EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(YÜKSEK LİSANS TEZİ)

KAYNAK KULLANARAK BİR BOYUTLU KESME PROBLEMİ ÜZERİNE

Ecem Esma AKBAŞ

Tez Danışmanı: Prof. Dr. Urfat NURİYEV

Matematik Anabilim Dalı

Sunuş Tarihi: 16.08.2017

Bornova-İZMİR

2017

Ecem Esma AKBAŞ tarafından **Yüksek Lisans Tezi** olarak sunulan **"Kaynak Kullanarak Bir Boyutlu Kesme Problemi Üzerine"** başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **16.08.2017** tarihinde yapılan tez savunma sınavında aday oybirliği/~~oyçokluğu~~ ile başarılı bulunmuştur.

Jüri Üyeleri: İmza

Jüri Başkanı	**: Prof. Dr. Urfat NURİYEV**	.....
Raportör Üye	**: Yrd. Doç. Dr. Arif GÜRSOY**	.....
Üye	**:Yrd. Doç. Dr. Refet POLAT**	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Yüksek Lisans Tezi olarak sunduğum "**Kaynak Kullanarak Bir Boyutlu Kesme Problemi Üzerine**" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

16/08/2017

Ecem Esma AKBAŞ

ÖZET

KAYNAK KULLANARAK BİR BOYUTLU KESME PROBLEMİ ÜZERİNE

AKBAŞ, Ecem Esma

Yüksek Lisans Tezi, Matematik Anabilim Dalı

Tez Danışmanı: Prof. Dr. Urfat NURİYEV

Ağustos 2017, 57 sayfa

Günümüzde çelik, ahşap, plastik ve kağıt gibi birçok endüstri alanında karşılaşılan bir boyutlu kesme problemi, yöneylem araştırmasında sık karşılaşılan problemlerden biridir. Klasik bir boyutlu stok kesme probleminin (SKP) asıl amacı, müşterilerin siparişleri oluşturulurken yürütülen kesme işlemlerinden oluşacak fireyi minimum yapmaktır. Fire miktarı minimizasyonunu bir adım öteye taşımak amaçlı, birleştirme fonksiyonu hesaplamaları gündeme gelmiştir. Örneğin, çelik endüstrisinde yer alan bazı şirketlerin amacı; kesme işlemi sırasında müşterilerin isteklerini daha küçük parçalara ayırdıktan sonra kaynak işlemi ile bu parçaları tekrar bir araya getirerek fire miktarının azaltmaktır. Başka bir örnek olarak kağıt endüstrisinde yer alan kesim işlemlerinde yine birleştirme yapılarak kayıp daha da azaltılmaktadır. Fakat kaynak ve birleştirme işlemleri de şirketler için ayrıca bir maliyet oluşturduğu için, klasik stok kesme problemi, hem fire miktarını hem de kaynak veya birleştirme sayısı minimizasyonunu sağlayan çift amaç fonksiyonlu yeni bir bir boyutlu stok kesme problemine dönüşmüştür. Bu tez çalışmasında literatür taraması ile stok kesme problemi incelenmiştir. Ayrıca günümüzde karşılaşılan büyük boyutlu gerçek hayat problemlerinin efektif çözümleri için dinamik programlamaya dayalı sezgisel algoritmalar değerlendirilmiştir.

Anahtar kelimeler: Bir boyutlu kesme stok problemi, sezgisel dinamik programlama, doğrusal olmayan tam sayılı programlama

ABSTRACT

ON THE ONE-DIMENSIONAL CUTTING STOCK PROBLEM USING WELD

AKBAŞ, Ecem Esma

MSc in Department of Mathematics.

Supervisor: Prof. Dr. Urfat NURIYEV

August 2017, 57 pages

Recently, the One-Dimensional Cutting-Stock Problem (SKP) which is encountered in many industrial areas such as steel, wood, plastic and paper industries, is a common problem in operational researches. Main purpose of the classical One-Dimensional-Cutting-Stock Problem is to minimize the loss originating from the cutting processes when order of the clients are provided. To further minimize the loss amount, skiving option calculations have been came into question, recently. For example, some of the companies in steel industry aim to lower the loss with splitting the orders of the clients into smaller pieces, then welding these pieces back together. In an another example, in paper industry, skiving option can reduce the loss. However, because welding and skiving create an extra cost, classical cutting stock problem has evolved to a new One-Dimensional Cutting Stock Problem with Two Objective Function, which grants the minimization of both the amount of loss and welding or skiving counts. In this thesis, cutting stock problem was investigated with literature review. Besides, heuristic algorithms, which are based on dynamical programming, were evaluated on recent real-life, large-sized problems.

Keywords: One-Dimensional Cutting-Stock Problem, Heuristic Dynamical Programming, non-linear integer programming

TEŞEKKÜR

Yüksek lisans çalışmam boyunca, bu tezi hazırlamam için benden bilgisini ve anlayışını hiçbir zaman esirgemeyen değerli hocam Prof. Dr. Urfat NURİYEV'e, teşekkürü kendime bir borç bilirim.

Hayatım boyunca her türlü desteği eksik etmeyen, beni her zaman anlayışla karşılayan aileme, eşim Mehmet Emre Yeğin'e ve tüm sevdiklerime çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

1. GİRİŞ

Günümüzde işletmeler yoğun rekabet sonucu ayakta kalabilmek için maliyetlerini minimize etmek ve etkinliklerini artırmak durumundalardır. Kâğıt, çelik, metal, tekstil ve deri benzeri birçok endüstri alanlarında, boyutları belli olan bir malzemeden, çeşitli şekil, boyut ve miktarlara sahip daha küçük parçaların kullanılması gerekmektedir. Bu küçük parçaları oluşturmak kesim işlemi ile gerçekleşmektedir ve kesim planları düzenlenirken amaç; üretim maliyeti veya fire miktarı gibi bazı amaç fonksiyonlarının minimize edilmesidir. Bu problemlere, genel olarak, stok kesme problemleri (Cutting Stock Problems) denir.

Stok kesme problemi her sektörde farklılık gösteren bir problemdir. Teorikte problem oldukça basit görünse de, probleme işletmelerin istekleri ve gereklilikleri doğrultusunda birden çok kısıt eklenebilir. Stok kesme problemleri NP-zor olarak nitelendirilen bir problem çeşididir. Bu nedenle, bu alanda geliştirilecek olan sezgisel algoritmalar büyük önem taşımaktadır.

Demir-çelik, kağıt, deri gibi vb. gibi pek çok endüstride karşılaşılan bu bir boyutlu stok kesme problemlerinin çözüm yöntemleri halen araştırılmaktadır. Problemlerde, her bir hammadde, talebe göre küçük parçalara bölünebilir veya tekrar birleştirilebilmektedir. Bu uygulama için değişik kesim ve birleştirme fonksiyonlarını gözeten birleşik problemler oluşturulmuştur. İşletmeler; kesim planı oluştururken hem fire miktarının hem de kesim sırasında birleştirme kaybının en küçüklenmesini talep etmektedir. Bu tezde, sözü edilen birleşik problemler ele alınmış ve bu problemler için üretilen sezgisel algoritmalar derlenmiştir. Çelik ve kağıt endüstrisindeki sezgisel algoritmanın efektifliği de, gerçek hayat örnekleri üzerinde test edilmiştir.

Tez, 12 bölüm ve kaynaklar listesinden oluşmaktadır.

Tezin aşağıdaki bölümlerinde yukarıda bahsedilen problemlere ve çözüm yöntemlerine değinilmiştir.

Giriş kısmında, tezin içeriği hakkında kısa özet verilmiştir.

Tezin ikinci bölümde, modelleme hakkında bilgi verilmiştir.

Üçüncü bölümde, problemlerin karmaşıklık sınıfları ve hesaplama teorisi verilmiştir.

Dördüncü bölümde, doğrusal programlamadan bahsedilmiştir.

Beşinci bölümde, tamsayı problemlerine kısaca yer verilmiştir.

Altıncı bölümde, sırt çantası problemlerinden bahsedilmiştir.

Yedinci bölümde, alt küme toplam problemi ve çeşitlerine yer verilmiştir.

Sekizinci bölümde, bidon paketleme probleminin tanımı ve algoritması verilmiştir.

Dokuzuncu bölümde, tam sayı problemlerinin çözmek için kullanılan yöntemler verilmiştir.

Onuncu bölümde, stok kesme probleminin tanımına ve stok kesme problemi ile ilgili çalışmalara yer verilmiştir.

Onbirinci bölümde, Stok kesme problemi ile ilgili gerçek hayat problemlerine örnek olarak kağıt ve çelik endüstrisindeki uygulamalara yer verilmiştir ve bu problemlerin modellemeleri ve çözümleri gösterilmiştir.

Onikinci bölümde, ise, **sonuçlar** özetlenmiştir

2. MODELLEME

Günümüzde "sistem analizi" önemli bir konu olmasına karşın bu analizin yapılabilmesi için gerekli matematiksel araçlar az miktardadır. Yüzlerce bilinmeyenli denklem sistemini çözmek, bilgisayarlar aracılığı ile oldukça kolay hale gelmiştir. Fakat daha büyük sistemlerin analizinde problemler daha da zorlaşmaktadır. Bütün bu zor durumlara çözüm olabilecek matematiksel modelleme teknikleri kısa zamanda geniş alanlarda işe yaradığını kanıtlamış ve ilerisi için de umut verici bir araç olmuştur.

Bu bölümde model, modelin kurulması, model çözüm teknikleri ve matematiksel modelleme ile ilgili bilgiler verilmiştir (Kara, 1991; Taha, 2000).

2.1 Model Teriminin Anlamı

Model gerçek hayat problemlerinin fonksiyon, eşitsizlik vb. kavramlarla ifade edilmesine denir. Ele alınan sistem gerçek veya hayali olabilir. Her iki türde de sistemde optimize edilmesi gereken bir *amaç* söz konusudur. Bu amaç doğrultusunda sistemin bileşenleri arasındaki ilişkiler de göz önüne alınmalıdır. Bir model genellikle ele alınan sistemin yapısını tahmin etme, öğrenme, planlama, programlama ve denetleme amacıyla kullanılabilir. Var olan bir sistemin bulunduğu ortamda amaç doğrultusunda optimum bir şekilde sürdürülebilir olması veya hayali bir sistemin sorunsuzca gerçekleştirilebilmesi için bir model kullanılır. Bu model üzerinde sistem kusursuz bir şekilde gerçekleştirilmeye çalışılır. Model kurulduktan sonra bunun çözümünde analitik, sayısal, grafik ve tesadüflik gibi matematiksel ve istatistiksel yöntemler kullanılabilir.

Sanayi, tıp, mühendislik vb. farklı alanlarda kullanılan modeller, problemin incelenmesini kolaylaştırmasının yanında maliyeti de oldukça azaltmaktadır. Genellikle modeller üç gruba ayrılır: Taklit (iconic) modeller, Benzer (Analog) modeller, Simgesel modeller.

Taklit modeller: Bu tür modellerde bir sistemin bazı yönleri açık şekilde taklit edilir. Bu tür modeller üzerinde değişiklik yaparak sistemin incelenmesi mümkün değildir.

Benzer modeller: Bu tür modeller sistemin bazı özellikleri ve bazı elemanları, başka bir özellik ve eleman kümesine dönüştürülerek elde edilirler. Akış şemaları benzer modellere örnektir.

Simgesel modeller: Sistemin bir bütün olarak ele alınıp, elemanlarının ve bunlar arasındaki ilişkilerin belirli simgelerle gösterildiği modellere denir. Bu modeller eşitsizlik gibi matematiksel simgeler kullanır ve bu simgelere farklı değerler verilerek çeşitli değişiklikler, modelde bir değişiklik yapılmaksızın, incelenebilir. Buna genel olarak *duyarlılık analizi* denir.

2.2 Model Çözüm Teknikleri

Modeller çeşitli şekillerde kurulabilir. Tekrarlı ve basit problemlerde model, karar verici tarafından içgüdüsel şekilde oluşturulur. Kullanılacak değişkenlerin tanımlanması ve aralarında ilişki kurulması değişkenlerin yapısına bağlıdır. Değişkenler ölçülebiliyorsa, özellikle kantitatif olarak temsil edilebiliyorlarsa modelin matematiksel ifadelerle gösterilmesi için geçerli sebepler vardır. Araştıran kişi hangi değişkenleri seçip hangilerini ihmal ettiğini açık bir şekilde belirtmeli, seçtiği değişkenler arasındaki ilişkiyi açıkça ifade etmelidir.

Matematiksel sembollerle ifade edilen bir model bütün durumları içerirse karmaşık bir hal alır. Modele eklenecek her yeni değişken ya da karakter her zaman yeteri kadar avantaj sağlamayabilir. Belirli değişkenlerle elde edilen model ile gerçek sistem karşılaştırıldığında bir sapma gözlenebilir. Bu sapma daha az öneme sahip değişkenlerin modele eklenmemesinden kaynaklanır. Bu durumda kararlardaki hatalar iki sebepten oluşur:

a) Mantık hataları ve yanlış yorumlama.

b) Yanlış veya yetersiz değişken seçimi

Daha geçerli bir modele sahip olmak için modelin işleyişi sırasında meydana gelen sakıncaların ortadan kaldırılması, *model kurma* ve *bilgi derleme* süreçlerinin dengelenmesi ile daha geçerli bir modele sahip olunabilir.

Kantitatif karar verme tekniklerinin uygulanabilmesi için model kurarken aşağıdakiler göz önünde bulundurulmalıdır.

1) Problemin kantitatif ölçüler doğrultusunda formüle edilmesi,

2) Ele alınan sistemin gerçek yapıdan soyutlanması ya da bir modelin kurulması,

3) Bilgiyi derleme ve modeli kurma aşamasında geri besleme yapılması,

4) Modelin, sistemin yapısal durumunu yorumlamaya ve değerlendirmeye tam olarak vermesi

Kantitatif olarak ifade edilen model *analitik, sayısal* ve *simülasyon* çözüm teknikleriyle çözülebilir.

2.3 Matematiksel Modelleme

Bir sistemin bileşenlerinin simgelerle tanımlanıp, bunlar arasındaki ilişkilerin fonksiyonlarla gösterimine *matematiksel model* denir (Kara, 1991). Bir problemin matematiksel model ile ifade edilmesi onun çözümünü elde etme yolunda atılan en önemli adımlardandır.

Bir optimizasyon probleminin matematiksel modellenmesi yapılırken tanımlanması gereken üç ana eleman vardır:

- Karar değişkenleri
- Kısıtlar
- Amaç fonksiyonu

Matematiksel karar modeli geliştirilirken göz önüne alınması gereken parametreler ile sistemin performansına etki eden durumlar belirlenir. Bu parametreler sistemi tasarlayan kişinin kontrolündedir. Amacımız ise bunlara en uygun değerlerin verilmesidir. Bu parametrelere *karar değişkeni* denir.

Amaç üzerindeki karar değişkenlerinin etkilerinin analitik olarak gösterilmesi ile *amaç fonksiyonu* oluşur.

Modeldeki kısıtlar ise, sistemin içinde bulunduğu koşulların matematiksel semboller ile ifade edilmesidir. Bunlar *arz kısıtları*, *talep kısıtları* vb. unsurlar olabilir.

Eğer karar değişkenleri üzerinde hiçbir sınırlama yoksa ve sadece bir amaç fonksiyonunun en iyilenmesi isteniyorsa bu durumda *kısıtsız modeller* meydana gelir. Gerçek hayatta karşılaştığımız modellerin bazıları bu yapıdadır. En az bir sınırlama olması kısıtlı modelleri meydana getirir. Ele alınan problemin bir tek dönem (haftalık, aylık, yıllık vb.) için çözümü söz konusu ise *statik model*, birden fazla dönem için çözümü söz konusu ise *dinamik model* denir. Problemin amaç fonksiyonu birden fazla ise *çok amaç fonksiyonlu* problem olarak ele alınır. Karar değişkenlerinin tamamı pozitif reel değerler alıyor ise *sürekli optimizasyon problemi*, tümü tam sayı değerler aldığında ise *kesikli optimizasyon problemi* olarak ele alınır. Karar değişkenlerinden bir kısmı reel, bir kısmı tamsayı değer alırsa *karma kesikli optimizasyon problemi* olur. Kombinatoryal bir seçme durumu söz konusuysa *kombinatoryal optimizasyon problemi* olarak ele alınırlar.

3. PROBLEMLERİN KARMAŞIKLIK SINIFLARI

Bir problemin çözümüne yönelik bir algoritma araştırılmadan önce bu problemin sonlu sayıda aşamada çözülüp çözülemeyeceğinin bilinmesi gereklidir. Algoritmalar teorisine göre evrensel algoritmik modellerin üç çeşidi ele alınmıştır. İlk olarak, algoritma kavramı hesaplama ve nümerik fonksiyonlar gibi matematiksel kavramlarla ilişkilendirilmektedir. Algoritma kavramını ilk olarak biçimlendiren *özyinelemeli fonksiyonlar (recursive function)* bu türün en gelişmiş modelidir. İkincisi, algoritmanın her ayrık zamanda çok basit işlemleri yapan bir deterministik makine ile bağdaştırılması ile oluşur ve bu türün temel teorik modeli 1930'larda oluşturulan *Turing makineleridir* (Turing machine). Bu modeller, yapısal olarak bilgisayara en yakın modellerdir. Üçüncüsü ise, herhangi bir alfabede sözcüklerin değiştirilmesine dayalı kelime işlemcilerdir (Nabiyev, 2009).

Kullanılan algoritma tipini esas alındığında tüm problemler aynı değildir. bazılarının problemlerin çözümü, sezgisel çözümler dikkate alınmazsa diğerlerinden daha hızlı olabilir. Algoritmaların yürütülmesi için gerekli işlemlerin sayısı temel alınarak bir problemin algoritmik çözümlerinin sınıflandırılması yapılabilir ve buna *algoritmik karmaşıklık* denir. Karmaşıklık değerlendirilmesi algoritmanın çalıştırılacağı bilgisayarın işlemcisinden bağımsız olarak yapılmalıdır (Nabiyev, 2009).

Hesaplama karmaşıklığı, bir algoritmanın ne kadar hızlı çalışacağı ve bilgisayarın belleğinde ne kadar yer kullanacağı konusunda bilgiler verir. Algoritmanın girdi ya da problem boyutu, girişi tanımlamak için gerekli sembollerin sayısı ile ilişkilidir. Bir algoritmanın hesaplama karmaşıklığı, iki açıdan incelenmektedir; birincisi, hesaplamayı yapmak için gerekli zamanı ölçen zamansal karmaşıklık değerlendirilmesidir. İkincisi ise hesaplama içinde gerekli bellek alanının ölçümü olarak adlandırılan yersel karmaşıklıktır. Zaman karmaşıklığı genelde sadece karmaşıklık olarak adlandırılmaktadır (Nabiyev, 2009).

Algoritmaların çalışma zamanlarını belirlenirken sabit çalışma zamanlı emirler dikkate alınmaz. Bir algoritmanın sergileyeceği performans ile çalışma

zamanının matematiksel özdeşliğinin yazılması orantısal olarak tahmin edilebilir. Örneğin, lineer-logaritmik ($n \cdot \log n$) çalışma zamanlı bir sıralama algoritması süper bilgisayarda da mikro bilgisayarda da orantısal olarak aynı çalışır. *Büyük O* (Big O) notasyonu, çalışma zamanının üst sınırıdır ve en kötü durumdaki çalışma zamanının belirlenmesinde kullanılır. Ayrıca büyük O notasyonu bir fonksiyonun asimptotik üst sınırını belirtir (Nabiyev, 2009).

n doğal sayısı ve bu doğal sayıyı herhangi bir pozitif değere atayan *f* ve *g* fonksiyonları olsun.

$\exists n_0, c > 0$ ve $\forall n \geq n_0$ için $0 \leq f(x) \leq c \cdot g(x)$ ise $f(x) = O(g(x))$ dir.

Örneğin, $n^2 + 6n + 5$ fonksiyonunda *n*'in büyük değerleri için n^2'nin dışındaki toplananlar önemsenmez ve algoritma karmaşıklığı $c = 1$ olmak üzere $c \cdot O(n^2)$, yani $O(n^2)$ şeklinde yazılır. Algoritma karmaşıklığı fonksiyonlarına örnek vermek gerekirse, ikili arama için $O(\log n)$, doğrusal arama için $O(n)$, hızlı sıralama algoritması için $O(n \cdot \log n)$ fonksiyonları verilebilir. Bu problemlerin hepsinde $g(x)$ fonksiyonu polinomiyal karakter taşımaktadır (Nabiyev, 2009; Cormen et al., 2001).

3.1 P, NP sınıflar, NP-tamlık ve NP-zorluk

Bir algoritmanın zaman karmaşıklığı, belli boyutlardaki girdi değerleri için harcanan basit hesapsal adımların sayısını veren bir fonksiyondur. *n* girdi verisinin boyutu olmak üzere, bir algoritmanın çalışma zamanı üstten herhangi bir $P(n)$ polinomu ile sınırlı ise buna *polinom sınırlı algoritma* denir ve bu algoritmalarla çözülebilen tüm problemler *P sınıfı* olarak tanımlanır. Deterministik makineler, algoritmik karakter taşıyan P sınıfındaki problemleri çözebilmektedir (Nabiyev, 2005).

Bir problemin çözümü için hazır formüller ve çözüme ilişkin özyinelemeli ifade bilinmediği zaman, çözüm için tarama ve sezgisel seçimlerin yapılması gerekir. Bir problemin karmaşıklığı polinomiyal biçimde tanımlanabilirse, bu

problem için sonlu sayıda adımla polinomiyal zamanda çözülebileceği söylenebilir. D*eterministik Turing makineleri* (DTM), polinomiyal algoritmaları çalıştıran sanal modellerdir ve P sınıfı, DTM ile polinomiyal zamanda çözülebilen problemler sınıfıdır. Polinomiyal zamanda deterministik olmayan Turing makineleriyle (NDTM) çözülebilen problemlere ise *NP* (Non-deterministic Polinomial) problemler denir (Nabiyev, 2009; Garey and Johnson, 1979).

NDTM, DTM ile çözülebilen tüm problemleri çözebilir olduğundan P sınıf problemlerine NP sınıfın bir alt kümesi olarak bakılabilir, $P \subset NP$ (Cormen et al., 2001).

P sınıfı polinomiyal zamanda çözülebilen problemlerden oluşur ve bu problemler, k bir sabit ve n problem girdisinin boyutu olmak üzere $O(n^k)$ zamanda çözülebilen problemlerdir. NP sınıfı, bir problemin herhangi bir çözüm sertifikası verildiğinde, problemin girdi boyutuna bağlı olarak bu sertifikanın polinomiyal zamanda doğrulanabildiği problemler sınıfıdır. Örneğin, Hamiltonian Devresi probleminde, V tepeler, E ayrıtlar kümesi olmak üzere yönlü bir $G=(V,E)$ grafı, $\langle v_1, v_2, ..., v_{|V|} \rangle$ şeklinde bir çözüm sertifikası verilsin. Kolayca, polinomiyal zamanda, $i=1,2,...,|V|-1$ için $(v_i, v_{i+1}) \in E$ ve $(v_{|V|}, v_1) \in E$ konrolü yapılabilir. Yani, $n=|V|$ olmak üzere $O(n)$ zamanda kontrol gerçekleştirilir. Bu tür problemleri polinomiyal zamanda doğrulayabilen algoritmalara deterministik olmayan algoritma denir.

Bir Π probleminin *NP-tam* sınıftan olabilmesi için, öncelikle bu Π probleminin NP sınıftan olduğu ve sonrasında NP sınıftaki her problemin polinomiyal zamanda bu probleme indirgenebileceği gösterilmelidir (Cormen et al., 2001):

1. $\Pi \in NP$ ve

2. $\forall \Pi' \in NP$ için $\Pi' \leq_P \Pi$.

Bu adımlardan ilki değil sadece ikincisi sağlanırsa Π problemi *NP-zor* sınıftandır denir.

Bir Π probleminin NP-tamlığının ispatı için aşağıdaki adımlar izlenmelidir:

1. Π 'nin NP sınıfından olduğu gösterilir,

2. Bilinen bir Π' NP-tam problemi seçilir,

3. Π' probleminden Π problemine bir *f* dönüşümü oluşturulur ve

4. *f* dönüşümünün bir polinomiyal dönüşüm olduğu ispatlanır.

Bu ispat yönteminin dışında en çok kullanılan ve en basit yöntem kısıtlama yöntemiyle ispattır. Bu ispat yönteminde, NP sınıftan olan ve NP-tam olduğu ispatlanmak istenen Π probleminin, bilinen bir Π' NP-tam probleminin özel bir durumu olduğunun gösterilmesi yeterli olur. Yani, $\Pi \in NP-tam$ olması için aşağıdaki iki durumun sağlanması yeterlidir (Garey and Johnson, 1979):

1. $\exists \Pi' \subset \Pi$

2. $\Pi' \in NP-tam$

4. DOĞRUSAL PROGRAMLAMA

Çalışmamızda kullanılacak olan tamsayılı programlama problemleri, doğrusal programlama problemlerinin özel bir durumu olması nedeniyle, bu bölümde doğrusal programlamanın gelişimi, kullanım alanları ve doğrusal programlama problemlerinin modellenmesi ile çözüm yöntemlerine yer verilecektir.

4.1 Doğrusal Programlamanın Gelişimi ve Kullanım Alanları

Doğrusal programlama bir matematiksel model olarak ortaya çıkmıştır ve 2. Dünya Savaşı sırasında, maliyet ve gelir planlaması yapılırken, ordunun masrafını azaltmak, aynı zamanda düşmana verilen zararın artırımı gibi konularda kullanılarak gelişmiştir.

1947 yılında yayınlanan *simpleks algoritması* ile George Dantzig, dualite teorisini geliştiren John von Neumann ve Dantzig'den önce benzer yöntemleri ekonomi alanında kullanan Rus Matematikçi Leonid Kantorovich doğrusal programlamanın öncüleri olarak bilinirler.

Doğrusal programlama çeşitli nedenlerden dolayı optimizasyonun önemli bir alanıdır. Doğrusal programlama problemi yöneylem araştırmasında birçok problem ifade etmek için kullanılabilir. Benzer olarak, ekonomi ve iş yönetiminde, eğitim, petrol işletmeleri, gıda sektörü gibi çeşitli alanlarda geliri en büyüklemek ya da maliyeti en küçüklemek vb. amaçlarla çok fazla kullanılır.

4.2 Doğrusal Programlama Modeli ve Çözüm Yöntemleri

Bir sistemin bileşenlerinin simgelerle tanımlanıp, bunlar arasındaki ilişkilerin fonksiyonlarla gösterimi **matematiksel model** olarak adlandırılır. Sistemin yöneticisinin kontrolünde olan ve **karar değişkeni** olarak isimlendirilen değişkenlere, hangi değerlerin verilmesi gerektiğini belirlemek amacıyla kullanılan matematiksel modellere de **karar modeli** denir. Karar vericinin kontrolü olmadan değer alan bileşenlere parametre; modelde karar değişkenleri ya da karar değişkenleri ile parametreler arasındaki mecburi ilişkilerin her birine ise

kısıt denir (Kara, 1991). Karar verici tarafından en büyüklenen ya da en küçüklenen karar değişkenlerinin fonksiyonuna **amaç fonksiyonu** denir (Winston, 1991).

Bir doğrusal programlama modeli, doğrusal bir amaç fonksiyonu ve tümü doğrusal olarak ifade edilmiş kısıtlardan oluşur. Örneğin m tane kısıt, n tane karar değişkeninden oluşan bir doğrusal programlama modeli şu şekilde ifade edilir:

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

$$x_j \geq 0, \qquad j = 1,...,n$$

kısıtları altında (k.a)

$$enb\ Z^* = \sum_{j=1}^{n} p_j x_j$$

Burada;

x_j : j. karar değişkenine atanacak değer,

p_j : Bir birim j. karar değişkeninin amaç fonksiyonuna katkısı (gelir, kar, maliyet vb.),

b_i : i. kaynak miktarı,

a_{ij} : Bir birim x_j için gerekli i. kaynak veya girdi,

olarak tanımlıdır.

Bir doğrusal programlama modelinin çözüm yaklaşımları,

- Grafik çözüm,
- Analitik çözüm,
- Ardışık sayısal çözüm

başlıkları altında toplanabilir (Kara, 1991).

Bu yöntemlere kısaca değinmek gerekirse; grafik çözüm en fazla üç karar değişkeni olduğunda uygulanabilmektedir. Gerçek hayatta karşılaşılan problemlerin çok sayıda karar değişkeni içeriyor olması grafik çözümün ancak kavramların anlaşılabilirliği için kullanılmasına neden olmaktadır. Analitik çözüm ise en iyi çözümün varlığının biliniyor olması halinde kullanılabilir, fakat genelde bu durum mümkün değildir. Bunun yanı sıra değişken ve kısıt sayısı arttıkça oluşan denklem sisteminin boyutu yani dolayısıyla işlem yoğunluğu artar.

Bu iki yöntemin gerektirdiği ihtiyaçlar doğrultusunda yapılan çalışmalar ardışık çözüm yöntemlerini geliştirmiştir. Bunlardan ilki ve en yaygını "Simpleks Algoritması" olup "Karmarkar Algoritması" da en çok bilinenler arasındadır (Kara, 1991).

5. TAMSAYILI PROGRAMLAMA

Bu bölümde çalışmamızla ilgili olan tamsayılı programlama probleminin genel tanım ve türlerine yer verilecektir.

5.1 Tamsayılı Programlamanın Genel Tanım ve Türleri

Bir problemde bütün değişkenlerin tamsayı olması isteniyorsa bu problem *saf tamsayılı programlama* (TP) problemi olarak adlandırılır.

Tamsayılı programlama, doğrusal programlamanın bir uzantısıdır ve amacı; doğrusal programlamada meydana gelebilecek gerçekçi olmayan sonuçları yok etmektir. Çünkü bazı doğrusal programlama modellerinde sonuçlar her zaman tamsayı çıkmamakta ve bu durum problemin gerçek hayattaki problemlere uygunluğunu bozmaktadır. Örneğin; bir üretim probleminde sonuçların 1.2 masa, 3.6 bardak gibi kesirli çıkması gerçekçi olmamaktadır. Bu sonuçların tamsayıya yuvarlatılması akla gelebilecek ilk yöntem olabilir; ancak böyle bir yol takip etmek çözümü optimallikten oldukça uzaklaştırabilir. Tamsayılı programlama tekniği, kısıtları bozmadan sonucun tamsayı olmasını sağlamaktadır (Bakır ve Altunkaynak, 2003).

Tamsayı değerli değişkenler içeren modellerin çoğu büyük ölçekli planlama modelleridir. Örneğin üretim problemleri, gezgin satıcı problemleri, sırt çantası, vb. problemler tamsayılı programlamaya göre modellenir.

Tamsayılı programlama problemlerinin formülasyonu, sürekli değişkenli matematiksel modellerin formülasyonuyla benzerlik göstermesine rağmen, bazı kısıtlamalar hesaplama bakımından tamsayılı problemleri daha zor bir hale getirmektedir. Genellikle tamsayılı problemler NP-zor sınıfına aittir; bu nedenle genel doğrusal modeller en kötü durumda etkin olarak çözülebilirken, tamsayılı programlama problemleri üstel hesaplama zamanı gerektirebilir.

6. SIRT ÇANTASI PROBLEMLERİ

Sırt çantası problemleri (KP) tamsayılı programlama problemleri içinde en kolay olan türdür. Bunun nedeniyse problemin tanımının anlaşılabilir olması ve gerçek hayatta karşılaşılan pek çok durumun bu probleme bağlı olarak kolayca ifade edilebilmesidir.

Son yıllarda sırt çantası problemleri çok fazla çalışılmaktadır. Bunun nedeni teorisyenler açısından problemin yapısı basit olup uygulama alanındakiler için ise birçok endüstriyel duruma uygulanabilmesidir. Örneğin sermaye bütçeleme, kargo yükleme, kesme gibi problemler KP'nin en klasik uygulamalarındandır ve bu problem ile çözülebilirler.

Problem sadece tek bir kısıt ve sadece pozitif katsayılar ile en basit tamsayılı modeldir; ancak değişkenlerin tamsayı olma şartının eklenmesiyle sırt çantası problemi zor problemler sınıfına yerleşir.

Sırt çantası problemi, bir maksimizasyon problemi için en basit örnek olması nedeniyle yüz yıllardır çalışılmaktadır. 1897 yılında G.B. Mathews çeşitli kısıtların nasıl tek bir sırt çantası kısıtında toplanabildiğini göstermiştir. Bu, genel bir tamsayılı problemin sırt çantası problemine indirgemesinin ilk örneklerinden biridir ve dolayısıyla sırt çantası problemini çözmenin en az bir tamsayılı problemi çözmek kadar zor olduğunu ispatlar. Salkin ve De Kluyver (1975) tamsayılı doğrusal programların KP'ye dönüştürülmesinde çok sayıda endüstriyel uygulama ve sonuçlarını sunmuşlardır. Martello ve Toth (1979) 0-1 KP için kesin algoritmalar ve bunların ortalama performanslarını düşünmüşler sonrasında ise bu çalışmayı diğer KP türleri ve yaklaşık algoritmalar için genişletmişlerdir. (Kellerler et al., 2004)

6.1 Sırt Çantası Problemi Nedir?

Sırt çantası problemine küçük bir örnek vermek gerekirse; sırt çantasını hazırlayan bir dağcı ele alınabilir. Bu dağcı, bir tur için sırt çantasını hazırlamaktadır. Problemde, tur için kullanışlı olabilecek birçok nesne yanında,

kapasitesi sınırlı olan bir çanta mevcuttur. Bu nesnelerin her biri 1'den n'ye kadar numaralandırılmış ve her bir nesnenin sağladığı yarar $p_j > 0$ sayısıyla, ağırlıkları ise $a_j > 0$ sayısıyla ölçülmektedir. Bu kişi çantasını maksimum faydayla ve çanta kapasitesi olan 'c' yi aşmayacak şekilde doldurmak istemektedir. Hangi eşyaları almalıdır?

Yukarıda anlatılan örnek problem de KP çerçevesinde çözülen bir problemdir. Problemin formal tanımı ve TP formülasyonu ise aşağıda gösterildiği gibidir:

Yanımıza almak istediğimiz *n* adet nesne olsun ve bu nesneleri *j* ile gösterelim, $j = \{1, 2, ..., n\}$,

p_j : j. nesnenin sağladığı fayda,

a_j : j. nesnenin ağırlığı,

b : Sırt çantasının toplam kapasitesi

olsun. (Genellikle bütün değerler pozitif tamsayı olarak alınır.) Buradaki amaç, nesneler kümesinin bir alt kümesini seçmek öyle ki seçilen nesnelerin sağladığı toplam fayda maksimum olsun ve toplam ağırlık b kapasitesini geçmesin.

kısıtlar:

$$\sum_{j \in J} a_j x_j \leq b \qquad (6.1)$$

$$x_j \in \{0,1\}, \quad j \in J \qquad (6.2)$$

amaç fonksiyonu:

$$enb\ Z^* = \sum_{j \in J} p_j x_j \qquad (6.3)$$

Karar değişkenlerinin tanımı bakımından sırt çantası problemini (KP) genel olarak üç türde sınıflandırabiliriz:

1. 0/1 Sırt Çantası Problemleri

2. Negatif olmayan tamsayılı sırt çantası problemleri (Kısaca tamsayılı diyeceğiz.)

3. Sınırlı tamsayılı sırt çantası problemleri

Yukarıdaki formülasyon birinci tür sırt çantası problemidir. İkinci türdeki sırt çantası problemleri içinse (6.2) kısıtı yerine

$$x_j \geq 0 \text{ ve tamsayı}, \ j \in J \qquad (6.2)'$$

kısıtı kullanılır. (Bu problem *sınırsız tamsayılı sırt çantası problemi* olarak da bilinir.)

Verilen problemde çantaya yerleştirilecek nesnelerin sayıları sınırlandırılmış olabilir. Bu durumda problem sınırlı tamsayılı sırt çantası problemidir ve bu modelde (6.2) kısıtı,

$$0 \leq x_j \leq c_j \text{ ve tamsayı}, \ j \in J \qquad (6.2)''$$

kısıtıyla yer değiştirir ve dolayısıyla problem çözümünde her hangi bir x_j nesnesi en fazla c_j değeriyle yer alır.

Problem içerdiği kısıt sayısı bakımından ise iki türde sınıflandırılır:

1. Bir boyutlu sırt çantası problemleri

2. Çok boyutlu sırt çantası problemleri

Eğer sırt çantası probleminde sadece bir tane ana kısıt varsa bu tür problemler “Bir boyutlu sırt çantası problemi” olarak adlandırılır; yukarıdaki formülasyon ve tanım bu türdeki probleme örnektir. Bu tür problemlerin günümüzdeki yansımalarından birisi de, bidon paketleme problemleridir.

7. ALTKÜME TOPLAMI PROBLEMİ

7.1 Altküme Toplamı Problemine Giriş

Altküme toplamı problemi (ATP) aşağıdaki gibi tanımlanır (Andonov et al., 2000).

N doğal sayılar kümesinin bir $A=\{a_1,a_2,...,a_n\}$, $A\subset N$ *altkümesi ve* $b\in N$ *tamsayısı verilmiştir;* $\exists S,\ S\subseteq A,\ \sum_{s\in S} s=b$ *, yani A kümesinin elemanlarının toplamı b' ye eşit olan S altkümesinin olup-olmadığı sorulur.*

Göründüğü gibi, bu problem aşağıdaki *diophantine denklemi* ile ilişkilidir:

$$\left\{\sum_{i=1}^{n} a_i x_i = b,\ x_i \in \{0,1\},\ i=\overline{1,n}\right\}$$

Bilindiği gibi bu problem NP-tam sınıftandır. (Balev et al., 2001)

7.2 Boole Değişkenli Altküme Toplamı Problemi (BDATP)

Ağırlıkları $a_i,\ i=\overline{1,n}$ olan n adet eşya ve kapasitesi *b* olan çanta verilmiştir. Bu eşyaların içinden bir kısmını seçerek öyle bir altküme oluşturmalıyız ki, bunların ağırlıkları toplamı çantanın kapasitesinden küçük, aynı zamanda ona en yakın olsun. x_i değişkenlerini aşağıdaki gibi tanımlayarak

$$x_i=\begin{cases}1, & \text{eğer } i.\text{ eşya seçilirse;}\\ 0, & \text{aksi halde.}\end{cases}$$

BDATP'nin matematiksel modelini yazabiliriz (Bertsimas and Demir, 2002):

$$\max z = \sum_{i=1}^{n} a_i x_i \tag{7.1}$$

$$\sum_{i=1}^{n} a_i x_i \le b \tag{7.2}$$

$$x_i = \{0,1\} \qquad i = \overline{1,n} \tag{7.3}$$

BDATP aynı zamanda 0-1 sırt çantası probleminin özel bir durumu olmasından dolayı *Değer Bağımsız Sırt Çantası Problemi* olarak da adlandırılmaktadır. Buradaki özel durum; eşyaların değerlerinin, eşyaların ağırlıklarına eşit alınmasıdır, yani $c_i = a_i$, $i = \overline{1,n}$.

Açıktır ki, problemin anlamlı olabilmesi için aşağıdaki koşullar sağlanmalıdır:

$$\sum_{i=1}^{n} a_i x_i > b \tag{7.4}$$

$$a_i < b \qquad i = \overline{1,n} \tag{7.5}$$

BDATP'nin çözümü için bilinen dinamik programlama algoritmalarından biri aşağıdaki gibidir (Caprara et al., 2003):

$$z_0(y) = 0 \qquad (0 \le y \le b) \tag{7.6}$$

$$z_k(0) = 0 \qquad (0 \le k \le n) \tag{7.7}$$

$$y < 0 \quad \Rightarrow \quad z_k(y) = -\infty \tag{7.8}$$

$$k=1 \;\Rightarrow\; z_1(y)=\lfloor y/a \rfloor a \qquad (0 \le y \le b) \tag{7.9}$$

$$k>1 \;\Rightarrow\; z_k(y)=\max\{z_{k-1}(y)\,,\, z_k(y-a_k)+a_k\} \tag{7.10}$$

Yukarıdaki formüllerle oluşturulan tablodan yalnızca amaç fonksiyonunun en iyi değeri elde edilebilir, çözüm vektörünü bulmak için ikinci bir tablo gereklidir. O tabloyu oluşturmak için gereken formüller ise aşağıdaki gibidir:

$$i(1,y)=\begin{cases}0, & \text{eğer } z_1(y)=0\\ 1, & \text{eğer } z_1(y)\neq 0\end{cases} \tag{7.11}$$

$$i(k,y)=\begin{cases}i(k-1,y), & \text{eğer } z_{k-1}(y)\ge z_k(y-a_k)+a_k\\ \text{k} \quad , & \text{eğer } z_{k-1}(y)< z_k(y-a_k)+a_k\end{cases} \tag{7.12}$$

7.3 Değişkenleri Sınırlı Tamsayılı Altküme Toplamı Problemi (STATP)

Sınırlı tamsayılı ATP'de n adet farklı eşyanın her birinin sayısının 1'den büyük ve bir v_i sınırından küçük olduğu varsayılır. Bu probleme *Çubuk Kesme Problemi* de denir.

x_i değişkenlerini aşağıdaki gibi tanımlayarak

$$x_i=\begin{cases}\text{k}, & \text{eğer } i\text{. eşyadan k tane seçilirse;}\\ 0, & i\text{. eşyadan hiç seçilmezse.}\end{cases}$$

STATP 'nin matematiksel modelini yazabiliriz (Caprara et al., 2003):

$$S(x)=\max_x\{\sum_{i=1}^{n} a_i x_i \mid \sum_{i=1}^{n} a_i x_i \le b,\quad 0\le x_i \le v_i,\quad x_i \in N_0,\quad i=\overline{1,n}\} \tag{7.13}$$

Burada;

$$a_i, v_i \in N, \quad i = \overline{1,n}, \quad b \in N, \quad N = \{1,2,...,n,...\}, \quad N_0 = N \cup \{0\} \tag{7.14}$$

Özyinelemeli bağıntıları vermek için aşağıdaki gibi $S_k(y)$ fonksiyonunu tanımlayalım. $(y = \overline{0,b} \quad , \quad k = \overline{1,n})$

$$S_k(y) = \max_x \{ \sum_{i=1}^{k} a_i x_i \mid \sum_{i=1}^{k} a_i x_i \le y, \ 0 \le x_i \le v_i, \ x_i \in N_0, \ i = \overline{1,k} \} \tag{7.15}$$

Bu fonksiyonun başlangıç değerleri (yani $k = 1$ için) aşağıdaki gibi belirlenir. $(y = \overline{1,b})$

$$S_1(y) = \left\{ \begin{array}{lll} (1, y/a_1) & , \ eğer & y/a_1 = \lfloor y/a_1 \rfloor \\ (1,0) & , \ aksi\ halde & \\ (1,0) & , \ eğer & v_1 a_1 < y \le b \end{array} \right\} \quad eğer \quad 1 \le y \le v_1 a_1 \tag{7.16}$$

$k > 1$ için özyinelemeli formül aşağıdaki gibi olur. $(k = \overline{2,n} \quad , \quad y = \overline{1,b})$

$$S_k(y) = \left\{ \begin{array}{lll} S_{k-1}(y) & , \ eğer & 1 \le y < a_k \\ (k,1) & , \ eğer & S_{k-1}(a_k) = (z,0) \\ S_{k-1}(y) & , \ eğer & S_{k-1}(a_k) \ne (z,0) \\ * & , \ eğer & a_k < y \le \sum_{j=1}^{k} a_j v_j \\ (k,0) & , \ eğer & \sum_{j=1}^{k} a_j v_j < y \le b \end{array} \right\} \quad , \quad eğer \quad y = a_k \tag{7.17}$$

Burada * durumu aşağıdaki gibi olur.

$$* = \begin{cases} (k,0) \quad , \quad eğer \quad S_k(y-a_k) = (z,0) \\ (k,p) = \begin{cases} p = 1 & , \quad eğer \quad S_k(y-a_k) \neq (k,z) \\ p = S_k(y-a_k)+1 & , \quad eğer \quad S_k(y-a_k) = (k,z) \\ p = 0 & , \quad eğer \quad p > v_k \end{cases} \end{cases} \tag{7.18}$$

8. BİDON PAKETLEME PROBLEMİ

Bidon paketleme problemi bir çok endüstriyel oluşumda kullanılmaktadır ve son zamanlarda kombinatoriyal optimizasyon alanında sık çalışılan konulardan bir tanesidir.

Belirli bir sayıda birbiriyle aynı bidon ve birbirinden farklı biçimlerde çok sayıda paket olduğunu varsayalım. Bidonlar içine, bütün paketleri dışarı taşırmadan koyabilmek için en az kaç bidona ihtiyaç olduğunu bulduğumuz probleme *Bidon Paketleme Problemi* denir. Ev yapımında kullanılan kalasların farklı uzunluklardaki parçalara biçilmesi, kargo yükleme, bütçe planlama vb. gibi pek çok uygulama alanı vardır. Bidon paketleme problemi, NP-tam sınıftandır (Kellerler et al., 2004). Bu problemi çözmek için birçok sezgisel algoritma bilinmektedir (Kellerler et al., 2004; Martello and Toth, 1990). Bu bölümde bidon paketleme problemini çözmek için dinamik programlamaya dayalı bir algoritma önerilmiştir.

Aşağıda problemin matematiksel modeli ve önerilmiş çözüm algoritması verilmiştir.

8.1 Problemin Matematiksel Modeli

Bidon paketleme problemi sırt çantası problemindeki terminoloji kullanılarak aşağıdaki gibi tanımlanabilir (Kellerler et al., 2004). Bize n adet paket ve n adet sırt çantası (veya bidon) verilmiş olsun,

w_j = j. paketin ağırlığı

c = her bir bidonun kapasitesi

Her eleman bir bidona atanmalı ve buradaki amaç; her bir bidondaki paketlerin toplam ağırlığı c' yi aşmayacak şekilde en az bidon kullanarak tüm paketleri yerleştirmektir. Bu amacı gerçekleyecek olası bir matematiksel model aşağıdaki gibidir;

Amaç fonksiyon:

$$\text{Enk} \quad z = \sum_{i=1}^{n} y_i$$

Kısıtlar:

$$\sum_{i=1}^{n} w_j x_{ij} \le c y_i, \quad i \in N$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j \in N$$

$$y_i = 0 \; veya \; 1, \quad i \in N$$

$$x_{ij} = 0 \; veya \; k, \quad i \in N, j \in N$$

Burada,

$$y_i = \begin{cases} 1, & eğer \; i. bidon \; kullanıldıysa; \\ 0, & aksi \; halde, \end{cases}$$

$$x_{ij} = \begin{cases} 1, & eğer \; j. paket \; i. bidona \; koyulduysa \\ 0, & aksi \; halde. \end{cases}$$

Burada,

w_j , c pozitif tamsayıdırlar ve $w_j \le c$ ($j \in N$). $N = \{1, 2, ..., n\}$

8.2 Problemin Çözüm Yöntemi ve Algoritması

Problemi çözmek için dinamik programlama temelli sezgisel bir algoritma geliştirilmiştir. Bu algoritmada her bir bidonu yüklemek için yeni bir dinamik programlama algoritması uygulanmaktadır. BP algoritması aşağıdaki gibidir (Berberler et al., 2011):

BP1) $WV = \{(w_1 v_1), (w_2 v_2), ..., (w_n v_n)\}$ ve c girilir.

BP2) $BS = 0$

BP3) Eğer $\sum_{j=1}^{n} w_j v_j \le c \quad \Rightarrow$ BP10'a git.

BP4) Yukarıdaki giriş verileri göz önüne alınarak bir bidonu paketlemek için dinamik programlama algoritması kullanılarak kesin çözüm bulunur. Bu çözümü şöyle gösterelim: $(k_1, k_2, ..., k_n)$. Burada k_i, a_i eşyasından kutuya koyulanların sayısını bildirir.

BP5) $\max_{i=1,n} \left\{ \frac{k_i}{v_i} \right\} = \frac{k_p}{v_p}$ belirlenir.

BP6) $v_p = k_p - 1$ varsayıp problem bir bidon için yeniden çözülür ve bu süreç bidonun tam paketlenebileceği tüm durumlar için tekrar edilir. Bu çözümlerden büyük boyutlu eşyaların daha çok kullanıldığı çözümler tercih edilir. Varsayalım ki sonuçta aşağıdaki çözüm tercih edilmiş olsun:

$$(k_1^*, k_2^*, ..., k_n^*) \tag{8.1}$$

BP7) $\min_{i=1,n} \left\{ \frac{v_i}{k_i^*} \right\} = t$ bulunur ve t sayıda kutu (8.1) çözümüne uygun şekilde doldurulur.

BP8) $v_i = v_i - k_i^* t$

BP9) $BS = BS + t$, BP3'e git.

BP10) $BS = BS + 1$ SON.

Burada BS , kullanılan bidonların sayısını bildirir.

Bidon paketleme problemlerinin bir alt uygulaması stok kesme problemleridir ve giriş bölümünde de bahsedildiği üzere SKP, birçok endüstriyel sektör tarafından günlük uygulamada sıkça kullanılmaktadır.

9. TAMSAYILI PROGRAMLAMA PROBLEMLERİNİ ÇÖZMEK İÇİN KULLANILAN YÖNTEMLER

Günümüzde, matematiksel modellerin özelliklerini göz önüne alarak ayrı ayrı tamsayılı problem sınıflarını çözmek için bir çok sayıda yöntem; ayrıca kesin ve yaklaşık çözümler üreten bazı genel algoritma şemaları bilinmektedir. Ancak unutulmaması gereken bir nokta vardır; tecrübeler göstermiştir ki bu problemler hesaplama açısından en zor çözülen problemlerin başında gelir.

Tamsayılı problemlerin çözüm yöntemleri ikiye ayrılmaktadır. Bunlar, kesin ve yaklaşık çözüm yöntemleridir. *Kesin yöntemler* optimal çözümü garanti ederler; ancak gerek hesaplama zamanı gerekse sarf edilen çaba (zaman veya maliyet) nedeniyle sonuçlar her zaman tatmin edici olmayabilir. Böyle durumlarda *yaklaşık yöntemler* kullanılabilir. Yaklaşık algoritmalar optimal çözüme ulaşmasalar da makul ölçüde yaklaşarak kısa zamanda çözüm üretebilirler.

9.1 Kesin Yöntemler

Bu sınıftaki bazı yöntemler aşağıdaki gibidir:

- Sayımlama (Enumeration),
- Kesen düzlemler (Cutting planes),
- Dinamik programlama (Dynamic Programming),
- Dal-sınır (Branch and bound),
- Dal-kesme (Branch and cut).

Sayımlama

Sayma metodu tüm olasılıkları saymaya dayanan bir yöntemdir. Bulunan tüm olasılıklar için bir amaç fonksiyonu değeri hesaplanır ve sonunda bunlar

arasından en iyi çözüm seçilir. Bu yöntem az değişkene sahip problemler için kullanışlı olmasına rağmen, değişken sayısı arttıkça incelenen olasılıklar üstel biçimde artar. Örneğin n değişkenli sırt çantası probleminde incelenmesi gereken olasılık sayısı 2^n dir. Yani n değeri arttıkça işlem yoğunluğu artar ve çok büyük n değerleri için şu andaki mevcut bilgisayarlarla çözüm süresi yüz yıllar sürebilir.

Kesme yöntemi

Kesme yöntemi tamsayılı doğrusal problemleri çözmek için önerilen yöntemlerden birincisidir. Yöntemin şeması ilk defa Dantzig tarafından açıklanmış sonrasında ise Gomory tarafından geliştirilmiştir.

Kesme yöntemi, tamsayılı programlama probleminin doğrusal programlama problemine gevşetilmesiyle oluşturulur. Yani tamsayılı bir problemdeki tüm kısıtlar olduğu gibi ele alınırken, değişkenlerin tamsayı olması kısıtı ortadan kaldırılır. Bu şekilde doğrusal programlama metotlarından her hangi biri kullanılarak sonuç bulunur; bulunan sonuç tamsayı ise sorun yoktur. Fakat değilse, tamsayı olmayan çözümleri dışarıda bırakacak aynı zamanda tüm tamsayı çözümleri koruyacak yeni bir kısıt eklenir. Bu işlemlere optimal tamsayı çözüm bulunana kadar devam edilir.

Kesme yöntemleri genelde düzensizdir; ayrıca tüm verilerin simpleks tablo şeklinde saklanması büyük boyutlu problemler için soruna neden olmaktadır.

Dinamik programlama

Kesin çözüm yöntemlerinin içinde dinamik programlama ayrı bir yere sahiptir. Bu yöntem, örtüşen alt problemler ve optimal alt yapı özelliklerini sergiler; bu nedenle toplam işlem sayısını ve kullanılan zamanı azaltır. Dinamik programlama algoritmalarının karmaşıklığı teorik olarak oldukça kolay hesaplanabilir ve bu değer reel karmaşıklığa yakındır (Cormen et al., 2001). Ancak iteratif bir yöntem olması dolayısıyla kısıt ve değişken sayısının çok olması durumunda veya çok büyük katsayılar için etkili değildir.

Dal-sınır yöntemi

Dal-sınır algoritması ise optimal çözüme ulaşmak için sistemli bir şekilde uygun çözümleri saymaya dayanır. Ancak sayma yöntemindeki gibi tüm olasılıklar değil, olasılıkların daha küçük bir kısmı incelenir, ümit vermeyen bazı olasılıklar sayma aşamasında kesip atılır. Dal-sınır algoritmasının avantajı hızlı olması, verilmiş her hangi bir tamsayılı problemin özelliklerinin göz önünde bulundurulması ve yöntemin içinde başka tamsayılı metotların kullanılmasına imkan vermesidir. Hesaplama sürecinde ortaya çıkan çözümler genellikle iyi yaklaşık çözümlerdir. Ancak bir dezavantajı bilgisayar belleğine daha çok gerek duymasıdır. Ayrıca değişken sayısının artması hesaplama süresini üstel olarak artırır ve bulunan çözümün optimal çözüm olduğunu ispatlamak çalışma zamanından daha fazla zaman gerektirebilir (Papadimitriou and Steiglitz, 1982).

Dal-kesme yöntemi

Dal-kesme algoritması hibrid bir yaklaşımdır ve dal-sınır algoritması ile kesme yönteminin birleşmesinden oluşur. Verilen problem için öncelikle kesme yöntemi kullanılır, bu süreç bir tamsayı çözüm bulana kadar ya da daha fazla kesme bulunamayıncaya kadar devam eder. Bu aşamada dal-sınır algoritması çözüme dahil olur, bulunan tamsayıya göre dallanarak çözüm bulununcaya kadar çalışmasını sürdürür.

9.2 Yaklaşık Yöntemler

Yaklaşık yöntemler, optimizasyonda kesin yöntemlerin aksine optimal çözümü garanti etmezler. Bununla birlikte;

- Verilen problemin kesin çözüm yönteminin olmadığı durumlarda tek seçenek yaklaşık çözümlerdir.

- Uzun zamanda bulunan kesin çözüme kıyasla kısa zamanda bulunan yaklaşık çözüm daha değerlidir.

- Bilinen kesin çözüm yöntemleri yeteri kadar iyi değildir, çünkü çoğu problem çözümü için zorluklarla karşılaşıyorsa;

- Kesin çözüm yönteminin getirdiği yoğun hesaplama ve çözüm zamanı söz konusuysa yaklaşık yöntemler tercih edilebilir.

9.3 Sezgiseller

Bilgisayar bilimlerinde sezgisel algoritma, ele alınan problemin özelliğine göre tasarlanmış bir algoritmadır ve çözümün doğruluğunun kanıtlanabilir olup olmadığını göz ardı eder. Ancak genellikle en iyi mümkün çözüme makul ölçüde yaklaşarak oldukça hızlı çözümler üretir. Sezgiseller en kötü durum düşünüldüğünde çok başarısız performans gösterebilirler; hatta kötü bir problem örneği seçildiğinde optimumdan çok uzak bir sonuç döndürebilir ve/veya üstel çalışma zamanı gerektirebilir. Bununla beraber iyi sezgiseller bir problemin çoğu örneğinde en iyi yaklaşım algoritmalarının performansını geride bırakabilirler (Ausiello et al., 1999).

Literatürdeki sezgisellerin çoğu ya yinelemeli olarak tek bir uygun çözüm inşa eden "yapıcı sezgisel" ya da uygun çözümlerin bir kümesini inceleyip en iyi sonucu döndüren "sayma sezgiseli" dir. Yapıcı sezgiseller güçlü olarak problem bağımlıdırlar ve genellikle polinom zaman gerektirirler. Diğer taraftan sayma sezgiselleri incelenen küme çok büyük olduğunda üstel çalışma zamanı gerektirebilirler.

Bazı iyi bilinen sezgiseller şunlardır:

- Açgözlü (greedy) algoritmalar,

- Yerel arama (Local search),

- Genetik algoritmalar,

- Tabu araması (Tabu search),

- Karınca kolonisi (Ant colony) vs.

Çalışmamızda açgözlü türündeki algoritmalara yer verilmiştir.

Yerel arama

Yerel arama algoritmaları bazı diğer algoritmalar tarafından bulunmuş bir başlangıç çözümle çalışmaya başlar ve daha iyi bir komşu çözüme taşınma yoluyla yinelemeli olarak geçerli çözümü geliştirir. Çözümü daha fazla geliştirebilecek hiç bir çözüm bulunamadığı takdirde algoritma bir yerel optimumda sonlanır (Aarts and Lenstra, 1997).

Özel bir problem için bir yerel arama algoritması elde etmek için başlangıç uygun çözümü bulan bir algoritma ve bir komşuluk yapısına ihtiyaç vardır. Açıktır ki NP-zor problemler için optimal çözüme ulaşmayı sağlayan bir komşuluk yapısı polinom zamanda bulunamaz; böyle bir durumda yaklaşık çözüm üreten yerel arama algoritmaları araştırılır.

Genetik algoritmalar

Genetik algoritmalar en iyi olanın yaşamasını ve doğal seçim mekanizmasını esas alan algoritmalardır. Bu yöntemde bireylerin popülasyonuyla işe başlanır; 0-1 lerden oluşan her bir birey parametre uzayında bir noktayı temsil eder. Her nesilde her bir birey için amaç fonsksiyonunun sonucu değerlendirilir ve sonunda daha iyi olan bireyler seçilerek yeni bir popülasyon elde edilir. Özel olarak, yeni bir nesil üç aşamayla elde edilir. Bunlar:

- Kalite (sağlamlık) ölçümü,
- Seçim,
- Yeni bireylerin oluşturulması.

Tabu araması

Tabu araması hafızayı temel alan bir arama stratejisidir. Önceki aşamalarda elde edilen bilgiler daha sonraki aşamalardaki yönelimleri belirlemek için kullanılmaktadır. Yöntemdeki amaç, miyop yaklaşımın neden olduğu risklerden kaçınmak için basit yerel aramayı genelleştirmektir. Tabu araması, sonuç için daha iyi bir yerel optimuma götürebilir düşüncesiyle bulunduğu çözümden daha kötü sonuç veren bir komşuya geçişe izin verir.

Açgözlü algoritmalar

Sezgisel bir yöntem olan açgözlü türündeki algoritmalar tasarlama kolaylığı ve optimum çözüme sıkça iyi yaklaşımlar vermesi bakımından oldukça elverişlidir. Eğer verilen bir problem sınıfı için tasarlanan açgözlü algoritmasının optimal değeri ürettiği ispatlanabilirse daha hızlı olan açgözlü türündeki bu algoritma diğer optimizasyon yöntemlerine tercih edilir. Açgözlü algoritmalar her zaman olmasa da bazı problemler için optimal çözümü verirler (Cormen et al., 2001). Bu yöntem oldukça güçlü bir yöntemdir ve problemlerin geniş bir sınıfı için iyi çalışırlar. Örneğin, Dijkstra'nın en kısa yol algoritması, Chvátal'ın küme örtüsü sezgiseli ve Kruskal algoritması iyi bilinen açgözlü algoritmalarından bazılarıdır.

Genel olarak açgözlü algoritmalarının özellikleri aşağıdaki gibidir:

- Tek seferde tek bir karar verir.
- Karar verirken yerel bilgi kullanır.
- Kararı verirken o an için en fazla faydayı almaya bakar; o nedenle açgözlü olarak adlandırılır. Başka bir deyişle global optimal çözüme götürebilir umuduyla yerel optimal seçimler yapar (Cormen et al., 2001).
- Açgözlülüğü hızlı karar vermeyi gerektirir, herhangi bir adımda çözüm kümesinde elde edilen bir değer takip eden adımlarda

değiştirilmez. Yani açgözlü algoritmaları asla seçimlerini tekrar düşünmez.

Açgözlü yöntemi başlangıç olarak nesneleri bazı şartlara göre sıraya koyar ve boş kümeden başlayarak çözüm kümesini genişletir. Yani yukarıda verilen birinci özellik kullanılır: Bir seferde tek nesne için karar verilir. Eğer sondaki çözüm uygun oluyorsa nesne o anki geçerli çözüme eklenir ya da bir daha düşünülmemek üzere elenir.

Açgözlü algoritmasının çalışma zamanı başlangıçta n tane nesnenin sıralanması ve n tane uygunluk testi nedeniyle $O(n)+O(n\log n)=O(n\log n)$ dir (Martello and Toth, 1999). Ayrıca yaklaşık çözümün kalitesi başlangıçtaki sıralamaya bağlıdır. Açıktır ki her problem örneği için her zaman optimal bir sıralama vardır; fakat hesaplama açısından zor bir problemin tüm örnekleri için böyle bir sıralamayı polinom zamanda bulmak pek olası değildir. Bununla birlikte bazı durumlarda, açgözlü yönteminin iyi yaklaşık çözümler bulmasını sağlayacak basit sıralamalar bulunabilir (Ausiello et al., 1999).

9.4 Yaklaşım Algoritmaları

Önemli uygulama alanlarında karşılaşılanlar da dahil olmak üzere çoğu tamsayılı optimizasyon problemi NP-zor problemdir. Dolayısıyla $P\neq NP$ olduğu sürece bu problemler için kesin çözüm aramak sadece zaman kaybıdır. Bu nedenle de polinom zamanlı algoritmalar kullanarak yaklaşık çözümler üretmek bilgisayar bilimleri ve matematiğin ilgi çekici konularından biri haline gelmiştir. NP-zor problemler kesin çözümler bulmak için elverişli değillerse de yakın optimal çözümler bulunabilir. Dolayısıyla yaklaşım algoritmalarının tasarımı aslında kesin çözüm veren algoritmaların tasarım işleminden pek de farklı değildir (Jain and Vazirani, 2001).

10. STOK KESME PROBLEMİ

10.1 Stok Kesme Probleminin Tanımı

Bir boyutlu kesme problemi, tamsayılı ve değişkenleri kısıtlı olan bidon paketleme probleminin bir uygulamasıdır. Küçük ebada sahip parçaların büyük boyutlardaki malzemelerden en uygun hangi şekilde karşılanması gerektiğini bulmaya çalışan problemlere de stok kesme problemi denir. Küçük parçalar büyük nesnelerden kesilirken ortaya iki tür problem çıkmaktadır. Birinci problem ayırma problemi olup, bu problem büyük nesneler için uygun boyutların seçimi konusunu ele alır. İkinci problem ise, işe yaramayan kısmı enküçükleyecek şekilde belirli olan büyük nesnelerden küçük parçaların nasıl kesilmesi gerektiğini içeren kesme kaybı (fire) problemidir (Hinxman, 1979), (Adakci et al., 2010). Uygulamada, sipariş edilen ürünler küçük parçalar, stok malzemeleri ise büyük nesnelerdir. Kesim kaybı problemi ve ayırma probleminin birleşimine stok kesme problemi denir. (Adakci et al., 2010)

Stok kesme problemine, kağıt, çelik, demir, tekstil, cam, vb. gibi pek çok endüstri alanında karşılaşılabilir (Dyckhof, 1990). Bunun dışında ambalaj sektöründeki paketleme işlemi sırasında da benzer problem ile karşılaşılmaktadır. Kesme işlemi, stok malzemesi hacmi içerisinde küçük parçaların paketlenmesi olarak; paketleme işlemi de küçük parçalar için stok malzemelerin kesilmesi olarak ele alınırsa bu benzerlik açıkca ortaya konmaktadır. Bu nedenden dolayı çoğu literatürde kesme ve paketleme problemi (Cutting and Packing Problem) olarak ele alınmaktadır. (Adakci et al., 2010)

Kesme problemlerinin temel özelliklerini aşağıdaki gibi sıralayabiliriz:

- Ana malzeme boyutları
- İstenen parça çeşidi
- Her bir parçanın boyutu
- Talep miktarı
- Kesme tekniği

Stok kesme problemleri, eğer küçük bir çalışma uzayı içinde değilse, bu problemler en iyi çözümün üretilmesinini mümkün olmadığı NP-tam problemlerdir. Arama uzayı çok büyük olduğundan problemlerin çözümü için yönlendirilmemiş bir şekilde arama yapmak verimsiz olduğundan, probleme ait optimal çözümün bulunabilmesi için büyük arama uzayı içerisinde düzenli arama yapılması gerekir (Callaghan et al., 1999 ve Adakci et al., 2010). Bu sebeple araştırmalar, en iyi çözüme yakın iyi çözümleri verimli bir şekilde bulan olasılıksal yaklaşım teknikleri üzerinde yoğunlaşmaktadır.

Kesme problemlerinde elde edilebilecek en iyi çözümün amacı, stok malzemesinin kullanılabilirliğini artırmak ve böylece fire miktarı en az olan yerleşim planını bulmaktadır (Hopper and Turton, 2001).

Stok kesme problemlerinin ilk formülü 1930'lu yılların sonunda Kantorovich tarafından geliştirilmiş; ancak literatüre 1960 yılında geçmiştir (Kantorovich, 1960). Günümüze kadar geçen altmış yıla yakın sürede de bu konudaki çalışmalarla literatür zenginleştirilmektedir.

Yerleşim planı (desen), daha küçük ebattaki parçaların stok içindeki yerleşimini gösterir. Yerleşim alanında kullanılamayan alan veya yerleşim sırasında bulunmasına rağmen, desen içinde yer almayan parçalar, fire olarak isimlendirilir (Adakci et al., 2010).

Stok kesme probleminin gruplandırılması Dyckhoff tarafından geliştirilmiş olan sınıflandırma şeması ile yapılabilir (Dyckhof, 1990). Bu şema kesme ve paketleme problemlerinin yakın ilişkisinden dolayı her iki türdeki problemlerin gruplandırılması için kullanılabilir (Adakci et al., 2010).

10.2 Stok Kesme Problemi İle İlgili Çalışmalar

İlk olarak SKP'nin formülasyonu 1939'da Kantorovich tarafından gösterilmiştir (Kantorovich, 1960). Bir boyutlu SKP'nin çözümünde en önemli gelişme Gilmore ve Gomory'nin problem çözümü için doğrusal programlamayı kullanarak gecikmiş desen oluşturma yöntemini önerdikleri yenilikçi

çalışmalarıdır (Gilmore and Gomory, 1961). Dyckoff, SKP'yi iki sınıfa ayırmıştır. Bunlar desen odaklı ve madde odaklıdır. Dahası dört ayırt edici özellik kullanarak kesme problemlerini sınıflandırmıştır; boyutluluk, tesisat çeşidi, geniş nesnelerin özellikleri, küçük nesnelerin özellikleri.

Waescher ve Gau daha bir çok durumda uygun tam sayılı çözümlerin elde edilebileceği sonucuna varmıştır (Waescher and Gau, 1996). Gradisor ve ark. Bir boyutlu SKP'yi optimize etmek için tüm stok uzunluklarının farklı olduğu durumlarda ardışık (sekansiyel) sezgisel işlemler sunmuştur (Gradisar et al., 1999). Vance ve ark. (Vance et al., 1994), Vance (Vance, 1998), Vanderbeck (Vanderbeck, 1999, 2000) ve Valerio de Carvalho (Valerio de Carvalho, 1999) sütun üretimi ve dal-ve-sınır (branch-and-bound) kombinasyonlarını kullanan bazı girişimler sunmuşlardır. SKP'lerin büyük çoğunluğu için kesin çözümler elde edebilmişlerdir.

Scheithauer ve ark., bir boyutlu SKP'nin kesin çözümü için bir kesim plan algoritması sunmuşlardır (Scheithauer et al., 2001). Mukhacheva ve ark. bir boyutlu SKP için modifiye bir branch-and-bound yöntemi önermişlerdir (Mukhacheva et al., 2001). Belov ve Scheithauer, çoklu stok uzunluğu olan bir boyutlu SKP için sütun üretimi ile kesim plan algoritmasının kombinasyonunu içeren bir yaklaşım sunmuşlardır (Belov and Scheithauer, 2002). Umetani ve ark., bir boyutlu SKP'de verilen değişik kesim desenlerinin sayısının, verilen bir bağ ile sınırlandırıldığını varsaymışlar; sonra, meta-sezgisel bir yaklaşım ile uyumlu desen üretim yöntemini oluşturmuşlardır (Umetani et al., 2003).

Johnson ve Sadinlija bir boyutlu SKP'nin doğrusal olmayışını, desen değişikliklerini ve desen çalışma uzunluklarını 0-1 değişkenlerini kullanarak çözen yeni bir model oluşturmuşlardır (Johnston and Sadinlija, 2004). Belov ve Scheithuer, branch-and-cut-and-price algoritmasını geliştirerek bir boyutlu ve iki boyutlu 2 aşamalı kesme problemlerine uygulamış ve her branch-and-price düğümünde LP-rahatlandırması kombinasyonunun, Chvatal-Gomory ve Gomory mixed-integer kesmeleri kombinasyonu ile güçlendiğini saptamışlardır (Belov and Scheithauer, 2006).

Son yıllarda farklı yaklaşımlarla problemi çözmek için de bir takım girişimlerde bulunulmaktadır. Dikili ve ark., gemi üretiminde bir boyutlu SKP'nin çözümü için matematik modelleme aşamasında kesme kalıplarının direkt olarak kullanılmasıyla elde edilen yöntemlerle, başarılı eliminasyon yöntemi sunmuşlardır (Dikili et al., 2007). Reinertsen ve Vossen, SKP'yi siparişlerin teslim tarihi gözetilerek yeni bir optimizasyon modeli ve çözüm yolları önermiştir (Reinertsen and Vossen, 2010). Cherri ve ark., SKP'yi kullanılabilir artıklar ve literatürdeki iyi bilinen sezgiselleri geliştirmişlerdir (Cherri et al., 2009). Berberler ve Nuriyev, alt küme toplam problemini bir boyutlu SKP'nin alt problemi olarak düşünüp bir boyutlu SKP'yi çözmüş ve dinamik programlama tabanlı sezgisel çözüm önermişlerdir (Berberler and Nuriyev, 2010). Mobasher ve Ekici, klasik SKP'nin daha genel bir vakasını çalışmış olup, kesme stok problemine setup maliyetini ekleyip toplam üretim maliyetine materyal ve setup maliyetlerini ekleyerek minimize etmeyi amaçlamışlardır. Bunun için klasik bir tam sayılı doğrusal modeli geliştirmişler ve problemin özel bir durumu için algoritma önermişlerdir (Mobasher and Ekici, 2013).

11. STOK KESME PROBLEMİ İLE İLGİLİ GERÇEK HAYAT PROBLEM VE ÇÖZÜM ÖRNEKLERİ

Endüstriyel sektörlerin her birisinde ilgili hammaddenin optimal kesimi ile minimum fire miktarına ulaşılmaya çalışılması SKP'nin çözüm konusudur. Bunun için belirli hammadde boyutları gözetilir. Sipariş edilen parça sayısı ve boyutlarına uygun kesim uygulaması ile SKP çözülebilir. Ancak birleştirme fonksiyonu eklenerek problemin daha az fire miktarı ile çözümlenmesi mümkündür. Örnek olarak çelik endüstrisi, kağıt sanayi vb. kullanıma sunulmuş problemler ve çözüm algoritmaları verilebilir.

11.1 Kağıt Endüstrisi

Bu bölümde, SKP'lerin çözümüne farklı bir bakış açısı kazandıran tekrar birleştirme fonksiyonunun kağıt endüstrisinde nasıl kullanıldığına kısaca değinilmektedir.

Problem

Endüstriyel sektörlerin birçoğu, günümüzde hammaddenin işlenişi ve talebin cevaplanması için, artık hammadde olarak görülen firenin, tekrar değerlendirilmesini sağlamaktadırlar. Kağıt endüstrisinde de, sipariş eldesi sonrası arta kalan kısımlar atılmayıp, klasik SKP'nin üzerine "birleştirme" fonksiyonu da eklenerek, fire miktarını minimum seviyeye indirmiş, bu sayede kar marjını arttırmıştır. Peki, bu birleştirme fonksiyonu nedir? Birleştirme fonksiyonu, kesme işlemi sonucu oluşan artıkların siparişleri karşılamak üzere birleştirilmesidir. Bunun için artıklar öncelikle depolanmalı, gelen siparişe göre tekrar birleştirme veya kesim işlemine tabi tutulmalıdır. Artıkların depolaması için önşart, depolama için kullanılacak artıkların sabit yedek makara genişliğinden küçük olmasıdır. Ancak, SKP zaten NP-tam bir problemdir. Ayrıca bu yedek makaraların genişliği ve birleştirme hesaplamaları da eklendiği için, daha zor bir hale gelmektedir.

Bu problemde SKP'nin çözümü MAJIQTRIM adlı bir yazılım tarafından yapılmıştır.

Öncelikle yedek makaralar için normalde klasik stok kesme probleminde (KSKP) artıkların fazla veya kısıtlı üretimi kabul edilebilir bir durum olarak düşünülmektedir. Fakat işin içine birleştirme opsiyonu girerse, bu durum fire kabul edilir. Çünkü birleştirici ancak kesicinin ürettiği artıkları birleştirebilir. Bu nedenle kısıtlı üretim yetersiz sayıda artık demektir. Fazla üretim ise fire olarak karşımıza çıkacaktır. Optimum sonuçta, hem tek makara ve tek kesme aleti kullanılacaktır hem de sipariş bittiğinde depoda hem yedek kalmayacaktır. (Johnson et al., 1997).

Veri

Girdi verileri;

Bitmiş rulo (finished roll) özelliklerini ve kesici (winder) ve birleştiricinin (skiver) fiziksel kısıtlarını belirtir.

Bitmiş rulo verileri;

width(i) = bitmiş makaranın genişliği. (i ϵ I_0)

qty(i) = bitmiş ruloların sayısı. (i ϵ I_0)

Kesici (winder) verileri;

trim_max (trim_min) = max ve min uygulanabilir kesme deseni genişliği.

crolls_max = kesim desenindeki max rulo sayısı.

Birleştirme verileri;

in_max (in_min) = birleştirme işlemine girecek yedek ruloların max veya min genişliği.

srolls_max = birleştirme işlemine girecek olan yedek ruloların max sayısı.

overlap_max (overlap_min) = birleştirilecek ruloların maximum üst üste gelebilme miktarı.

J_c = mümkün olan bütün kesim desenleri.

J_s = tüm uygulanabilir birleştirme desenleri.

Çıkış bilgisi de kesme ve birleştirme desenlerini ve bunların aktivitelerini içermektedir.

width(i) = i yedek rulonun genişliği, i ϵ I_x. I_x oluşturulan yedek ruloların kümesi.

$$a(\mathrm{i,\ j}) = \begin{cases} j \text{ kesim deseni } (j \in J_c): \text{ ana makaradan çıkmış olan bitmiş veya} \\ \quad i \text{ yedek makaraların,} \qquad i \in (I_0 \cup I_x) \\ j \text{ birleştirme deseni } (j \in J_s): \text{ gerekli boyuttaki bitmiş makaralar} \\ \quad \text{için kullanılan } i \text{ yedek makara sayısını gösterir.} \qquad (i \in I_x) \end{cases}$$

x(j) = j deseninin aktivitesini gösterir.

$I_0 \cap I_x \neq \varnothing$. Çünkü yedek ruloların bazıları bitmiş rulolar ile aynı genişlikte olabilir.

I_0 = oluşturulmak istenen rulolar.

I_x = oluşturulan yedek rulolar.

Örnek problem

Gerçek hayattan alınmış küçük bir problem gösterilmiştir (Johnson et al., 1997).

Sadece 2 tane bitmiş rulo isteniyor. Genişlikleri width(1) = 160cm ve width(2) = 240 cm olsun.

Birincisinden 15 tane (qty(1) =15 tane), ikincisinden 12 tane (qty(2) =12 tane) rulo elde edilmelidir.

Elde de 390 cm'lik ana rulolar vardır.

Bunları kesip, birleştirip 15 tane 160 cm'lik ve 12 tane 240 cm'lik rulolar elde etmek gerekmektedir.

Kesici en fazla 390 cm'lik, en az 370 cm'lik (trim_max=390 cm, trim_min=370 cm) rulolar kesebilmektedir.

Ana rulonun kesiminde en fazla 4 tane rulo açığa çıkmaktadır. (crolls_max = 4)

Birleştiricinin parametreleri ise;

Birleştirilebilecek yedek ruloların maximumu 150 cm, minimumu 60 cm genişliğindedir. (in_max=150 cm, in_min=60 cm)

En fazla 2 adet rulo birleştirilebilmektedir. (srolls_max=2 rulo)

Birbiri üzerine gelen kısım da maximum 7cm, minimum 5 cm'dir. (overlap_max = 7 cm, overlap_min = 5 cm)

MAJIQTRIM adında bir program ile 3 tane kesme deseni hesaplanmıştır. 390 cm'lik ana rulo kesildikten sonra, 4 parça oluşmaktadır: (160 cm, 130 cm, 115 cm, 82.5 cm)

C1 (1.desen): 160 cm + 130 cm + 82,5 cm →17,5 cm'lik kısmı çöptür. (Bu desen 6 kere kullanılır.)

C2 (2. desen): 160 cm+115 cm+ 115 cm = 390 cm (6 kere daha kullanılır.)

C3 (3. desen): 130 cm +130 cm + 130 cm = 390 cm (2 kere daha kullanılır.)

Sonuçta elde 12 tane 160 cm' lik, 12 tane 130 cm' lik, 12 tane 115 cm'lik 6 tane 82,5 cm'lik ürün kalır.

Birleştirme aşaması;

MAJIQTRIM programında, 2 tane de birleştirme deseni verilmektedir.

S1: 82.5 cm + 82.5 cm - 5 cm = 160 cm (3 kez)

S2: 130 cm + 115 cm - 5 cm = 240 cm (12 kez)

3 adet yedek rulo oluşturulur. (82.5 cm, 115 cm, ve 130 cm) Bunlar birleştiricide tamamen kullanılır. Toplam fire miktarı 180 cm olur. (17.5*6 + 5*3 + 5*12)

Toplam kağıt tüketimi = 390*(6+6+2) = 5460 cm, toplam işe yararlılık = %96.703

Eğer birleştirici teknolojisi olmasaydı;

Toplam kağıt tüketimi = 7800 cm, toplam işe yararlılık = %69.74

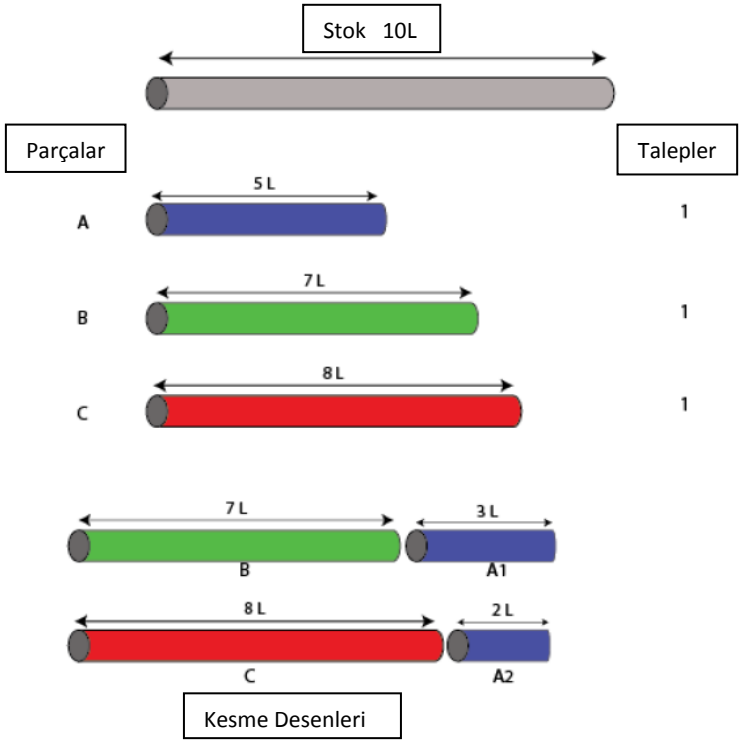
11.2 Çelik Endüstrisi

Problem

Çelik endüstrisindeki bazı şirketler kesme işlemi sırasında, müşterilerin isteklerini daha küçük parçalara ayırıp daha sonra kaynak işlemi ile bu parçaları tekrar bir araya getirerek fire miktarını azaltmayı amaçlamaktadır. Ancak kaynak işlemi şirket için bir maliyet olduğu için, klasik stok kesme problemi artık hem fire miktarını hem de kaynak sayısını minimizasyonu sağlayan çift amaç fonksiyonlu yeni bir bir boyutlu stok kesme problemine dönüşmüştür.

Çelik endüstrisinde en küçük boyutlu maddeler genellikle 1200 mm veya 6000 mm standart uzunluktan oluşur. Şirketlerin temel hedefleri minimum kayıp için optimal kesmeyi belirlemektir. Bir kaynak yaklaşık olarak 250 mm lik çöp miktarına denk gelmektedir. Ekli olarak kullanılacak parçalarında şirket parametrelerine bağlı olarak genelde 1000 mm den küçük olmaması gerekmektedir.

Optimal çelik üretim problemi bir boyutlu SKP olarak takip eden doğrusal olmayan tamsayılı programlama ile sağlanabilir (Tanir et al., 2016).



Şekil 1. Bölünebilir maddelerle bir boyutlu SKP

Şekilde, örnek olarak 10 L uzunluğundaki bir stoktan 1 er tane 5 L, 7L ve 8L uzunluğunda parça talep edilmiştir.

Talep doğrultusunda 2 tane kesim deseni oluşturulmuş ve kayıp yoktur.

Bir boyutlu stok kesme probleminden farklı olarak şekilde görüldüğü gibi A parçasını elde etmek 3 L ve 2 L lik parçalar ayrı desenlerden elde edilmiştir. Bu parçalar kaynak işlemi ile birleştirilerek A siparişi elde edilecektir.

Problemin modellenmesi

Bir boyutlu SKP için doğrusal olmayan tam sayılı programlama formülasyonu sunulmuştur. Önerilen bu yeni problemde n sayıda farklı tipte c uzunluğu olan limitsiz sayıda stok (m stok sayısı olsun) verilmiştir. Kullanılabilir artık e kaynakların üst sınırını belirlemek için kullanıcı tarafından belirlenen β ve θ parametreleri kullanılmıştır.

- β: fire miktarı için üst sınır.
- θ: kaynak sayısı için üst sınır.

β ve θ şirketin isteklerine göre düzenlenebilir. Amaç hem fire hem de kaynak sayısını minimize etmektir.

Problem aşağıdaki gibi formüle edilebilir.

- w_i : i inci siparişin uzunluğu $i=1,...,n$.
- v_i : i inci siparişin miktarı.
- x_{ij} : j inci stoktan kesilen i inci siparişin miktarı.
- y_j : j inci stoğun kesme planında kullanılıp kullanılmadığı
- t_j : j inci stokta kalan çöp miktarı
- z_{ij}^k : j inci stokta i inci siparişin k inci parçasının bölünüp bölünmediğini gösterir. (z_{ij}^k = 1 ise bölünmüştür.
- a_{ij}^k : j inci stoktaki i inci siparişin k ıncı parçasının ayrılmasından sonra kalan parça uzunluğunu gösterir.

- b_{ij}^{k} : i inci siparişinin k. parçası ayrıldıktan sonra kalan parçanın j inci stok uzunluğunda kullanılıp kullanılmadığını gösterir. (b_{ij}^{k} = 1 ise kullanılıyor.)

- u_j : j inci stoktan arta kalanların fre olarak sayılıp sayılmadığını gösterir. ($u_j = 0$ ise j inci stoğun arta kalanlarının uzunluğu β'dan büyüktür ve fire olarak sayılmaz.)

Aşağıda modelleme yapılmış ve tek tek açıklanmıştır.

$$\min \sum_{i=1} t_j u_j \tag{11.1}$$

$$\min \sum_{i=1}^{n} \sum_{k=1}^{v_i} \sum_{j=1}^{m} z_{ij}^{k} \tag{11.2}$$

$$\sum_{i=1}^{n} x_{ij} w_j + \sum_{i=1}^{n} \sum_{k=1}^{v_i} z_{ij}^{k} (w_i - a_{ij}^{k}) + \sum_{i=1}^{n} \sum_{k=1 l=1}^{v_i} \sum_{l \neq j}^{m} b_{ij}^{k} a_{il}^{k} + t_j = c \cdot y_i, \; j = 1, \ldots, m. \tag{11.3}$$

$$\sum_{j=1}^{m} (x_i) + \sum_{k=1}^{v_i} \sum_{j=1}^{m} z_{ij}^{k} = v_i, i = 1, \ldots, n. \tag{11.4}$$

$$\sum_{k=1}^{v_i} \sum_{j=1}^{m} z_{ij}^{k} = \sum_{k=1}^{v_i} \sum_{j=1}^{m} b_{ij}^{k}, i = 1, \ldots, n. \tag{11.5}$$

$$\sum_{k=1}^{v_i} \sum_{j=1}^{m} z_{ij}^{k} \leq 1, \mathrm{j} = 1, \ldots, \mathrm{m}. \tag{11.6}$$

$$a_{ij}^{k} \geq \theta, i = 1, \ldots, n; \; j = 1, \ldots, m; k = 1, \ldots, v_i. \tag{11.7}$$

$$w_i - a_{ij}^{k} \geq \theta, i = 1, \ldots, n; \; j = 1, \ldots, m; k = 1, \ldots, v_i. \tag{11.8}$$

$$v_j = \begin{cases} 0, t_j \geq U\ ise; \\ 1, aksi\ halde. \end{cases} \quad (11.9)$$

$a_{ij}^k \geq 0$ tamsayı, $i = 1, \dots, n;\ j = 1, \dots, m; k = 1, \dots, v_i.$ (11.10)

$x_{ij} \geq 0$ tamsayı, $i = 1, \dots, n;\ j = 1, \dots, m.$

$t_j \geq 0$ tamsayı, $j = 1, \dots, m.$

Yukarıdaki modellemede, (11.1)'de ki amaç fire miktarı toplamını, (11.2)'de ki amaç ise kaynak sayısını minimum yapmaktır. (11.3)'de her bir stok için optimum kesme desenlerinin karar değişkenleri *b, z, a, x* ile gösterilmiştir. (11.4)'te karşılaşılan her bir madde için tüm talepler karşılanmıştır. (11.5)'de bölünen maddelerin artık parçalarının kullanımı garantilenmiştir. (11.6)'da tek bir kesme modelinde sadece tek bölünmeyen (kesilmeyen) madde bölünebilir ve kullanılabilir olduğu gösterilmiştir. (11.7-11.8)'de bölünen maddelerin parçalarının uzunluğu, θ üst sınırından büyük mü kontrol edilir. (11.9)'da ise gelecekte artık stokların kullanılıp kullanılmayacağına karar verilir.

Problemin çözüm yöntemi

Bu problem NP-zor bir problemdir. Zamandan tasarruf amacıyla sezgisel algoritma önerilmiştir. Bu sezgisel işlem Berberler ve ark (2011)'de gösterildiği şekilde (BBP) modifikasyonuyla elde edilmiştir. Önerilen algoritma öncelikle fire miktarını ikinci olarak da hem kullanılabilen artık hem de kaynak sayısını minimize etmek için dizayn edilmiştir. BBP algoritması altküme problemini, bir boyutlu SKP'nin bir alt problemi olarak düşünür.

Algoritma her basamakta, dinamik programlama ile kesme deseni bulmaya çalışır. Ancak standart dinamik programlamanın aksine algoritma, n*c boyutlu iki dizi yerine tek boyutlu A dizisini kullanır. A'yı inşa emek için aşağıdaki fonksiyon kullanılır.

$$F_k(\mathrm{s}) = \max_x \{ \sum_{i=1}^{k} a_i x_i \mid \sum_{i=1}^{k} w_i x_i \le s, \quad 0 \le x_i \le v_i, \quad x_i \in N \ , \quad i = \overline{1,\mathrm{k}} \}$$

$(s = \overline{1,\mathrm{c}} \quad k = \overline{1,n})$ *iken*

$(s = \overline{1,\mathrm{c}} \quad k = 1)$

$$F_1(\mathrm{s}) = \begin{cases} (1, \mathrm{s}/\, w_1) \ , & \textit{eğer} \quad s \le v_1 w_1 \quad \textit{ve} \quad s / w_1 = \lfloor s / w_1 \rfloor \\ (0,0) \quad , & \textit{aksi halde} \end{cases}$$

$(k = \overline{2,n} \quad s = \overline{1,\mathrm{c}})$

$$F_k(\mathrm{s}) = \begin{cases} F_{k-1}(\mathrm{s}) \quad , & \textit{eğer} \quad F_{k-1}(\mathrm{s}) \ne 0 \ \textit{veya} \ s \le w_k \ \textit{veya} \ s \ge \sum_{j=1}^{k} w_j v_j \\ (k, p+1) \ , & \textit{eğer} \quad F_k(\mathrm{s} - w_k) = (k, p) \ \textit{ve} \ p < v_k \\ (k,1) \quad , & \textit{eğer} \quad \mathrm{s} = w_k \ \textit{veya} \ F_k(\mathrm{s} - w_k) \ne 0 \end{cases}$$

Dinamik programlama gerçekleştirildikten sonra eğer dizinin son hücresi doluysa stok dolu demektir. Ama dolu değilse maximum numaralı dolu hücre optimal çözümü gösterir. Bu işlem tüm talepler tamamlanana kadar devam eder (Berberler et al., 2011). Bu yeni probleme göre BBP'yi modifiye etmek için A ile aynı boyutlu başka bir B dizisi kullanılmıştır ve her iki dizi de aynı boyutta üretilmiştir. A dizisi üretilirken, B dizisi de A'daki çözümleri geliştirmeye çalışmaktadır. Bunu her basamaktaki maddenin daha uygun olan daha kısa parçasını bularak sağlamaya çalışır. Eğer daha kısa bir parça çözüme eklenerek stoğu doldurabilirse, o zaman B dizisinde bu işaretlenir. Bu problem iki ana amaç içerdiği için kullanılacak kesim desenine karar vermek için δ ve γ parametreleri kullanılır.

δ : 1 mm'lik fire miktarının bize maliyeti

γ : bir kaynak işleminin maliyeti

A ve B dizisi oluşturulduktan sonra şirket için en az maliyetli olan kesme deseni dizilerden seçilir. Sonra kalan parçalar yeni bir ürün olarak diğer basamaklara sunulur ve A dizisin tekrar oluşturulmak için kullanılır. Bu işlem tüm talepler karşılanana kadar devam ettirilir. Küçük δ değerleri için fire miktarlarının sıfıra indirildiği görülür. Ama daha büyük δ değerleri için algoritma, daha az kaynak işlemi içerecek ama daha fazla fire miktarına sebep olan çözümler bulacaktır. Bu nedenle parametreler pek çok değer verilerek şirketin amaçlarına yönelik değiştirilebilirler.

Önerilen bu yaklaşımın avantajı her basamakta optimal bir kesim deseni belirlemektir. Ayrıca bu algoritma sayesinde en iyi kesim deseni için her basamakta bölünmüş maddeleri içeren en iyi kesim deseni bulunmaktadır. Fire miktarını, kaynak sayısını ve kullanılabilir artık sayısını minimize ederek önerilen yöntem, analitik yöntemler içindeki en ideal çözümü bulabilmektedir.

Algoritma

A1) Siparişlerin uzunluk ve adet değerleri girilir.

A2) Desen sayısı 0 alınır.

A3) Eğer siparişteki parçalar elde edildiyse A8'e git.

A4) Giriş verileri göz önüne alınarak bir deseni elde etmek için dinamik programlama algoritması kullanılarak kesin çözüm bulunur.

A5) Elde edilen desenin maximum kullanılabileceği adet sayısı parça sayıları dikkate alınarak belirlenir.

A6) Siparişlerdeki parça sayısından kullanılabilecek desen sayısı ile desende parçanın kullanıldığı sayının çarpımı çıkarılır. Sipariş adet değerleri güncellenir.

A7) Toplam desen sayısı elde edilen desenin kullanıldığı sayı kadar arttırılır. A3'e git.

A8) SON.

12. SONUÇ

Günümüzde, işletmeler, piyasa rekabetinin şirket lehine devam ettirilebilmesi için, maliyet azaltılması için hammadde kullanımının optimizasyonuna yönelmiştir. Bu amaçla uygun maliyetli hammadde kullanımının yanında, bu hammaddenin etkin kullanımı da çok önemlidir. Etkin hammadde kullanımı gözetilirken çözülmesi gereken problemler atık yönetimi, makine kullanımı, depolama vb. problemlerdir. Öyle ki pek çok endüstriyel sektörde bu problemlerin değişik boyutlarda çözümleri aranmaktadır.

İkinci dünya savaşı başlangıç döneminden itibaren deneme yanılma yöntemleriyle hammadde yönetimi yapılagelmişken; 1960'lı yıllarda hammaddenin yönetimine yönelik stok kesme problemi oluşturularak çözülmüştür. Bu sayede göz kararı bir yöntem izlemekten ziyade bilimsel dayanağı olan bir problem çözümü sağlanmıştır. Bunun sonucunda, hammaddenin daha etkin kullanımı sağlanmış ve birbirinden bağımsız şirketlerin rekabet ortamında ayakta kalması kolaylaştırılmıştır.

Demir, çelik, kağıt, deri, ahşap gibi pek çok sektörün odak noktası olan stok kesme problemleri hammaddenin etkin kullanımı problemine yönelik bir çözümdür. Ne var ki; bu çözüm hammadde atıklarının artması sebebiyle optimal bir çözüm sunamamaktadır. Birleştirme fonksiyonunun eklenmesi durumunda bu kayıp minimuma indirgenerek maliyetin daha küçülmesi sağlanmış; böylece kuruluşların kar marjı artmıştır.

Bu tez çalışmasında, öncelikle matematiksel modelleme ve problemlerin karmaşıklık sınıfları hakkında bilgi verilmiş olup, daha sonra doğrusal programlama, tamsayılı programlama, sırt çantası problemi, altküme toplamı problemi, bidon paketleme problemi ve stok kesme problemleri hakkında bilgi verilmiştir. Stok kesme probleminin tanımı, ve kağıt ve çelik sektöründe bir boyutlu stok kesme problemi ile ilgili bir uygulamalı problem çözümü yapılarak bu iki sektörde de birleştirme fonksiyonunun kullanımı ile birlikte fire miktarının ciddi anlamda azaldığı gösterilmiştir.

KAYNAKLAR DİZİNİ

Aarts E. and Lenstra J. K., 1997, Local Search in Combinatorial Optimization, John Wiley&Sons, England, pp. 512.

Adakci et al., 2010, Cutting Stock Problem: An Application In Aluminium Industry, Published Master Thesis, Istanbul Technical University, Istanbul.

Andonov R., Poirriez V., Rajopadhye S., 2000, Unbounded knapsack problem: dynamic programming revisited, European Journal of Operational Research, 123, pp. 394-407.

Ausiello, Giorgio, et al., 1999, "Heuristic methods." *Complexity and Approximation*. Springer Berlin Heidelberg, pp. 321-351.

Bakır, M.A. ve Altunkaynak, B.T., 2003, Tamsayılı Programlama, Nobel Yayınları, Ankara, 620p.

Balev S., Yanev S. Freville A., Andonov R., 2001, A dynamic programming based reduction procedure for the multidimensional 0-1 knapsack problem, Technical report, University of Valenciennes.

Bertsimas D., Demir R., 2002, An approximate dynamic programming approach to multidimensional knapsack problems, Management Science, 48, pp. 550-565.

Belov, G., Scheithauer, G., 2002, A cutting plane algorithm for the one-dimensionalcutting stock problem with multiple stock lengths. European Journal of Operational Research. 141, pp. 274-294.

Belov, G., Scheithauer, G., 2006, A branch-and-cut-and-price algorithm for one dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research. 171, pp. 85-106.

KAYNAKLAR DİZİNİ (Devam)

Berberler, M. E., Nuriyev, U. G., 2010, A New Heuristic Algorithm for the One-Dimensional Cutting Stock Problem. Appl. Compuy. Math, 9, 1, pp. 19-30.

Berberler, M. E., Nuriyev, U., Yıldırım, A., 2011., A software for the one-dimensional cutting stock problem. Journal of King Saud University-Science, 23, 1, pp. 69-76.

Callaghan, A. R., Nair, A. R. and Lewis, K. E., 1999, An extension of the orthogonal packing problem through dimensional flexibility, *ASME Design Engineering Technical Conferences*, pp. 12-15.

Caprara A., Kelerler H. Pferschy U., 2003, Approximation algorithm for the multiple subset sum, Journal of Heuristics, 9, pp. 99-111.

Cherri, A.C., Arenales, M.N., Yanasse, H.H., 2009, The one-dimensional cutting stock problems with usable leftover: a heuristic approach. European Journal of Operational Research. 196, 3, pp. 897-908.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C., 2001, Introduction To Algorithms, The MIT Press, Cambridge, 1180p.

Dikili A.C., Sarz E., Pek N. A., 2007, A successive elimination method for one-dimensional stock cutting problems in ship production. Ocean engineering. 34, 13, pp. 1841-1849.

Dyckhof, H., 1990, A typology of cutting and packing problems. European Journal of Operational Research. 44, pp. 145-159.

Garey, M.R. and Johnson, D.S., 1979, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 338p.

KAYNAKLAR DİZİNİ (Devam)

Gilmore, P.C., Gomory, R.E., 1961, A linear programming approach to the cutting-stock problem. Operations Research. 9, pp. 849-859.

Gradisar M., Resinovic G., Kljajic, M., 1999, A hybrid approach for optimization of one-dimensional cutting. European Journal of Operational Research. 119, 3, pp. 719-728.

Hinxman, A. I., 1979, The trim-loss and assortment problems: A survey. European Journal of Operational Research, 5, pp. 8–18.

Hopper, E. and Turton, B., 2001, A Review of The Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems, *Artificial Intelligence*, pp. 257-300.

Jain, Kamal, and Vijay V. Vazirani., 2001, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation." *Journal of the ACM (JACM)* 48, 2, pp. 274-296.

Johnson, M. P., Rennick, C., Zak, E., 1997, Skiving Addition to the Cutting Stock Problem in the Paper Industry. Siam Review. 39, 3, pp. 472-483.

Johnston, R.E., Sadinlija, E., 2004, A new model for complete solutions to one-dimensional stock problems. European Journal of Operational Research. 153, pp. 176-183.

Kantorovich, Leonid V., 1960, Mathematical methods of organizing and planning production. Management Science 6, 4, pp. 366-422.

Kara, İ., 1991, Doğrusal Programlama, Bilim Teknik Yayınevi, Eskişehir, 270s.

KAYNAKLAR DİZİNİ (Devam)

Kellerler H., Prefschy V., Pisinger D., (2004), Knapsack Problems, Berlin, Springer, 546p.

Martello S., Pisinger D., Toth P., 1999, Dynamic programming and strong bounds for the 0-1 knapsack problem, Management Science, 45, pp. 414-424.

Martello, S., Toth, P., 1990, Knapsack Problems : Algorithms and Computer Implementations, J. Wiley & Sons, 296p.

Mobasher, A., Ekici A., 2013, Solution approaches for the cutting stock problem with setup cost.Computers and Operations Research. 40, pp. 225-235.

Mukhacheva, E. A., Belov, G,, Kartak, V., Mukhacheva, A. S., 2001, Onedimensional cutting stock problem: Numerical experiments with the sequential value correction method and a modi_ed branch-and-bound method. Pesquisa Operacional. 2000, 2, pp. 153-168.

Nabiyev, V.V., 2005, Yapay Zeka, Seçkin Yayıncılık, Ankara, 764s.

Nabiyev, V.V., 2009, Teoriden Uygulamalara Algoritmalar, Seçkin Yayıncılık, Ankara, 824s.

Papadimitriou, C.H. and Steiglitz,K., 1982, Combinatorial Optimization: Algorithms and Complexity.Prentice Hall, 510p.

Reinertsen, H., Vossen Thomas W.M., 2010, The one-dimensional cutting stock problem with due dates. European Journal of Operational Research. 201, 3, pp. 701-711

KAYNAKLAR DİZİNİ (Devam)

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

Taha, H. A., 2000, Yöneylem Araştırması, Literatür Yayıncılık, İstanbul.

Tanir, D., et al., 2016, "One-dimensional Cutting Stock Problem with Divisible Items." *arXiv preprint arXiv:1606.01419*.

Umetani, S., Yagiura, M., Ibaraki, T., 2003, One-dimensional cutting stock problem tominimize the number of di_erent patterns. European Journal of Operational Research. 146, pp. 388-402.

Vance, P., Barnhart, C., Johnson, E.L., Nemhauser, G.L., 1994, Solving binary cutting stock problems by column generation and branch-and-bound. Computational Optimization and Applications. 3, pp. 111-130.

Vance, P., 1998, Branch-and-price algorithms for the one-dimensional cutting stock problem. Computational Optimization and Applications. 9, pp. 211-228.

Vanderbeck, F., 1999, Computational study of a column generation algorithm for bin-packing and cutting stock problems. Mathematical Programming A. 86, pp. 565-594.

Vanderbeck, F., 2000, On DantzigWolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research. 48, pp. 111-128.

Valerio de Carvalho, J.M., 1999, Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operation Research. 86, pp. 629-659.

KAYNAKLAR DİZİNİ (Devam)

Waescher, G., Gau, T., 1996, Heuristics fort he Integer one-dimesional Cutting Stock Problem. A Computational Study. OR Spektrum 18, 3, pp. 131-144.

Winston W. L., 1991, Operations Research. Application and Algorithms, PWS Publishing, Boston, pp. 1262.

ÖZGEÇMİŞ

Ecem Esma AKBAŞ, 31.10.1992 tarihinde Nürnberg' de doğdu. İlkokulu Zonguldak'ta Bahçelievler İlköğretim Okulu'nda tamamlayıp, ortaokula İskenderun'da Beş Temmuz İlköğretim Okulu' na devam etti. Lise öğrenimini Edirne'de Yıldırım Beyazıt Anadolu Lisesi'nde tamamladıktan sonra 2010 yılında Ege Üniversitesi Fen Fakültesi Matematik Bölümü'nü kazandı. 2015 yılında Matematik Bölümü Bilgisayar Opsiyonu'ndan mezun olarak aynı yıl Ege Üniversitesi Fen Bilimleri Enstitüsü Matematik Anabilim Dalı'nda yüksek lisans öğrenimine başladı.