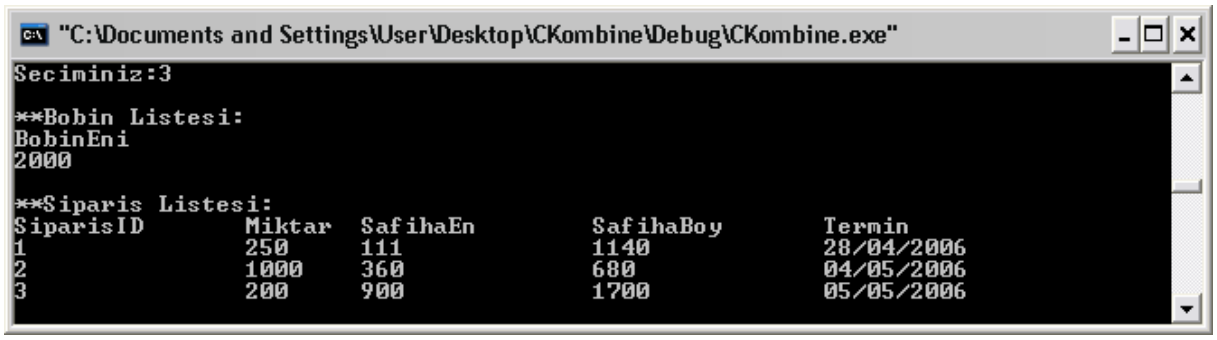
Sipariş kayıtları da aynı düzende kaydedilmiştir. İlk ENTER karakterine kadar olan bilgi "siparisadedi" olarak isimlendirilir. Ve o kadar kayıt siparişler olarak hafızaya alınır.

```
struct _Siparis{
    int SiparisID;
    int Miktar;
    int SafihaEn;
    int SafihaBoy;
    char Termin[11]; // "dd/mm/yyyy" Formatinda girilecek
    int TerminD,TerminM,TerminY;
};
```

Şekil 5-4 Bilgilerin dosyadan okunması.

5.5 Girilen bilgilerin Listelenmesi

Girilen Bilgiler ekranda kullanıcıya aşağıdaki şekilde gösterilir. Önce bobin listesi hemen ardından da sipariş listesi ekrana bastırılır.



Şekil 5-5 Bilgilerin listelenmesi.

5.6 Kombinasyonların Denenmesi ve Optimum Olanın Seçilmesi

Buraya kadar olan kısımlar kombinasyon işleminin başlaması için gereken hazırlık safhalarıydı. Asıl konu olan optimizasyonun yapılacağı kısım burasıdır. Bu kısımda verilen siparişlerden yola çıkarak bütün kombinasyonlar denenecek ve bobin eni kısıtına ve maksimum fire kısıtına uyan bütün kombinasyonlardan en az fireyi veren kombinasyon seçilecektir.

Excel örneğindeki aynı dataları kullanarak problemi çözmeye çalışalım. Daha sonra sipariş sayısını artırarak tekrar sonucu görebiliriz.

Şekil 5.5 de görülen siparişler için kombine işlemini başlatalım.

Şekil 5-6 Kombine işlemi sonucu.

Kombine işlemi başarıyla tamamlandı. Şekil 5-6 da görüldüğü üzere 1 numaralı kombinede 4 adet sipariş var. Bunların sipariş ID'leri sırasıyla 2,2,2 ve 3 dür. Yani bu demek oluyor ki ilk üc sırada 2 no'lu sipariş sonraki sırada 3 no'lu sipariş üretilecektir.

Şimdi bir de sipariş sayısını arttırarak kombinasyon işlemini başlatalım.

Tablo 5-3 Kombine testi için 5 adet sipariş.

SiparisID	Miktar	SafihaEn	SafihaBoy	Termin
1	250	111	1140	28.04.2006
2	1000	360	680	04.05.2006
3	200	900	1700	05.05.2006
4	300	950	1900	05.05.2006
5	3000	610	1521	31.05.2006

Yukarıdaki Tablo 5.3 de listelenmiş olan 5 adet sipariş için kombine işlemi başlatıldığında optimum kombinasyonun 1,2,3,5 nolu siparişlerden oluşan kombinasyon sonucunu alıyoruz. Gerçekten de toplam safiha eni 1981 olan bu kombinasyon diğer kombinasyonlar arasında en optimal sonuçtur. Toplam safiha eni istenen aralıkta olan kombinasyon listesini program içerisinde görübiliriz.

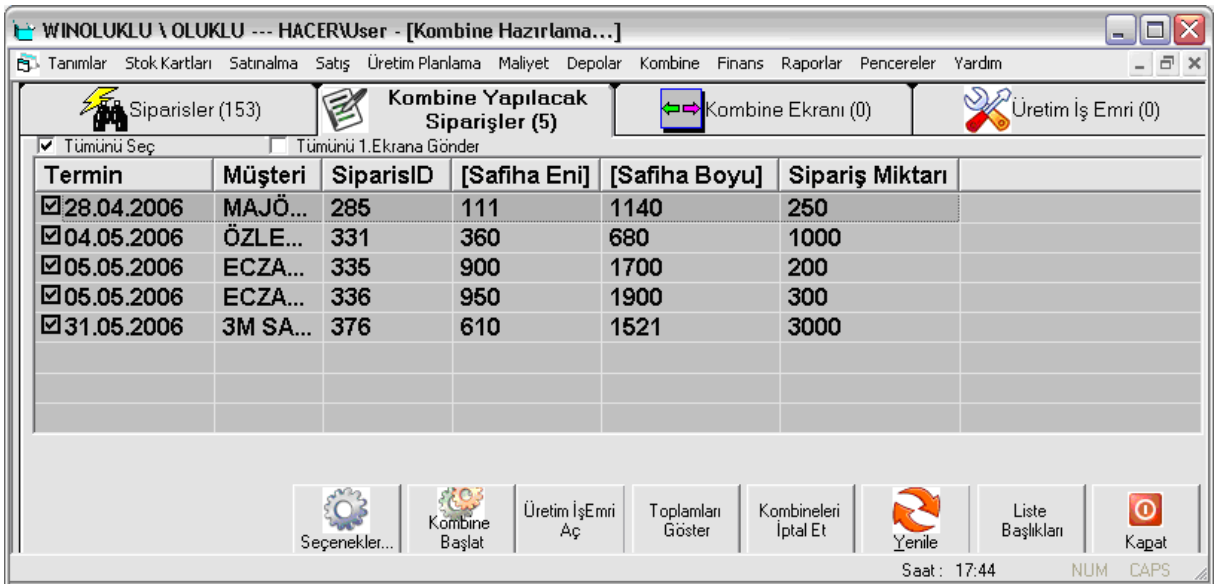
Şekil 5-7 Kombinasyon için ikinci bir örnek.

5.7 Sonuçların Değerledirilmesi

Bu sonuçlarla Excel LP çözümündeki sonuçları karşılaştıralım. Orada da aynı sonuçların elde edildiğini göreceksiniz. Böylece her iki yöntem de aynı sonucu veriyor. İstenildiği kadar sipariş üzerinde test edilirse edilsin sonuç değişmeyecektir. Çünkü kurulan model optimumluğu gerçekten sağlıyor. Eğer kombinasyonlardan hiçbirisi istenen koşullara uymuyor olsaydı hem program hem de Excel çözümü Çözüm bulamadığını gösterecekti.

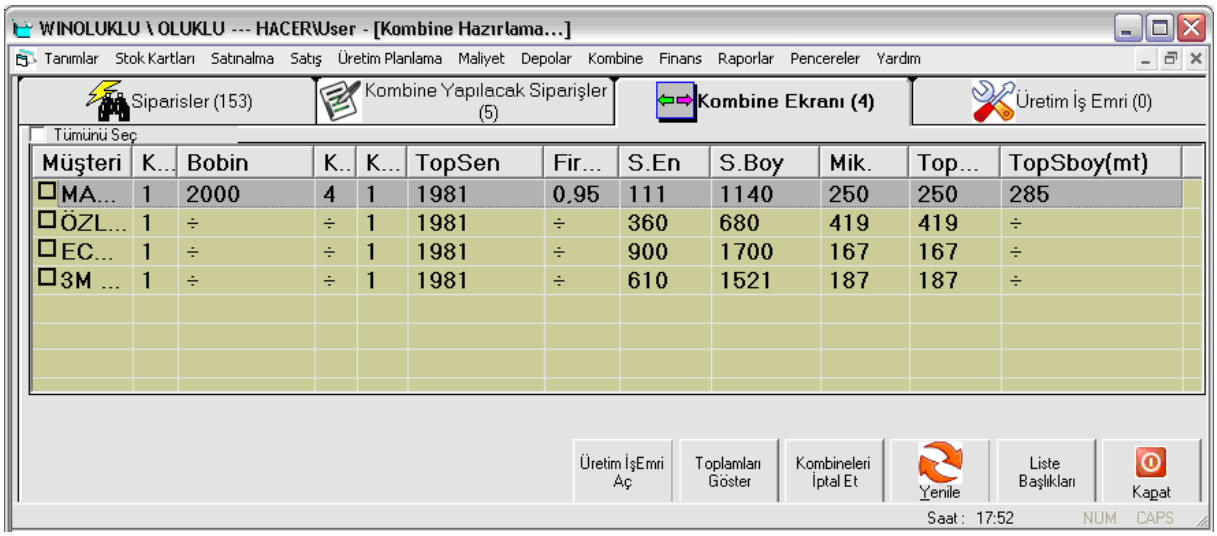
6. PROBLEMİN GERÇEK HAYATA UYGULANMASI

Bu çalışmanın ardından elde edilen sonuçların tatmin ediciliğini gördükten sonra algoritma, oluklu mukavva sektörüne çözümler üreten bir firma vasıtası ile var olan ERP programına entegre edildi.



Şekil 6-1 Entegrasyon Projesi - Kombine yapılacak siparişler.

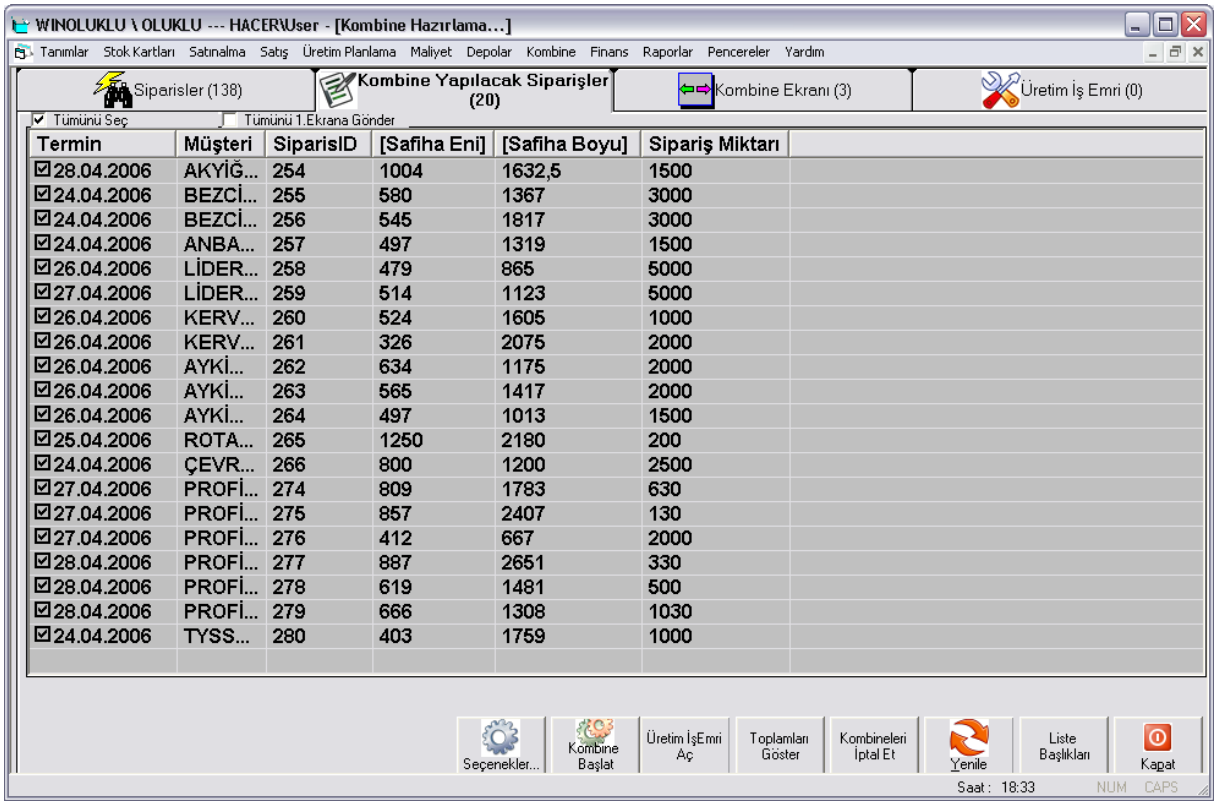
İkinci ekran olan kombine yapılacak siparişler bölümünde kombine yapılması istenen siparişlerin önüne işaret konuyor. Kombine başlat butonuyla beraber kombine işlemi başlıyor. Test dataları olarak Tablo 5.3 deki örnek ele alındı. Sonuç olarak aynı kombinasyon elde edildi. Görüldüğü gibi %0.95 fire ile üretebilecek ürünü tespit etmiş olduk.



Şekil 6-2 Entegrasyon Projesi - Yapılan kombine.

6.1 Hızı Etkileyen Faktörler

Hızı etkileyen faktörlerin başında kombinasyon yapılmak istenen sipariş sayısı gelmektedir. Sipariş sayısı arttığında hız da üstel fonksiyon şeklinde artmaktadır. Önerilen algoritmadaki kombinasyon sayısı, normal kombinasyondan biraz daha az sayıda kombinasyon denediği için hız olarak avantaj sağlanmıştır.



Şekil 6-3 Yirmi sipariş seçildi.

Yukarıdaki Şekil 6.3'de yirmi siparişin seçildiği görülmektedir. 20 sipariş üzerinde yapılan denemede 3,797 saniyede sonuca ulaşıldı. Ulaşılan sonuç %0 fire ile üretilen bir kombinasyondur.

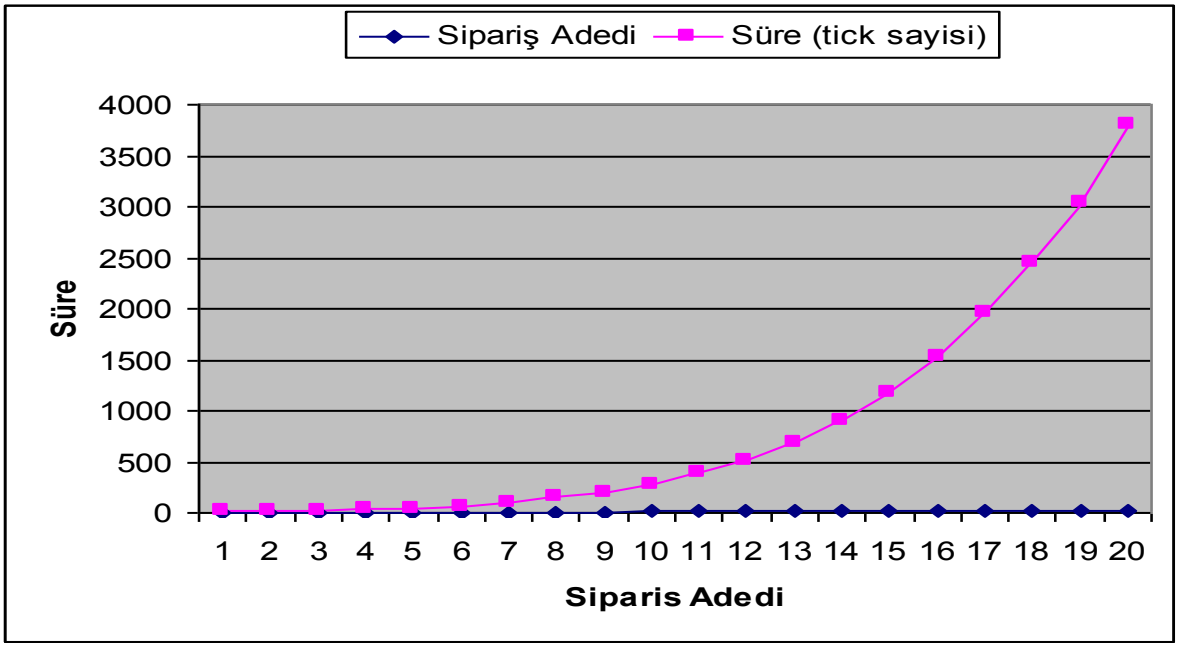
Şekil 6-4 Yirmi siparişin optimum kombinasyonu.

Yukarıdaki sonuca göre üretim yapan bir firma hiç fire vermeden bu siparişlerden üretim yapabilir. Böylece firma, maliyeti, fire vermek koşuluyla arttırmadan üretimini gerçekleştirebilir. Ya daha fazla kâr elde eder ya da fiyatı düşürerek müşteriyi elde tutma yolunda kararlar verebilir.

Tablo 6-1 Sipariş adedine göre sonuç bulma süreleri.

Sipariş Adedi	Süre (Tick Sayisi*)
1	16
2	16
3	16
4	31
5	47
6	63
7	94
8	156
9	203
10	266
11	390
12	516
13	687
14	906
15	1172
16	1531
17	1953
18	2453
19	3031
20	3797

* Tick sayısı Windows'un kendi zamanını hesaplaması için kullandığı birimdir. 1000 tick = 1 saniyeye denk gelmektedir.



Şekil 6-5 Siparis adetlerine göre sonuca ulaşma sürelerini gösteren grafik.

Yapılan zaman testlerinde seçilen sipariş sayısı ve sonuca ulaşma zamanına bağlı grafik yukarıdaki Şekil 6.5'de gösterilmiştir. Bu sürelere bakarak üstel bir şekilde arttığını görebiliriz. Bunun sebebi kombinasyonların üstel artan fonksiyon olmasıdır.

7. SONUÇLAR ve ÖNERİLER

Tez Çerçevesinde, oluklu mukavva ve kutu üreten firmaların en büyük problemlerinden birisi olan kombinasyon firesinin minimum hale getirmek için çalışıldı. Bu çalışmada iki yol önerildi ve ikisinde de başarılı sonuçlar elde edildi.

Birinci yol problemin LP modelini kurarak lineer programlama ile çözülebileceği yönünde idi. Bu amaca ulaşmak için öncelikle LP modeli kuruldu ve örnek veriler ile sonuca ulaşıldığı görüldü.

İkinci yol olarak bir algoritma geliştirildi. Bu algoritma sayesinde bütün kombinasyonlar tek tek deneyerek optimum sonuç bulunmaya çalışıldı. Bu denemeler sırasında vakitten tasarruf etmek için daha önce denenmiş siparişlerin kombinasyonu hiç denenmedi ve firesi istenen aralık dışında olan siparişler, diğer koşullara bakılmaksızın reddedildi.

20 adet siparişin 3,797 sn gibi kısa bir sürede taranmasından sonra sonuca ulaşılması, önerilen algoritmanın oldukça hızlı olduğunu göstermektedir. El yordamı ile dakikalar süren 20 adet siparişin uygulaması bilgisayarda yazılan algoritma ile saniyeler içinde sonuç vermektedir.

Firma, gözden kaçmayan bu %0'a yakın fireler sayesinde büyük bir kâra geçmiştir. Program kullanılmadan önce aylık ortalama fireleri %6 - %7 civarında seyrederken, program ile birlikte %1,5 - %2'lere kadar düşmüştür. Bir firmanın ortalama yıllık üretimi 1.000.000 ton olduğu düşünülürse oldukça büyük bir kazanç sağlamış olduğu görülür.

Tez kapsamında önerilen algoritma tam olarak istenileni yapmaktadır. Ancak program gelişmeye açıktır. Programda en az fireyi veren sipariş en iyi olarak seçilmiştir. Buna ek olarak fire sayıları eşit fakat kombine derecesi daha fazla olan veya toplam safiha boyu daha fazla olan kombinasyon seçme özellikleri de eklenebilir.

Bu konuda yapılacak araştırmaların mümkünse bizzat üretimin yapıldığı firmada çalışarak iyi bir gözlemle yapılması gerekmektedir. Kombine üretimin koşulları ancak bu şekilde doğru tespit edilebilir.

KAYNAKLAR

Alan, M.A, Yeşilyurt, C., (2004), "Doğrusal Programlama Problemlerinin Excel ile Çözümü", C.Ü. İktisadi ve İdari Bilimler Dergisi, Cilt 5, Sayı 1, pp. 151-162

Campbell, P. J., (1977), "Gauss and the 8-Queens Problem: A Study in the Propagation of Historical Error." Historia Math. 4, pp. 397-404.

Chowdhury, S., (1989), "Analytical Approaches to the Combinatorial Optimization in Linear Placement Problems", IEEE Transactions on computer-aided design, Vol. 8, No.6, pp. 630-639.

Dauer, J.P., (1991), "Optimization over the efficient set using an active constraint approach", Journal of Mathematical Methods of Opertaional Research, Vol. 35, No. 3, pp. 185-195.

Gilmore, P.C. and Gomory, R.E., (1961), "A linear Programming Approach to the Cutting Stock Problem", Operations Research, V. 9, No.6, pp. 849-859.

Haessler, R.W, Sweeney, P.E., (1991), "Cutting Stock Problems and Solution Procedures", European Journal of Operation research, Vol. 54, No. 2, pp. 141-150.

Horst, R., Thoai, N.V., Yamamoto, et.al, (2007), "On Optimization over the Efficient Set in Linear Multicriteria Programming", Forthcoming in Journal of Optimization Theory and Applications, Vol. 134, No 1

Puchinger, J., Raidl, G.R., Koller, G., (2004), "Solving a real-world glass cutting problem", Proceedings of the Fourth International Conference on Combinatorial Optimization (EvoCOP 2004).

Trubin, V.A., Grad, I.N., (1977), "Method for optimal cloth cutting", Journal of Cybernatics and System Analysis, ISBN 1060-0396 (Print) 1573-8337(Online), pp. 601-605.

Watkins, J., (2004), "Across the Board: The Mathematics of Chess Problems", Princeton: Princeton University Press, ISBN 0-691-11503-6.

EKLER

Ek 1 Kombine üretim problemini çözün C programı Kaynak kodu

EK 1 Kombine Üretim Problemini Çözen C Programının Kaynak Kodu

```
#include <stdio.h>
#include <stdlib.h> //malloc icin
#include <ctype.h> //isDigit icin
#include <math.h> //pow icin

typedef struct _Siparis Siparis;
typedef struct _Kombine Kombine;
typedef struct _Bobin Bobin;
typedef struct _Ayar Ayar;

struct _Bobin{
	int BobinEni;
};

struct _Siparis{
	int SiparisID;
	int Miktar;
	int SafihaEn;
	int SafihaBoy;
	char Termin[11]; // "dd/mm/yyyy" Formatinda girilecek
	int TerminD,TerminM,TerminY;
};

struct _Kombine{
	int KombineNo;
	int KombineDerecesi;
	int KombineFiresi;
	int KombineUzunluk;
	int BobinEni;
	int SiparisID;
```

```
        int SipMiktar;
        int Adet;
};

struct _Ayar{
        int MaxFire;
        double FireDuyarlilik;
        int MinUzunluk;
        double UzunlukDuyarlilik;
        int SelfKombine;
        int MaxKombine;
};

const char fBobin[]="fbobin.txt";
const char fSiparis[]="fsiparis.txt";
const char fKombine[]="fkombine.txt";

int SiparisAdedi=0;
int BobinAdedi=0;
int KombineAdedi=0;
int MaxKombine=0;

Kombine *K,*KBest,*KTest;
Siparis *S;
Bobin *B;
Ayar A;
int *secilen;


int isTarih(Siparis *S);
void Yeni_S();
```

```
void Yeni_B();
void Dosyadan_S();
void Dosyadan_B();
void listele_S();
void listele_B();
void listele_K();
void Listele_A();

void KombineBaslat();
void KombYap(int yer,int iyer);
void sifirla(int yer);



void main()
{
	char ch;

	A.MaxFire=50;
	A.FireDuyarlilik=0.01;
	A.MinUzunluk=100;
	A.UzunlukDuyarlilik=0.015;
	A.SelfKombine=0;
	A.MaxKombine=6;


	do{
		printf("\n\n**********\nMENU\n**********\n");
		printf("1-Yeni Giris\n");
		printf("2-Dosyadan Oku\n");
		printf("3-Listele\n");
```

```
printf("4-Ayarlar\n");
printf("5-Kombineyi Baslat\n");
printf("0-Cikis\n");
printf("\n");
printf("Seciminiz:");
fflush(stdin);
scanf("%c",&ch);
switch(ch){
case '1': //Yeni Giriş
        Yeni_B();
        Yeni_S();
        break;
case '2': //Dosyadan Oku
        Dosyadan_S();
        Dosyadan_B();
        break;
case '3': //Listele
        listele_B();
        listele_S();
        break;
case '4': //Ayarlar
        Listele_A();
        break;
case '5': //Kombineyi Baslat
        KombineBaslat();
        break;
case '0': //cıkış
        break;
default:
        printf("\nLutfen Menudeki Seceneklerden Secin\n\n");
}
```

```
        }while(ch != '0');
}

int isTarih(Siparis *S){
        char dd[3],mm[3],yyyy[5];
        int d,m,y;
        int us,as;
        if (sizeof(S->Termin)/sizeof(S->Termin[0]) != 11 )
                return false;
        dd[0]=S->Termin[0];
        dd[1]=S->Termin[1];
        dd[2]=NULL;

        mm[0]=S->Termin[3];
        mm[1]=S->Termin[4];
        mm[2]=NULL;

        yyyy[0]=S->Termin[6];
        yyyy[1]=S->Termin[7];
        yyyy[2]=S->Termin[8];
        yyyy[3]=S->Termin[9];
        yyyy[4]=NULL;

        if(isdigit(dd[0]))          d=atoi(dd);          else          return false;
        if(isdigit(mm[0]))          m=atoi(mm);          else          return false;
        if(isdigit(yyyy[0]))  y=atoi(yyyy);  else          return false;

        as=1;
        switch(m){
        case 1:
                us=31;          break;
```

```
case 2:
        if(y%4) us=28; else us=29;
        break;
case 3:
        us=31;          break;
case 4:
        us=30;          break;
case 5:
        us=31;          break;
case 6:
        us=30;          break;
case 7:
        us=31;          break;
case 8:
        us=31;          break;
case 9:
        us=30;          break;
case 10:
        us=31;          break;
case 11:
        us=30;          break;
case 12:
        us=31;          break;
default:
        return false;
}
if (d<as || d>us) return false;
if (y<2000 || y>2100) return false;

S->TerminD=d;
S->TerminM=m;
```

```
	S->TerminY=y;
	return true;
}

void Yeni_S(){
	int i;
	FILE *fp;
	printf("Siparisleri Giriniz:\n");
	printf("Kac Adet Siparis Var?:");
	scanf("%d",&SiparisAdedi);
	S=(Siparis *) malloc(sizeof(Siparis)*SiparisAdedi);
	for(i=0;i<SiparisAdedi;i++) {
		printf("\n%d. Siparis\n",i+1);
		printf("SiparisID...:");
		scanf("%d",&S[i].SiparisID);
		printf("Miktar...:");
		scanf("%d",&S[i].Miktar);
		printf("Safiha Eni...:");
		scanf("%d",&S[i].SafihaEn);
		printf("Safiha Boyu...:");
		scanf("%d",&S[i].SafihaBoy);
		do{
			printf("Termin...:");
			scanf("%s",S[i].Termin);
			if (isTarih(&S[i]))
				break;
			else
				printf("Gecersiz Tarih. dd/mm/yyyy (or: 01/01/2007) olarak yeniden giriniz\n");
		}while(1);
	}
	printf("Dosyaya Yaziliyor. \"%s\" ",fSiparis);
```

```
	fp=fopen(fSiparis,"w+");
	if (!fp) { printf("Dosya acilamadi!!!"); exit(0);}

	fprintf(fp,"%d\n",SiparisAdedi);
	for(i=0;i<SiparisAdedi;i++){
		fprintf(fp,"%d ",S[i].SiparisID);
		fprintf(fp,"%d ",S[i].Miktar);
		fprintf(fp,"%d ",S[i].SafihaEn);
		fprintf(fp,"%d ",S[i].SafihaBoy);
		fprintf(fp,"%s\n",S[i].Termin);
	}
	printf("\rDosyaya Yazildi. \"%s\" \n",fSiparis);
	fclose(fp);
}

void Yeni_B(){
	int i;
	FILE *fp;
	printf("Bobinleri Giriniz:\n");
	printf("Kac Adet Bobin Var?:");
	scanf("%d",&BobinAdedi);
	B=(Bobin *) malloc(sizeof(Bobin)*BobinAdedi);
	for(i=0;i<BobinAdedi;i++) {
		printf("\n%d. BobinEni...:",i+1);
		scanf("%d",&B[i].BobinEni);
	}
	printf("Dosyaya Yaziliyor. \"%s\" ",fBobin);
	fp=fopen(fBobin,"w+");
	if (!fp) { printf("Dosya acilamadi!!!"); exit(0);}
	fprintf(fp,"%d\n",BobinAdedi);
	for(i=0;i<BobinAdedi;i++){
```

```
                fprintf(fp,"%d\n",B[i].BobinEni);
        }
        printf("\rDosyaya Yazildi. \"%s\" \n",fBobin);
        fclose(fp);
}

void  Dosyadan_S(){
        int i;
        FILE *fp;
        printf("Siparisler Okunuyor:\n");
        fp=fopen(fSiparis,"r+");
        if (!fp) { printf("Dosya acilamadi!!!"); exit(0);}
        fscanf(fp,"%d\n",&SiparisAdedi);
        S=(Siparis *) malloc(sizeof(Siparis)*SiparisAdedi);
        for(i=0;i<SiparisAdedi;i++){
                fscanf(fp,"%d",&S[i].SiparisID);
                fscanf(fp,"%d",&S[i].Miktar);
                fscanf(fp,"%d",&S[i].SafihaEn);
                fscanf(fp,"%d",&S[i].SafihaBoy);
                fscanf(fp,"%s",S[i].Termin);
                printf("%d ",S[i].SiparisID);
                printf("%d ",S[i].Miktar);
                printf("%d ",S[i].SafihaEn);
                printf("%d ",S[i].SafihaBoy);
                printf("%s\n",S[i].Termin);
                if (!isTarih(&S[i])) { printf("Tarih Bicimi yanlis!!!"); exit(0);}
        }
        fclose(fp);
}
void Dosyadan_B(){
        int i;
```

```
	FILE *fp;
	printf("Bobinler Okunuyor:\n");
	fp=fopen(fBobin,"r+");
	if (!fp) { printf("Dosya acilamadi!!!"); exit(0);}
	fscanf(fp,"%d",&BobinAdedi);
	B=(Bobin *) malloc(sizeof(Bobin)*BobinAdedi);
	for(i=0;i<BobinAdedi;i++){
		fscanf(fp,"%d",&B[i].BobinEni);
		printf("%d\n",B[i].BobinEni);
	}
	fclose(fp);
}
void listele_S(){
	int i;
	printf("\n**Siparis Listesi:\n");
	printf("SiparisID\tMiktar\tSafihaEn\tSafihaBoy\tTermin\n");
	for (i=0;i<SiparisAdedi;i++){
		printf("%d\t\t",S[i].SiparisID);
		printf("%d\t",S[i].Miktar);
		printf("%d\t\t",S[i].SafihaEn);
		printf("%d\t\t",S[i].SafihaBoy);
		printf("%s\n",S[i].Termin);
	}
}

void listele_B(){
	int i;
	printf("\n**Bobin Listesi:\n");
	printf("BobinEni\n");
	for (i=0;i<BobinAdedi;i++){
		printf("%d\n",B[i].BobinEni);
```

```
        }
}
void listele_K(){
        int i,j;
        printf("\n**Kombine Listesi:\n");
        printf("K. No\tSaf.ID\tFire\n");
        for (i=0;i<KombineAdedi;i++){
                for(j=0;j<KBest[i].KombineDerecesi;j++)
                {
                        printf("%d\t",KBest[j].KombineNo);
                        printf("%d\t",KBest[j].SiparisID);
                        printf("%d\n",KBest[j].KombineFiresi);
                }
        }
}
void Listele_A(){
        printf("\n**Ayarlar:\n");
        printf("Max Fire: %d\n",A.MaxFire);
        printf("Fire Duyarlilik: %lf\n",A.FireDuyarlilik);
        printf("Min Uzunluk: %d\n",A.MinUzunluk);
        printf("Uzunluk Duyarlilik: %lf\n",A.UzunlukDuyarlilik);
        printf("Self Kombine: %d\n",A.SelfKombine);
        printf("Max Kombine: %d\n",A.MaxKombine);
}


void KombineBaslat(){
        int i;

        MaxKombine=A.MaxKombine;
```

```
	secilen=(int *) malloc(sizeof(int)*MaxKombine);
	for(i=0;i<MaxKombine;i++)
		secilen[i]=-1;

	KombYap(0,0);
	listele_K();
}

void KombYap(int yer,int iyer){
	int i,j,k,topsen,topsboy;
	if (yer >= MaxKombine)
		return;

	for(i=iyer;i<SiparisAdedi ;i++)
	{
		if (S[i].Miktar<=0)
			continue;
		secilen[yer]=i;
		sifirla(yer);

		if(secilen[0]==1)
			printf("");

		//en iyi kombine mi diye bakalım
		topsen=0;topsboy=0;
		for(j=0;j<MaxKombine;j++)
		{
			if(secilen[j]==-1)
				continue;
			topsen+=S[secilen[j]].SafihaEn;
		}
```

```
if(topsen >= B[0].BobinEni-A.MaxFire && topsen <= B[0].BobinEni)
{
	printf("Top. S.En: %d \n",topsen);
	if (KombineAdedi==0)
	{
		KombineAdedi=1;
		KBest=(Kombine *) malloc(sizeof(Kombine)*(yer+1));
		for(k=0;k<yer+1;k++)
		{
			KBest[k].BobinEni=B[0].BobinEni;
			KBest[k].KombineDerecesi=yer+1;
			KBest[k].KombineNo = KombineAdedi;
			KBest[k].BobinEni=B[0].BobinEni;
			KBest[k].SiparisID=S[secilen[k]].SiparisID;
			KBest[k].SipMiktar=S[secilen[k]].Miktar;
			KBest[k].KombineFiresi=B[0].BobinEni-topsen;
		}
	}
	else
	{
		if((double)KBest[0].KombineFiresi/KBest[0].BobinEni >=
(double)(B[0].BobinEni-topsen) / B[0].BobinEni
			)
		{
			KombineAdedi=1;
			KBest=(Kombine *) malloc(sizeof(Kombine)*(yer+1));
			for(k=0;k<yer+1;k++)
			{
				KBest[k].BobinEni=B[0].BobinEni;
				KBest[k].KombineDerecesi=yer+1;
				KBest[k].KombineNo = KombineAdedi;
				KBest[k].BobinEni=B[0].BobinEni;
```

```
                                        KBest[k].SiparisID=S[secilen[k]].SiparisID;
                                        KBest[k].SipMiktar=S[secilen[k]].Miktar;
                                        KBest[k].KombineFiresi=B[0].BobinEni-topsen;
                                    }
                                }
                            }
                        }
                        //---

                        KombYap(yer+1,i);
                }
        }

void sifirla(int yer)
{
        int j;
        for(j=yer+1;j<MaxKombine;j++)
        {
                secilen[j]=-1;
        }
}
```

ÖZGEÇMİŞ

Doğum tarihi	05.08.1982	
Doğum yeri	Kahramanmaraş	
Lise	1997 – 2000	Özel Tercüman Lisesi
Lisans	2000 – 2005	Yıldız Üniversitesi Kimya - Metalürji Fak. Matematik Mühendisliği Bölümü
Yüksek Lisans	2005 – 2007	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Matematik Mühendisliği Anabilim Dalı

Çalıştığı kurumlar

2005 – 2006	TET Yazılım Ltd. Şti.
2006 – devam	Doğuş Üniversitesi Bilgisayar Mühendisliği Araştırma Görevlisi