EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

BAZI AYRIK OPTİMİZASYON PROBLEMLERİNİN MODELLENMESİ VE ÇÖZÜM YÖNTEMLERİ ÜZERİNE

Arif GÜRSOY

Tez Danışmanı: Prof. Dr. Urfat NURİYEV

Matematik Anabilim Dalı

Bilim Dalı Kodu: 619.03.03
Sunuş Tarihi: 01.11.2012

Bornova - İZMİR

2012

Arif GÜRSOY tarafından doktora tezi olarak sunulan "**Bazı Ayrık Optimizasyon Problemlerinin Modellenmesi ve Çözüm Yöntemleri 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 01.11.2012 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	**: Prof.Dr. Urfat NURİYEV**	**.....**
Raportör Üye	**: Doç.Dr. Mücella GÜNER**	**.....**
Üye	**: Prof.Dr. Efendi NASİBOĞLU**	**.....**
Üye	**: Prof.Dr. Pınar DÜNDAR**	**.....**
Üye	**: Prof.Dr. Alpay KIRLANGIÇ**	**.....**

ÖZET

BAZI AYRIK OPTİMİZASYON PROBLEMLERİNİN MODELLENMESİ VE ÇÖZÜM YÖNTEMLERİ ÜZERİNE

GÜRSOY, Arif

Doktora Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Prof. Dr. Urfat NURİYEV
Kasım 2012, 83 sayfa

Bu tezde, ayrık optimizasyon problemleri olan sırt çantası problemi, bidon paketleme problemi ve tekstil sektöründe karşılaşılan hat dengeleme problemleri incelenmiştir. Hat Dengeleme Problemi, operasyonların iş istasyonlarına dağıtım problemidir. Hat dengelemede, akıcı bir üretim hattı için boş zamanların ortadan kaldırılması ve işin çalışma noktalarına dengeli olarak dağıtılması amaçlanır. Yalın üretim yaklaşımı üretime yük getiren israflardan arınmayı hedef alır ve ana stratejisi hızı artırıp, akış süresini azaltarak kalite, maliyet ve teslimat performansını iyileştirmektir.

Tezde, "Esnek yalın üretim hat dengeleme problemi" için yeni bir matematiksel model verilmiş, problemin NP-tamlığı ispatlanmış, polinomiyal karmaşıklıkta yeni bir sezgisel algoritma, bu algoritmaya dayalı bir bilgisayar yazılımı (paket program) hazırlanmış ve örnek bir model kullanılarak hesaplama denemeleri yapılmıştır. Ayrıca, "Performansa dayalı hat dengeleme problemi", "Esneklik kısıtlı yalın üretim hat dengeleme problemi" ve "Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi" adında üç yeni hat dengeleme problemi sunulmuş, bu problemler için matematiksel modeller oluşturulmuş, NP-tamlık ispatları yapılmış ve sezgisel algoritmalar oluşturulmuştur.

Anahtar sözcükler: kombinatoriyal optimizasyon, sırt çantası problemi, bidon paketleme problemi, hat dengeleme problemi, tamsayılı programlama, sezgisel algoritma, genetik algoritma, tekstil ve konfeksiyon teknolojisi, yalın üretim, tam zamanlı üretim, paket program

ABSTRACT

ABOUT MODELING AND SOLUTION APPROACHES OF SOME DISCRETE OPTIMIZATION PROBLEMS

GÜRSOY, Arif

Ph. D. in Mathematics
Supervisor: Prof. Dr. Urfat NURİYEV
November 2012, 83 pages

In this thesis, the knapsack problem, the bin packing problem and the line balancing problems encountered in the textile industry, which are discrete optimization problems, are investigated. The line Balancing Problem is the problem of the distribution of the operations to the workstations. It is aimed that elimination of the idle time and distribution to the workstations equally in the line balancing. Lean manufacturing eliminates the need to keep stocks and aims to enable the low-cost and high-quality production and its main strategy is to improve quality, cost and delivery performance while reducing the flow time.

In the thesis, for the “Flexible lean production line balancing problem”, a new mathematical programming, a new heuristic algorithm having polinomial complexity, a pocket program based on the heuristic and the computational experiments on a sample model are prepared, and its NP-completeness is proved. Besides, three new line balancing problems whose names are to the “Performance based line balancing problem”, the “Flexibility constrained lean production line balancing problem” and the “Performance based flexibility constrained lean production line balancing problem” are presented, new mathematical programmings and new heuristic algorithms are created, and their NP-completenesses are proved for the three new line balancing problems.

Keywords: combinatorial optimization, knapsack problem, bin packing problem, line balancing problem, integer programming, heuristic algorithm, genetic algorithm, textile and apparel technology, lean manufacturing, just-in-time production, pocket program

TEŞEKKÜR

Matematik ve Bilgisayar Bilimleri alanlarındaki geniş bilgi ve tecrübesiyle çalışmalarıma yön veren, danışmanım, değerli hocam sayın Prof. Dr. Urfat NURİYEV'e ve Tekstil Mühendisliği alanındaki teorik ve teknik anlamdaki bilgi ve tecrübeleriyle bana yardımcı olan hocam sayın Doç. Dr. Mücella GÜNER'e en içten teşekkülerimi sunarım.

Nice zorluklarla beni büyüten, eğitim görmemi sağlayarak bugünlere gelmemde üzerimde sonsuz emekleri olan Annem'e ve rahmetli Babam'a saygı ve şükranlarımı sunar, varlığıyla destek olan sevgili Kardeşim'e teşekkür ederim.

Son olarak, desteğini hiçbir zaman benden esirgemeyen ve beni sürekli teşvik eden yoldaşım, sevgili eşim Öğr. Gör. Necla GÜRSOY'a sonsuz teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ÖZET v

ABSTRACT vii

TEŞEKKÜR ix

1. GİRİŞ 1

2. OPTİMİZASYON PROBLEMLERİ 3

2.1 Problemlerin Karmaşıklık Sınıfları 4

2.1.1 P, NP sınıflar, NP-tamlık ve NP-zorluk 6

2.2 Ayrık Optimizasyon Problemlerinin Çözüm Yöntemleri 8

2.2.1 Kesin yöntemler 8

2.2.2 Yaklaşık yöntemler 10

3. SIRT ÇANTASI PROBLEMİ 13

3.1 Sırt Çantası Problemlerinin Sınıflandırılması 13

3.1.1 Negatif olmayan tamsayı sırt çantası problemi 13

3.1.2 Sıfır-Bir sırt çantası problemi 14

3.1.3 Bir boyutlu sırt çantası problemi 15

3.1.4 Çok boyutlu sırt çantası problemi 15

İÇİNDEKİLER (devam)

Sayfa

3.2 Sırt Çantası Probleminin NP-tamlığı 17

4. BİDON PAKETLEME PROBLEMİ 19

4.1 Bidon Paketleme Probleminin Matematiksel Modeli 19

4.2 Bidon Paketleme Probleminin NP-tamlığı 21

4.3 Bidon Paketleme Probleminin Uygulama Alanları 21

4.4 Bidon Paketleme Problemi için Bazı Önemli Sezgisel Algoritmalar ... 22

4.4.1 Online algoritmalar 23

4.4.2 Offline algoritmalar 26

4.4.3 Online ve offline algoritmaların karşılaştırılması 27

5. HAT DENGELEME PROBLEMİ 29

5.1 Planlama 29

5.1.1 Üretim aracı ve personelin planlanması 29

5.1.2 İşletmenin planlanması 30

5.2 Dikim Hattının Dengelenmesi 30

5.3 Yalın Üretim 32

5.3.1 Tam zamanlı üretim 32

İÇİNDEKİLER (devam)

Sayfa

5.3.2 Yalın üretim unsurları 33

5.4 Hat Dengeleme Problemi Modelleri 35

5.4.1 Esnek yalın üretim hat dengeleme problemi.....35

5.4.2 Esnek yalın üretim hat dengeleme problemi – yeni bir model ve sezgisel algoritma.....39

5.4.3 Performansa dayalı hat dengeleme problemi.....49

5.4.4 Esneklik kısıtlı yalın üretim hat dengeleme problemi 56

5.4.5 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi 62

6. SONUÇ 73

KAYNAKLAR DİZİNİ 75

ÖZGEÇMİŞ 79

EK – Esnek Yalın Üretim Hat Dengeleme Programı Yazılımının C# Programlama Dili Kodları.....

ŞEKİLLER DİZİNİ

Şekil Sayfa

5.1 Esnek yalın üretim hat dengeleme problemi yazılımı……... 48

5.2 Performansa dayalı hat dengeleme problemi için operasyonlar ve standart zamanları……49

5.3 Performansa dayalı hat dengeleme problemi için operatörler ve günlük çalışma zamanları……..... 50

5.4 Performansa dayalı hat dengeleme problemi için operasyon standart zamanının operatördeki değişimi……..... 51

5.5 Performansa dayalı hat dengeleme problemi için operatörlerin operasyonlar üzerindeki performans oranları matrisi……..... 51

5.6 Esneklik kısıtlı yalın üretim hat dengeleme problemi için operasyonlar ve standart zamanları……... 56

5.7 Esneklik kısıtlı yalın üretim hat dengeleme problemi için operatörler ve çalışma zamanı……..... 57

5.8 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için operatörlerin operasyonlar üzerindeki performans oranları matrisi... 63

TABLOLAR DİZİNİ

Tablo Sayfa

5.1 Esnek yalın üretim hat dengeleme problemi için örnek bir modelin operasyon detayları …….. 40

5.2 Esnek yalın üretim hat dengeleme problemi için operasyon ataması ve verimlilik sonuçları……... 46

5.3 Esnek yalın üretim hat dengeleme problemi için örnek modelde %90 ve üstü verimlilikli çözümler……..... 47

5.4 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için örnek model operasyon detayları……... 63

1. GİRİŞ

Yöneylem araştırması, İkinci Dünya Savaşı'nda yeni bir bilimsel disiplin alanı olarak gelişmiş ve savaştan sonra özellikle sanayi problemlerinin optimizasyonunda yaygın olarak kullanılmaya başlanmıştır. Hem bilimsel olarak hem de uygulama olarak hızlı bir gelişme gösteren yöneylem araştırması, üniversitelerde, yöneylem araştırması, matematiksel programlama ve optimizasyon gibi adlar altında öğretilmektedir (Bakır ve Altunkaynak, 2003).

Yöneylem araştırması içinde bir grup problemler alanını tanımlayan ve temelini matematikten alan ayrık optimizasyon, mühendislik, ekonometrik, askeri vb. uygulama alanlarına karar stratejileri geliştirmek için kullanılır (Bakır ve Altunkaynak, 2003).

Bu tezde paketleme mantığına dayanan bazı ayrık optimizasyon problemlerinin modellenmesi ve çözüm yöntemleri üzerinde araştırmalar yapılmış ve yeni matematiksel modeller ve sezgisel algoritmalar literatüre kazandırılarak NP-tamlık ispatları yapılmıştır.

Tezin ikinci bölümünde optimizasyon problemlerinin genel tanımı ve modeli verilmiş, belirli kriterlere göre yapılan sınıflandırmaları anlatılmıştır. Bu bölümde ayrıca optimizasyon problemlerinin karmaşıklık sınıfları, ayrık optimizasyon problemlerinin tanımı ve bu problemlere yönelik olarak geliştirilmiş bazı kesin ve yaklaşık çözüm yöntemleri hakkında bilgi verilmiştir.

Tezin üçüncü bölümünde sırt çantası problemine ilişkin bazı tanımlamalar yapılmış ve karar değişkenine ve kısıtlarına göre problem sınıflara ayrılmıştır. Bunun yanında yapılan sınıflandırmalar genel formlardaki matematiksel modeller ile detaylandırılmıştır.

Tezin dördüncü bölümünde sırt çantası tipli bidon paketleme problemine değinilmiştir. Bu bölümde bidon paketleme problemi, sırt çantası problemi terminolojisi kullanılarak modellenmiş ve problemin yaklaşık çözümü için oluşturulmuş bazı sezgisel algoritmalar sınıflandırılmıştır. Bidon paketleme probleminin anlatılan sezgisel algoritmaları, online ve offline sezgisel algoritmalar olmak üzere iki kümede sınflandırılmıştır.

Tezin beşinci ve son bölümünde ise tekstil sektöründeki hat dengeleme problemleri ele alınmış ve hat dengeleme sürecinde gerekli olan planlama, yalın üretim, yalın üretim unsurları gibi kavramlara, literatürdeki esnek yalın üretim hat dengeleme probleminin tanımına ve matematiksel modeline değinilmiştir.

Bu bölümde, esnek yalın üretim hat dengeleme problemi için yeni bir tamsayılı matematiksel model, problemin NP-tamlık ispatı, sezgisel bir algoritma ve bu algoritma temelli C# dilinde kodlanmış bir paket program oluşturulmuştur.

Bu bölümde ayrıca, performansa dayalı hat dengeleme problemi, esneklik kısıtlı yalın üretim hat dengeleme problemi ve performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi olmak üzere üç yeni hat dengeleme problemi tanımlanmış, bu problemler için matematiksel modeller ve sezgisel algoritmalar oluşturulmuş ve problemlerin NP-tamlık ispatları yapılmıştır.

2. OPTİMİZASYON PROBLEMLERİ

Optimizasyon problemleri, kendi karar değişkenlerine bağlı en uygun çözümü araştırılan problemlerdir. Optimizasyon problemlerinde amaç, bir fonksiyonun (*amaç fonksiyonu*) en büyüklenmesi veya en küçüklenmesidir. Örnek olarak yapılan işlerin tamamlanma sürelerinin en küçüklenmesi, sırt çantasına konulacak eşyalarla karın en büyüklenmesi gibi amaçlar karar değişkenlerine bağlı fonksiyonlarla ifade edilir. İlgili karar değişkeni x ile gösterilirse amaç fonksiyonu $f(x)$ ile gösterilir (Cura, 2008). Genel bir optimizasyon problemi aşağıdaki şekilde ifade edilebilir:

Amaç fonksiyonu:

$$Enb\,(veya\,Enk)\,f(x) \tag{2.1}$$

Kısıtlar:

$$g_i(x) \geq b_i, \qquad i = \overline{1,m} \tag{2.2}$$

$$h_j(x) = c_j, \qquad j = \overline{1,n} \tag{2.3}$$

$$p_k(x) \leq d_k, \qquad k = \overline{1,s} \tag{2.4}$$

$$x \in G \tag{2.5}$$

Bir sistemin davranışını matematik dilini kullanarak (2.1 – 2.5) ifadelerinde olduğu gibi tanımlayan soyut modellere *matematiksel model* denir. Bu matematiksel modelde, (2.2 – 2.4) modelin kısıtlarını, (2.5) modelin karar değişkeni x'i göstermektedir. x karar değişkeni herhangi bir G kümesinin elemanıdır. (2.2 – 2.5) kısıtları altında (2.1) ifadesi modelin amaç fonksiyonunu göstermektedir (Cura, 2008).

Optimizasyon problemleri çeşitli şekillerde sınıflandırılabilir: $f(x)$ amaç fonksiyonunun x karar değişkeni ile ilgili herhangi bir sınırlama olmaksızın optimizasyonu *kısıtlamasız optimizasyon (unconstrained optimization)*, x karar değişkeni ile ilgili sınırlamanın olduğu optimizasyon problemleri *kısıtlamalı optimizasyon (constrained optimization)* olarak adlandırılır (Karaboğa, 2004).

Bir optimizasyon probleminin amaç ve kısıt fonksiyonları doğrusal fonksiyon ise bu probleme *doğrusal programlama problemi*, bu fonksiyonlardan herhangi biri doğrusal değil ise *doğrusal olmayan programlama problemi* denir. Karar değişkenleri negatif olmayan tamsayı değerler alan doğrusal programlama problemine *tamsayılı programlama problemi*, aksi halde *tamsayı olmayan programlama problemi* denir (Karaboğa, 2004).

Optimizasyon problemlerinin başka bir sınıflandırması ise *sürekli (continious) optimizasyon* ve *ayrık (combinatorial) optimizasyon* problemleri şeklindedir. Sürekli optimizasyonda karar değişkenleri arama uzayında herhangi bir değer alabilir. Ayrık niceliklerin optimal olarak düzenlenmesi, gruplanması, sıralanması veya seçilmesi problemi *ayrık (discrete) optimizasyon problemi* olarak adlandırılır. Ayrık optimizasyonda karar değişkenleri arama uzayında önceden tanımlanmış değerler alabilirler. Bir *x* karar değişkeninin tam sayı değerler alması veya sadece sıfır veya bir değerler alması örnek olarak verilebilir (Karaboğa, 2004; Cura, 2008).

Bir optimizasyon problemi, optimizasyon versiyonu ve tanıma versiyonu olma üzere iki türlü ifade edilebilir. Bir problemin optimizasyon versiyonunda her uygun çözüm bir aday çözümdür ve en iyi değere sahip uygun çözüm aranmaktadır. Örneğin, gezgin satıcı probleminde herhangi tepe kombinasyonu bir uygun çözüm iken en kısa uzunluklu tur (tepe kombinasyonu) aranmaktadır. Tanıma versiyonunda ise, verilen bir sayısal sınır *B*'den daha iyi maliyete sahip çözüm olup olmadığı aranmaktadır. Örnek olarak, "gezgin satıcı probleminde *B*'den daha az maliyetli bir tur var mıdır?" sorusuna cevap aranır. Tanıma problemleri cevapları "evet" veya "hayır" olan sorulardır (Garey and Johnson, 1979; Cormen et al., 2001).

2.1 Problemlerin Karmaşıklık Sınıfları

Bir problemin çözümüne yönelik bir algoritma araştırılmadan önce bu problemin sonlu sayıda aşamada çözülüp çözülemeyeceğinin bilinmesi gereklidir. Algoritmalar teorisine göre evrensel algoritmik modellerin üç türü ele alınmıştır. Birinci türde, algoritma kavramı hesaplama ve nümerik fonksiyonlar gibi matematiksel kavramlarla ilişkilendirilmektedir. Algoritma kavramını ilk olarak biçimlendiren *özyinelemeli fonksiyonlar (recursive function)* bu türün en gelişmiş ve en incelenmiş modelidir. İkinci tür, 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. Üçüncü tür ise, herhangi bir alfabede sözcüklerin değiştirilmesine dayalı kelime işlemcilerdir (Nabiyev, 2009).

Kullanılan algoritma tipine göre tüm problemler eşdeğer değildir. Sezgisel çözümler dikkate alınmazsa, bazılarının problemlerin çözümü diğerlerinden daha hızlı olabilir. Bir problemin algoritmik çözümlerinin sınıflandırılması, algoritmaların yürütülmesi için gerekli işlemlerin sayısı temel alınarak yapılabilir ve buna *algoritmik karmaşıklık* denir. Karmaşıklık, algoritmanın çalıştırılacağı bilgisayarın işlemcisinden bağımsız olarak değerlendirilmelidir (Nabiyev, 2009).

Hesaplama karmaşıklığı, bir algoritmanın ne kadar hızlı çalışacağı ve bilgisayarın belleğinde ne kadar yer kullanacağına dair bilgiler verir. Algoritmanın girdi boyutu veya problem boyutu, girişi tanımlamak için gerekli sembollerin sayısı ile bağıntılıdır. Bir algoritmanın hesaplama karmaşıklığı, hesaplamayı yapmak için gerekli zamanı ölçen zamansal karmaşıklık değerlendirilmesi ve hesaplama için gerekli bellek alanının ölçümü olarak yersel karmaşıklık olmak üzere iki açıdan incelenmektedir. Genelde, zaman karmaşıklığı sadece karmaşıklık olarak isimlendirilmektedir (Nabiyev, 2009).

Algortimaların çalışma zamanlarının belirlenmesi için sabit çalışma zamanlı emirler dikkate alınmaz. Çalışma zamanının matematiksel özdeşliğinin yazılmasıyla bir algoritmanın sergileyeceği performans orantısal olarak tahmin edilebilir. Örneğin, lineer-logaritmik ($n \cdot \log n$) çalışma zamanılı bir sıralama algoritması mikro bilgisayarda da süper bilgisayarda da orantısal olarak aynı çalışır. Çalışma zamanının üst sınırı olan *büyük O* (Big O) notasyonu, en kötü durumdaki çalışma zamanının belirlenmesinde kullanılır ve 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 $f(x) = O\big(g(x)\big)$ dir.

Örneğin, $n^2 + 5n + 3$ fonksiyonunda n'in büyük değerleri için n^2'nin dışındaki toplananlar önemsenmez ve algoritma karmaşıklığı $c = 1$ olmak üzere $c \cdot O\left(n^2\right)$, yani $O\left(n^2\right)$ şeklinde yazılır. Algoritma karmaşıklığı fonksiyonlarına

örnek olarak, ikili arama için $O(\log n)$, doğrusal arama için $O(n)$, matris çarpımı işlemleri için $O(n^3)$, hızlı sıralama algoritması için $O(n \cdot \log n)$ fonksiyonları verilebilir. Bu problemlerin hepsinde $g(x)$ fonksiyonu polinomiyal karakter taşır (Nabiyev, 2009; Cormen et al., 2001).

2.1.1 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. Algoritmik karakter taşıyan P sınıfındaki problemler deterministik makinelerle çözülebilmektedir (Nabiyev, 2005).

Bir problemin çözümü için hazır formüller ve çözüme ilişkin özyinelemeli ifade bilinmediği zaman, çözüm için tarama ve sezgisel seçimlerin yapılması gerekliliği doğar. Bir problemin karmaşıklığı polinomiyal biçimde tanımlanabilirse, bu problemin sonlu sayıda adımla polinomiyal zamanda çözülebileceği söylenebilir. Polinomiyal algoritmaları çalıştıran sanal modeller *deterministik Turing makineleridir* (DTM) ve P sınıfı, DTM ile polinomiyal zamanda çözülebilen problemler sınıfıdır. Polinomiyal zamanda deterministik olmayan Turing makineleriyle (NDTM) çözülebilen problemler ise *NP* (Non-deterministic Polinomial) *sınıfını* oluşturur (Nabiyev, 2009; Garey and Johnson, 1979). P sınıf problemlerine NP sınıfın bir alt kümesi olarak bakılabilir, $P \subset NP$. Çünkü NDTM, DTM ile çözülebilen tüm problemleri çözebilir. (Cormen et al., 2001)

P sınıfı polinomiyal zamanda çözülebilen problemlerden oluşmaktadır. Bu problemler, *k* bir sabit ve *n* problem girdisinin boyutu olmak üzere $O(n^k)$ zamanda çözülebilen problemlerdir. NP sınıfı, polinomiyal zamanda doğrulanabilen problemlerden oluşur. Yani, 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. Kolay bir şekilde, 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 da 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 birincisi 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 sık 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 yeterlidir. Yani, $\Pi \in NP-tam$ olması için aşağıdaki iki durumun sağlanması yeterlidir (Garey and Johnson, 1979):

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

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

2.2 Ayrık Optimizasyon Problemlerinin Çözüm Yöntemleri

Ayrık optimizasyon problemlerine örnek olarak bazı sıralama, çizelgeleme, paketleme ve programlama türünden optimizasyon problemleri verilebilir. Ayrık optimizasyon problemlerinde iki sorun ortaya çıkmaktadır. Birincisi çözümün nasıl temsil edileceği diğeri ise mevcut bir çözümden yola çıkarak komşu çözümlerin seçilmesidir (Cura, 2008).

Ayrık optimizasyon problemlerinin çözüm yöntemleri kesin ve yaklaşık çözüm yöntemleri olarak ikiye ayrılmaktadır. Kesin çözüm yöntemleri optimal çözümü garanti etmesine rağmen hesaplama zamanı ve bellek gereksinimi nedeniyle makul sınırlar içerisinde gerçekleştirilemezler. Yaklaşık çözüm yöntemleri ise optimal çözüme ulaşmasalar da optimale makul sınırlarda yaklaşarak kısa zamanda çözüm üretebilirler.

2.2.1 Kesin yöntemler

2.2.1.1 Sayımlama yöntemi

Sayımlama yöntemi tüm durumları saymaya dayanır. Arama uzayındaki her durum için amaç fonksiyonunun değeri hesaplanır ve içlerinden en iyi olan çözüm seçilir. Bu yöntem az değişkenli problemler için kullanışlıdır. 0-1 sırt çantası problemi için, n adet karar değişkeni için 2^n mümkün durum vardır. Durum sayısının üstel olarak artması n'in büyük değerleri için hesaplanamayacak kadar aşırı hesaplama maliyeti getirecektir (Woolsey, 1998).

2.2.1.2 Kesme yöntemi

Tamsayılı doğrusal programlama problemlerini çözmek için ilk önerilmiş yöntem olan kesme yöntemi, tamsayılı programlama probleminin doğrusal programlama problemine gevşetilmesi mantığına dayanır. Yöntem, ilk defa Dantzig tarafından açıklanmış sonrasında Gomory tarafından geliştirilmiştir. Tamsayılı programlama problemindeki tüm kısıtlar olduğu gibi ele alınırken, değişkenlerin tamsayı olması kısıtı kaldırılır. Bu şekilde doğrusal programlama metotlarından her hangi biri kullanılarak sonuç bulunur. Bulunan sonuç tamsayı ise çözüm elde edilmiştir, aksi halde tamsayı olmayan çözümleri dışarıda bırakacak aynı zamanda tüm tamsayı çözümleri koruyacak yeni bir kısıt (kesme) eklenir. Bu işlemlere optimal tamsayı çözüm bulunana kadar devam edilir. Kesme

yöntemlerinin genelde düzensiz olduğu bilinmektedir. Bunun yanında tüm verilerin simpleks tablo şeklinde saklanması büyük boyutlu problemlerin çözümü için sorun yaratır (Papadimitriou and Steiglitz, 1998).

2.2.1.3 Dinamik programlama

Richard Bellman tarafından geliştirilmiştir. Birbirini izleyen ve birbirini etkileyen bir dizi kararın bütün olarak ele alındığı problemler için geliştirilen karar modellerinin çözümlerinde kullanılır. Kesin çözüm yöntemlerinin içinde dinamik programlama özel bir yere sahiptir. İleri Yineleme ve Geri Yineleme olarak temelde iki biçimde uygulanabilir. İleri Yinelemede, problemin başındaki en küçük birimden başlayarak sonuca ulaşma işlemi, Geri Yinelemede, problemin çözümündeki son aşamadan başlayarak sonuca ulaşma işlemi olarak ifade edilir (Cormen et al., 2001).

2.2.1.4 Dal-Sınır yöntemi

Dal-sınır yöntemi, optimal çözüme ulaşmak için sistemli bir şekilde uygun çözümleri saymaya dayanır. Sayma yöntemindeki gibi tüm durumları değil, durumların daha küçük bir kısmını inceler ve gereksiz bazı durumları sayma aşamasında kesip atar. Dal-sınır yöntemi, hızlı, verilmiş her hangi bir tamsayılı problemin özelliklerini göz önünde bulunduran ve yöntemin içinde başka metotların kullanılmasına imkan veren bir yöntemdir. Hesaplama sürecinde ortaya çıkan çözümler genellikle iyi yaklaşık çözümlerdir. Ancak sürekli dallanma yaptığı için bilgisayar belleğine çok gereksinim duymaktadır. Ayrıca değişken sayısının artması durumların sayısını üstel olarak artırır (Papadimitriou and Steiglitz, 1998).

2.2.1.5 Dal-Kesme yöntemi

Dal-kesme algoritması hibrid bir yaklaşımdır ve dal-sınır yöntemi ile kesme yönteminin birleşmesinden oluşur. Verilen problem için bir tamsayı çözüm bulana kadar veya daha fazla kesme bulunamayıncaya kadar kesme yöntemi kullanılır. Ardından dal-sınır yöntemi ile bulunan tamsayıya göre dallanarak optimal çözüm bulununcaya kadar çalışır (Nemhauser and Woolsey, 1998).

2.2.2 Yaklaşık yöntemler

Yaklaşık yöntemler, kesin yöntemlerin tersine optimal çözümü garanti etmezler. Verilen problemin kesin çözüm yönteminin olmadığı durumlarda veya kesin çözümün makul zaman sınırlarında mümkün olmadığı durumlarda tercih edilirler. Greedy, yerel arama, tabu arama, genetik, karınca kolonileri optimizasyonu, benzetilmiş tavlama ve yapay sinir ağları algoritmaları yaklaşık yöntemlerden bazılarıdır.

2.2.2.1 Greedy algoritmalar

Greedy (Açgözlü) algoritmalar programlama kolaylığı ve hızlı çalışması bakımından oldukça kullanışlıdır. Eğer bir problem sınıfı için tasarlanan greedy algoritmanın optimal değeri ürettiği ispatlanabilirse bu algoritmanın diğer optimizasyon yöntemlerine tercih edilmesi kaçınılmazdır. Greedy algoritmalar her zaman olmasa da bazı problemler için optimal çözümü verirler. Genel olarak greedy algoritmalar, her adımda tek bir karar verir, karar verirken yerel bilgiyi kullanır ve karar verdiği an için en fazla yararı sağlamak ister (Nabiyev, 2009).

Greedy algoritmalar, global optimal çözüme erişmek amacıyla yerel optimal seçimler yapar ve seçimlerini tekrar gözden geçirmez. Greedy yöntemi başlangıç olarak nesneleri bazı kriterlere göre sıraya koyar ve boş kümeden başlayarak çözüm kümesini genişletir (Cormen et al., 2001).

2.2.2.2 Yerel arama

Başka algoritmalar tarafından bulunmuş bir çözümü girdi olarak alan yerel arama algoritması, daha iyi bir komşu çözüme ulaşmak için iteratif olarak mevcut çözümü geliştirir. Çözümün daha fazla geliştirilemediği durumda bir yerel optimumda sonlanır (Aarts and Lenstra, 1997).

2.2.2.3 Tabu arama

Glover tarafından geliştirilmiş iteratif bir araştırma algoritmasıdır. Hafıza esaslı bir arama yöntemi olan tabu arama, önceki aşamalarda elde ettiği veriyi daha sonraki aşamalardaki yönelimleri belirlemek için kullanır. Bu yöntem, yerel en iyi çözümün daha ilerisinde bulunan çözümlerin araştırılabilmesi için yerel sezgisel araştırmaya kılavuzluk eder (Karaboğa, 2004; Glover and Laguna, 1997).

2.2.2.4 Genetik algoritma

Genetik algoritmalar, en iyi bireyin yaşamasını ve doğal seçim mekanizmasını temel alan evrimsel algoritmalardır. Temel olarak evrimsel sistemin doğal işleyişini canlandırabilecek biçimde şekillendirilmiştir. Bu yöntemde bireylerin başlangıç populasyonuyla işe başlanır. Her birey çözüm uzayında bir noktayı temsil eder ve her nesilde her bir birey için amaç fonsksiyonunun sonucu değerlendirilir. Mutasyon ve çaprazlama işlemlerinden sonra daha iyi olan bireyler seçilerek yeni bir populasyon oluşturulur ve istenilen şartlar sağlanana kadar nesiller üretilmeye devam edilir (Gen, 2006; Cura, 2008).

2.2.2.5 Karınca kolonileri optimizasyonu

Karınca kolonileri optimizasyonu gerçek karıncaların beslenme davranışından ilham alan bir yöntemdir. Karıncalar, salgıladıkları feromon maddesini kullanarak, yuvaları ve yemek kaynakları arasındaki en kısa yolu bulma yeteneğine sahiptirler. Bu özellikleri sayesinde karmaşık bir optimizasyon problemiyle benzer özellik gösteren yol bulma problemlerini çözmektedirler. Aynı zamanda kullandıkları bir yol kullanıma kapandığında veya daha kestirme bir yol alternatifi ortaya çıktığında yeni en kısa yolu en kısa zamanda bulabilmektedirler. Doğadaki bu yapının fark edilmesiyle karınca kolonileri gözlemlenmiş ve optimizasyon problemlerinin çözümlerine yönelik bazı yapay sistemler oluşturulmuştur (Cura, 2008; Başkaya, 2005).

2.2.2.6 Benzetilmiş tavlama

Benzetilmiş tavlama, hem sürekli hem de ayrık optimizasyon problemlerine uygun bir yöntemdir. Metallerin kontrollü bir şekilde soğutulmasını örnek almaktadır. Isının yavaş yavaş düşmesi atomların düzgün bir yapı oluşturmasına sağlamakta ve ortaya yüksek yoğunlukta, düşük enerjili bir madde çıkarmaktadır. Benzetilmiş tavlamada, değerini minimuma indirmek istediğimiz fonksiyon termodinamik sistemdeki enerjiye benzetilmektedir (Başkaya, 2005).

2.2.2.7 Yapay sinir ağları

Bilgisayarların birim zamanda yapabildiği işlem sayısının artması yapay sinir ağlarının daha etkin kullanılabilmesini sağlamıştır. İleri ve geri beslemeli olmak üzere iki temel mimari vardır. Yapay sinir ağı, herhangi bir problemi çözmek için çok yüksek sayıdaki birbiriyle etkileşimli parçacıkların (nöronların) uyum içinde çalışmasından oluşan bir bütündür ve yapay sinir ağları insanlar gibi örneklerden öğrenirler (Cura, 2008).

3. SIRT ÇANTASI PROBLEMİ

Sırt Çantası Problemi (Knapsack Problem, SÇP) ayrık optimizasyon problemleri içerisinde tamsayılı doğrusal programlamanın en basit türlerinden biridir. Problem, bir dağcının tırmanırken sınırlı kapasiteye sahip sırt çantasını optimum biçimde doldurması şeklinde tanımlanabilir. Sermaye bütçeleme, portföy seçimi, proje seçimi gibi gerçek dünya problemlerinde karşılaşılan bir çok durum SÇP yardımıyla kolayca ifade edilebilir (Bakır ve Altunkaynak, 2003).

SÇP'nin tanımlanması için aşağıdaki notasyon kullanılmaktadır: Elimizde isimleri "*1*" ile "*n"* arasında sayı ile ifade edilen *n* değişik madde ve sınırlı hacme sahip bir sırt çantası olduğunu düşünelim. Bu maddelerin bir alt kümesi ile sırt çantasını doldurmaya çalışalım. Her *j*. maddenin, p_j kadar yarar (değer) sağladığını ve sırt çantasında w_j kadar yer (ağırlık) kapladığını kabul edelim. Sırt çantasının toplam hacmi *c* kadar olup bu üst sınır aşılamaz. Bu kabuller altında, sırt çantasına sahip olan kişinin amacı, mümkün olduğu kadar kendisine en fazla toplam yararı (değeri) sağlayacak maddeler alt kümesi ile çantasını doldurmak olacaktır. Genel olarak hiç bir değer ve hiçbir ağırlık negatif olamaz ve her madde bir bütün olup parçalara ayrılamaz (Bakır ve Altunkaynak, 2003).

3.1 Sırt Çantası Problemlerinin Sınıflandırılması

SÇP'nin anlaşılmasını kolaylaştırmak için uygulamada karşılaşılan durumlara göre sınıflandırmak yararlı olacaktır. SÇP'ler, karar değişkenleri x_j'lerin tanımı bakımından "Negatif Olmayan Tamsayı Sırt Çantası Problemleri" ve "Sıfır-Bir Sırt Çantası Problemleri" olarak iki türde; problemin içerdiği kısıt sayısı bakımından ise "Bir Boyutlu Sırt Çantası Problemleri" ve "Çok Boyutlu Sırt Çantası Problemleri" şeklinde iki türde sınıflandırabilir (Bakır ve Altunkaynak, 2003; Martello and Toth, 1990; Kellerer et al., 2004).

3.1.1 Negatif olmayan tamsayı sırt çantası problemi

n madde sayısı olacak şekilde karar değişkenini aşağıdaki gibi tanımlayabiliriz.

x_j : sırt çantasına konulacak *j*. türdeki maddelerin sayısıdır ($j = \overline{1,n}$).

SÇP modeli aşağıdaki biçimde ifade edilebilir.

Amaç fonksiyonu:

$$Enb\ Z = \sum_{j=1}^{n} p_j \cdot x_j \qquad (3.1.1.1)$$

Kısıtlar:

$$\sum_{j=1}^{n} w_j \cdot x_j \le c \qquad (3.1.1.2)$$

$$x_j \ge 0 \quad ve\ tamsayı, \qquad j = \overline{1,n}. \qquad (3.1.1.3)$$

Yukarıdaki modelde p_j'ler, karar değişkeni x_j'lerin sırt çantasına konulduğu takdirde birim başına sağlayacağı yararı, (3.1.1.2) kısıtındaki w_j'ler ise *j.* maddenin sırt çantasında kaplayacağı alanı ifade eder. (3.1.1.2) kısıtındaki *c* değeri ise sırt çantasının toplam kapasitesini ifade etmektedir. (3.1.1.3)'e göre karar değişkeni x_j negatif olmayan tam sayı olmalıdır (Bakır ve Altunkaynak, 2003; Martello and Toth, 1990; Kellerer et al., 2004).

3.1.2 Sıfır-Bir sırt çantası problemi

SÇP tipleri arasında en iyi bilinen problem "Sıfır-Bir Sırt Çantası Problemi"dir. Negatif Olmayan Tamsayı SÇP'de karar değişkenine ilişkin oluşturulan kısıt aynı maddeden birden fazla miktarda çantaya konulabilmesini mümkün kılmaktaydı. Fakat Sıfır-Bir SÇP'de mevcut *n* maddeden her biri ya 1 birim olarak çantaya konulur ya da çantaya konulmaz. Bu durumda ikili (binary) bir karar durumu söz konusudur ve x_j değişkeni 0 ya da 1 değeri alabilen tamsayı değişken olarak tanımlanmalıdır (Bakır ve Altunkaynak, 2003; Martello and Toth, 1990; Kellerer et al., 2004).

Karar değişkeni, *j*. maddeyi sırt çantasına koyup koymamasını belirleyecek şekilde aşağıdaki biçimde 0-1 türünden tanımlanabilir.

$$x_j = \begin{cases} 1, & j.\,\text{madde çantaya konulursa} \\ 0, & \text{aksi halde} \end{cases}$$

Bu durumda Sıfır-Bir SÇP şöyle tanımlanabilir:

Amaç fonksiyonu:

$$Enb\ Z = \sum_{j=1}^{n} p_j \cdot x_j \tag{3.1.2.1}$$

Kısıtlar:

$$\sum_{j=1}^{n} w_j \cdot x_j \le c \tag{3.1.2.2}$$

$$x_j \in 0 \vee 1, \qquad j = \overline{1,n}. \tag{3.1.2.3}$$

3.1.3 Bir boyutlu sırt çantası problemi

Eğer Negatif Olmayan Tamsayı SÇP ve Sıfır-Bir SÇP'de olduğu gibi SÇP'de sadece bir tane kısıt var ise bu tür problemler *Bir Boyutlu SÇP* olarak adlandırılır. Yani, dağcının çantası için sadece hacim veya sadece ağırlık olmak üzere tek bir kısıt oluşturulması durumudur (Bakır ve Altunkaynak, 2003; Martello and Toth, 1990).

3.1.4 Çok boyutlu sırt çantası problemi

n tane madde olduğunu ve her bir maddenin hem ağırlığının hem de çantada kaplayacağı hacim değerlerinin verildiğini kabul edelim. Böylece SÇP, hem toplam ağırlık hem de toplam hacim bakımından iki kısıt altında toplam yararı en büyüklemek biçiminde modellenmelidir. Birden fazla kısıta sahip olan bu tür sırt çantası problemleri *Çok Boyutlu SÇP* olarak adlandırılır. Bu tür bir SÇP modeli *m* tane kısıt içerecek şekilde tanımlanabilir (Bakır ve Altunkaynak, 2003; Martello and Toth, 1990).

Çok Boyutlu Negatif Olmayan Tamsayı SÇP nin genel formu aşağıdaki gibidir:

$$Enb\ Z = \sum_{j=1}^{n} p_j x_j \tag{3.1.4.1}$$

Kısıtlar:

$$\sum_{j=1}^{n} w_{ij} \cdot x_j \le c_i, \quad i = \overline{1,m} \tag{3.1.4.2}$$

$$x_j \ge 0 \text{ ve tamsayı} \qquad j = \overline{1,n} \tag{3.1.4.3}$$

Yukarıdaki modelde karar değişkenleri 0-1 türünden olursa, bu durumda problem Çok Boyutlu Sıfır-Bir SÇP'ye dönüşür. Çok Boyutlu Sıfır-Bir SÇP'nin genel formu aşağıdaki gibidir:

$$Enb\ Z = \sum_{j=1}^{n} p_j x_j \tag{3.1.4.4}$$

Kısıtlar:

$$\sum_{j=1}^{n} w_{ij} \cdot x_j \le c_i, \quad i = \overline{1,m} \tag{3.1.4.5}$$

$$x_j \in 0 \vee 1 \qquad j = \overline{1,n}. \tag{3.1.4.6}$$

3.2 Sırt Çantası Probleminin NP-tamlığı

Teorem 3.2.1: SÇP NP-tam problemler sınıfındandır (Garey and Johnson, 1979).

İspat: SÇP nin NP-tamlığını kısıtlama yöntemi ile ispatlayacağız. Bunun için öncelikle SÇP'nin ve kısıtlamada kullanacağımız NP-tam problem Parçalama Probleminin (PP) tanıma versiyonlarını tanımlayalım (Garey and Johnson, 1979).

PP'nin tanıma versiyonu aşağıdaki gibidir.

Koşul: Sonlu bir A kümesi ve $\forall a \in A$ için $s(a) \in \mathbb{Z}^+$ olarak her elemanın ağırlığı verilsin.

Soru: Öyle bir $A' \subseteq A$ var mıdır ki $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$ olsun?

SÇP'nin tanıma versiyonu aşağıdaki gibidir.

Koşul: n elemanlı bir U kümesi, $j = \overline{1,n}$ için $w_j \in \mathbb{Z}^+$ olarak her elemanın ağırlığı, $p_j \in \mathbb{Z}^+$ olarak her elemanın değeri, ağırlık sınırı c ve değer için ulaşmak istenen amaç, K verilsin.

Soru: Öyle bir $U' \subseteq U$ var mıdır ki $\sum_{j \in U'} w_j \le c$ ve $\sum_{j \in U'} p_j \ge K$ olsun?

Genellikten uzaklaşmadan, her eşyanın ağırlığı ile değerinin eşit olduğunu, yani $w_j = p_j$ ve ağırlık sınırı ile ulaşmak istediğimiz değeri ağırlıklar toplamının yarısına eşit kabul etmeliyiz, yani $c = K = \frac{1}{2}\sum_{j \in U} w_j$.

SÇP'nin çözümü için seçilecek eşyalar toplam ağırlığın yarısına eşit ve $\sum_{j \in U'} w_j = \sum_{j \in U - U'} w_j$ kısıtı sağlanmış olacaktır. Bu durumda SÇP PP'ye kısıtlanmıştır ve böylece SÇP $\in$ NP-tam'dır.

4. BİDON PAKETLEME PROBLEMİ

Ağırlıkları $w_1, w_2, ..., w_n$ olan n nesne ve c kapasiteli özdeş bidonlar verilsin. Amaç "c kapasiteli bidonlarda taşma oluşturmayacak şekilde, elimizdeki nesneleri en az kaç bidon kullanarak yerleştirebiliriz?" sorusunun cevabını bulmaktır. Bu problem, boşluk veya zamanı en küçüklemekle ilgili birçok uygulaması olan (bir boyutlu) *Bidon Paketleme Problemi (Bin Packing Problem, BPP)* olarak bilinmektedir (Malkevitch, 2012; Korte and Vygen, 2008).

BPP'de kullanılan en büyük nesnenin ağırlığı bidon kapasitesinden büyük olmamalıdır ve paketlenecek nesneler bölünmez (yağ veya su gibi olmayan) nesneler olmalıdır. Her bir bidon için, o bidonda bulunan nesnelerin ağırlıkları toplamı bidon kapasitesini aşmamalıdır. Genel olarak c kapasitesi 1 ve nesnelerin ağırlıkları $(0,1]$ aralığından seçilir. Fakat kapasite (c) pozitif bir tam sayı ve ağırlıklar kapasiteden büyük olmayan pozitif tam sayılar olarak da alınabilir.

Bidon kavramı, problemin kullanıldığı alana bağlı olarak, zaman aralığı, çubuk uzunluğu, yüzey genişliği veya hacim büyüklüğü olabilir. Gerçek hayat problemlerinde BPP için önemli olan kapasitenin verimli olarak kullanılabilmesidir.

BPP bir ayrık optimizasyon problemidir ve BPP'nin optimizasyon versiyonu, verilen n adet nesneyi paketlemek için gerekli olan c kapasiteli özdeş minimum bidon sayısını arar (Skiena, 1997; Garey and Johnson, 1979).

4.1 Bidon Paketleme Probleminin Matematiksel Modeli

BPP, SÇP terminolojisi kullanılarak aşağıdaki şekilde tanımlanabilir (Martello and Toth, 1990):

n adet paket ve n adet sırt çantası (veya bidon) verilsin.

w_j, j. paketin ağırlığı

c, her bidonun kapasitesi

Karar değişkenleri:

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

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

Her paket bir bidona atanmalıdır ve her bir bidondaki paketlerin toplam ağırlığı c'yi geçmemelidir. Ayrıca kullanılan bidonlar minimum sayıda olmalıdır. BPP'nin matematiksel modeli aşağıdaki şekildedir (Martello and Toth, 1990):

Amaç fonksiyonu:

$$Enk\ Z = \sum_{i=1}^{n} y_i \tag{4.1.1}$$

Kısıtlar:

$$\sum_{j=1}^{n} w_j \cdot x_{ij} \le c \cdot y_i, \qquad i = \overline{1,n}, \tag{4.1.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = \overline{1,n}, \tag{4.1.3}$$

$$y_i = 0 \vee 1, \qquad i = \overline{1,n}, \tag{4.1.4}$$

$$x_{ij} = 0 \vee 1, \qquad i = \overline{1,n}, j = \overline{1,n}, \tag{4.1.5}$$

Burada (4.1.2) kısıtı ile bir bidona yerleştirilen nesnelerin ağırlıklar toplamının bidon kapasitesini geçemeyeceği ve (4.1.3) kısıtı ile de her nesnenin sadece bir bidona yerleştirilebileceği anlatılmaktadır. Ayrıca y_i $(i = \overline{1,n})$ ve x_{ij} $(i = \overline{1,n}$, $j = \overline{1,n})$ matematiksel modelin karar değişkenleri olup, amaç (4.1.3) kullanılacak bidon sayısını en küçüklemektir.

Burada w_j ağırlıkları pozitif tam sayı olmalıdır. Genellikten uzaklaşmadan aşağıdaki ifadeler kabul edilir;

$$c \in \mathbb{Z}^+, \tag{4.1.6}$$

$$w_j \le c, \qquad j = \overline{1,n}. \tag{4.1.7}$$

(4.1.7) kabulü, hiçbir nesnenin bidon kapasitesini aşmaması gerektiğini anlatmaktadır. Eğer herhangi bir nesne için (4.1.7) kabulü ihlal edilirse, verilen örneğin çözümü mümkün değildir.

4.2 Bidon Paketleme Probleminin NP-tamlığı

Teorem 4.2.1: BPP NP-Tam'dır (Garey and Johnson, 1979).

İspat: BPP'nin NP-tamlığını PP'ye kısıtlayarak ispatlayacağız. BPP'nin tanıma problemi versiyonu aşağıdaki şekildedir (Garey and Johnson, 1979):

Koşul: n elemanı olan A kümesi, $j=\overline{1,n}$ için $w_j \in \mathbb{Z}^+$ olarak her elemanın ağırlığı, bidon kapasitesi olarak $c \in \mathbb{Z}^+$ ve $K \in \mathbb{Z}^+$ verilsin.

Soru: A kümesinin $A_1, A_2, ..., A_K$ ayrık kümeleri olacak şekilde bir parçalanması var mıdır ki her bir A_i $(i=\overline{1,K})$ kümesindeki elemanların boyutlarının toplamı c sayısını geçmesin?

Genellikten uzaklaşmadan, bidon sayısını 2, yani $K=2$ ve bidon kapasitesini tüm eşyaların ağırlıklar toplamının yarısı olarak, yani $c=\frac{1}{2}\sum_{j\in A} w_j$ kabul edelim.

Bu durumda öyle bir $A' \subseteq A$ bulunmalıdır ki $A_1 = A'$ ve $A_2 = A - A'$ olmak üzere $\sum_{j\in A_1} w_j = \sum_{j\in A_2} w_j$ kısıtı sağlanmış olsun. Böylece BPP PP'ye kısıtlanmıştır ve BPP $\in$ NP-tam'dır.

4.3 Bidon Paketleme Probleminin Uygulama Alanları

BPP, imalat ve paketleme problemlerinde dikkat çekmektedir. Metal saçlardan parçalar kesilerek metal aletlerin imal edilmesi, kumaş toplarından parçalar kesilerek pantolonların yapılması, bilgisayar üzerindeki disk sürücülerine elektronik verilerin depolanması ve geniş bir müzik koleksiyonunun manyetik bant veya CD'lere yerleştirilmesi örnek olarak verilebilir. (Skiena, 1997)

Standart olarak C uzunluğunda boru veya kablo materyalleri olsun. İstenilen materyal parçaları C uzunluğundan büyük olmayan uzunluklardır. Amaç istenen materyal parçaları kümesini oluşturmak için en az sayıda standart materyal

uzunluklarını kullanmaktır. Bundan dolayı israf en küçüklenmelidir. Keyfi uzunluklardaki reklam parçalarının radyo veya televizyon reklam aralarına yerleştirilmesi gerekir. Her reklam arası sabit büyüklüktedir. Ağırlık limiti olan kamyonlar için eşyalar paketlenmelidir. Burada amaç, herhangi bir kamyon için maksimum ağırlığı aşmamak koşuluyla mümkün olan en az sayıda kamyonu kullanarak tüm eşyaları paketlemektir (Gürsoy, 2009).

Keyfi yürütme zamanına sahip görevler kümesinin bazı özdeş işlemcilerde çalıştırılması gerekir ve her işlemci için bir çalışma süresi sınırı vardır. Burada, en az sayıda işlemci kullanarak verilen çalışma süresi sınırına karşılık gelecek şekilde bir planlama oluşturmak gerekir. Bu problem bir çizelgeleme problemidir. İşlemciler bidonlara, çalışma süresi sınırı bidon kapasitesine ve her görevin yürütme zamanı da nesnelere karşılık gelir. Ayrıca bir çizelgeleme problemi olan Hat Dengeleme Problemi de bir BPP örneğidir (Gürsoy, 2009).

4.4 Bidon Paketleme Problemi için Bazı Önemli Sezgisel Algoritmalar

BPP için yaklaşık çözümler elde edebilmek amacıyla oluşturulan sezgisel algoritmaları *Online (Çevrimiçi)* ve *Offline (Çevrimdışı) Algoritmalar* olarak iki ana başlık altına ayırmak mümkündür (Coffman et al., 1996; Malkevitch, 2012; Coffman et al., 1998). Bu ayrım, işlem yapılacak elemanlar kümesinin bilinip bilinmemesine bağlıdır (Gürsoy, 2009).

Anlatılacak algoritmalarda aşağıdaki değişkenler kullanılacaktır:

n: eleman sayısı,

c: bidon kapasitesi,

bin(i): i. bidon,

k: kullanılan bidonun sayısı $\left(k \le n\right)$,

w_j : j. nesnenin ağırlığı.

4.4.1 Online algoritmalar

Online algoritmalarda, her adımda elemanların bir tanesi için işlem yapılır ve elemanların hangi sırada dizildiği bilinmemektedir. Her bir eleman, sonraki elemana geçmeden önce bir bidona yerleştirilmelidir (Skiena, 1997; Gu et al., 2002). Next-fit, First-fit ve Best-fit algoritmaları bunlardan bazılarıdır.

4.4.1.1 Next-Fit algoritması

Next-fit (NF, Sonraki Sığan) algoritmasında, ilk nesne 1. bidona atanır. 2,…,n nesneleri indis sırasıyla dikkate alınır. Eğer geçerli bidon uygunsa, nesne bu bidona atanır; aksi halde yeni bir bidona atanır ve yeni bidon artık geçerli bidon olarak kabul edilir. Açıkça anlaşılır ki, bu algoritmanın zaman karmaşıklığı $O(n)$ dir (Martello and Toth, 1990). NF algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: k.

```
k = 1
bin(k) = 0
For j = 1 To n
  If bin(k) + wj > C Then
    k = k + 1
    bin(k) = 0
  End If
  bin(k) = bin(k) + wj
Next j
```

4.4.1.2 First-Fit algoritması

First-fit (FF, İlk Sığan) algoritması, NF algoritmasına göre daha iyi bir algoritmadır. Nesneler indislerin artan sırasına göre hesaba katılır. Her bir nesne, mevcut bidonlardan uygun geldiği en düşük indisli bidona atanır. Eğer mevcut bidonlardan hiçbiri o nesneye uygun değilse yeni bir bidona geçilir ve nesne bu bidona atanır (Martello and Toth, 1990). FF algoritmasının zaman karmaşıklığı

$O(n \cdot \log n)$ dir (Johnson et al., 1974). FF algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: *k*.

```
k = 1
bin(k) = 0

For j = 1 to n
  m = False
  For i = 1 to k
if bin(i) + wj <= C Then
bin(i) = bin(i) + wj
m = True
Exit For
End If
  Next i
  if Not m Then
k = k + 1
bin(k) = 0
bin(k) = bin(k) + wj
  End If
Next j
```

4.4.1.3 Best-Fit algoritması

Best-Fit (BF, En İyi Sığan) algoritması yapısal olarak FF algoritmasına çok yakın bir algoritmadır. Bu algoritmada nesneler indislerin artan sırasına göre hesaba katılır. Her bir nesne, mevcut bidonlardan artık alanın en küçük olacağı bidona atanır. Eğer mevcut bidonlardan hiçbiri o nesneye uygun değilse yeni bir bidona geçilir ve nesne bu bidona atanır (Martello and Toth, 1990). BF

algoritmasının zaman karmaşıklığı $O(n \cdot \log n)$ dir (Johnson et al., 1974; Kenyon, 1996). BF algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: k.

```
k = 1
bin(k) = 0

For j = 1 To n
  m = False
  enk = C
  For i = 1 To k
    If bin(i) + wj <= C Then
      If enk > C - bin(i) - wj Then
        enk = C - bin(i) - wj
        ind = i
        m = True
      End If
    End If
  Next i
  If m Then
    bin(ind) = bin(ind) + wj
  Else
    k = k + 1
    bin(k)=0
    bin(k) = bin(k) + wj
  End If
Next j
```

4.4.2 Offline algoritmalar

Bir offline algoritmada, işlemin başında paketlenecek nesneler kümesinin tamamı bilinir (Skiena, 1997). Offline problemlerde tüm elemanlar sıralı bir şekildedir. İlk eleman yerleştirilmeden önce tüm elemanlar kontrol edilir (Coffman et al., 1996). Next-fit-decreasing, First-fit-decreasing ve Best-fit-decreasing bazı offline algoritmalardır.

4.4.2.1 Next-Fit-Decreasing algoritması

Next-fit-decreasing (NFD, Azalan Sonraki Sığan) algoritmasında, nesneler $w_1 \geq w_2 \geq \cdots \geq w_n$ şeklinde sıralanır ve ardından nesneler için yeni oluşturulan sıralama üzerinden NF algoritması uygulanır (Baker and Coffman, 1981). NFD algoritmasının zaman karmaşıklığı $O(n \cdot \log n)$ dir (Martello and Toth, 1990). NFD algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: *k*.

1. *I* kümesinin elemanlarını artmayan şekilde sırala.

2. *NF* algoritmasını uygula.

4.4.2.2 First-Fit-Decreasing algoritması

First-fit-decreasing (FFD, Azalan İlk Sığan) algoritmasında, nesneler $w_1 \geq w_2 \geq \cdots \geq w_n$ şeklinde sıralanır ve ardından nesneler için yeni oluşturulan sıralama üzerinden FF algoritması uygulanır (Martello and Toth, 1990). FFD algoritmasının zaman karmaşıklığı $O(n \cdot \log n)$ dir (Johnson, 1973; Johnson et al., 1974). FFD algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: *k*.

1. *I* kümesinin elemanlarını artmayan şekilde sırala.

2. *FF* algoritmasını uygula.

4.4.2.3 Best-Fit-Decreasing algoritması

Best-fit-decreasing (BFD, Azalan En İyi Sığan) algoritmasında, nesneler $w_1 \geq w_2 \geq \cdots \geq w_n$ şeklinde sıralanır ve ardından nesneler için yeni oluşturulan sıralama üzerinden BF algoritması uygulanır (Martello and Toth, 1990). BFD algoritmasının zaman karmaşıklığı $O(n \cdot \log n)$ dir (Johnson, 1973; Johnson et al., 1974). BFD algoritması aşağıdaki şekildedir (Gürsoy, 2009):

Girdi: $I=\{w_1,...,w_n\}$ kümesi.

Çıktı: *k*.

1. *I* kümesinin elemanlarını artmayan şekilde sırala.

2. *BF* algoritmasını uygula.

BFD algoritması ile FFD algoritmasını karşılaştırmak gerekirse; iki algoritma sonucunda da sayısal veriler olarak aynı çıktılar elde edilmiştir fakat BFD algoritması sonucunda FFD ye göre daha büyük boşluklar kalabilmektedir (Gürsoy, 2009).

4.4.3 Online ve offline algoritmaların karşılaştırılması

Online problemler offline problemlere kıyasla daha zor görünmektedir. Bir online algoritma her zaman optimal sonucu elde edemeyebilir (Suri, 2012). Örneğin,

$0 < e < 0.001$ ve bidon kapasitesini 1 kabul edelim. Toplamda *2m* adet eleman olmak üzere $1/2 - e$ boyutlu *m* adet küçük eleman ve $1/2 + e$ boyutlu *m*

adet büyük eleman verilsin. Optimal çözüm, her bidonda bir küçük ve bir büyük eleman olmak üzere *m* adet bidon gerektirir.

Online algoritma, elemanları verilen sırada yerleştirmeye başlar ve ilk *m* adet küçük elemanı *m/2* bidona ve devam eden *m* adet büyük elemanı *m* bidona yerleştirir. Sonuç olarak da *3m/2* bidona ihtiyaç duyulur.

Verilen örnek için offline algoritma kullanılsaydı, elemanlar artmayan sıraya göre yerleştirilecek ve sonuç *m* bidon bulunarak optimal çözüme erişilecekti. Fakat, her örnekte optimal çözüme ulaşmak yukarıdaki örnekte olduğu kadar açık ve kolay olmayabilir. Çünkü optimal çözümün belirli bir düzeni yoktur.

5. HAT DENGELEME PROBLEMİ

Üretim hatları farklı faaliyetler ve çeşitli parçalar içermektedir. Bir üretim hattı ard arda sıralı yerleştirilmiş iş istasyonlarından oluşmaktadır. Bir üretim hattı operasyonlara bölünür ve her bir operasyon üretim işinin en küçük birimidir. Bu operasyonların birbirinden farklı özellikler göstermesi, farklı sürelerde tamamlanması ve kimilerinin farklı makinelerde yapılması zorunluluğu bu hatların zamanlama ve koordinasyonunu önemli kılar (Güner, 2001; Salveson, 1955).

Hat Dengeleme Problemi (HDP), operasyonların iş istasyonlarına dağıtım problemidir. Operasyonlar iş istasyonlarına gruplandırıldığında, tüm grupların süreleri birbirine eşit ise hat mükemmel dengelenmiştir ve düzgün bir iş akışı gerçekleşecektir. Genellikle her işçi grubun toplam çalışma süresi birbirinden farklı olabilmektedir. Fakat gruplar eşit çalışma sürelerine sahiptir. Bu nedenle bazı gruplarda boş süre kalacak, bazı grupların ise çalışma süreleri yeterli gelmeyecektir. Boş süre işletmeler için kayıp zamandır ve çalışanların motivasyonunu bozduğu için minimum seviyede olmalıdır. Ancak sıfır boş süre ile hat dengeleme yapmak da hemen hemen imkansızdır (Güner, 2001; Boysen et al., 2008; Scholl, 1999; Salveson, 1955).

5.1 Planlama

Üretim için gerekli makine, yardımcı alet ve teçhizatın istenilen anda ve yeterli miktarda hazır olma çalışmaları planlamanın içeriğine dahildir. Aynı şekilde yeterli sayıda ve istenilen nitelikte işgücünün, gerektiği zaman gereken yerde bulundurulması da önemli bir planlama çalışmasıdır (Acar, 1998; Güner, 2001).

5.1.1 Üretim aracı ve personelin planlanması

Üretim hatlarında üretim araçlarının ekonomik olarak kullanımını sağlayabilmek için, hangi görevlerin, hangi üretim araçlarında yapılacağı üretim aracının özelliklerine uygun olarak planlanmalıdır. Üretim aracı kullanım planlaması ve yönetiminde göz önünde alınacak ölçütler çok yönlüdür. Üretim alanlarında kullanılabilecek araçlar, teknik performans güçlerine göre, hem tüm görevlerin elden geldiğince termin ve kaliteye uygun olarak yapılması, hem de kendilerinden mümkün olan en yüksek ölçüde yararlanılması sağlanacak biçimde yüklenilmelidir. Üretim aracı kullanımının planlanmasına paralel olarak

yürütülecek personel görevlendirme planlaması, personel gereksinimi mevcut personelin planlarından yola çıkar. Personel görevlendirme yönetimi ise kapasite planlama ve yüklemeye dayanır ve iş dağıtımı ile beraber yürütülür. Personel görevlendirme yönetimi, daha önceden belirlenmiş personel ve üretim aracı gereksinimi ve her çalışma noktasında gerçekleştirilmesi gereken üretim miktarından insana yönelik olarak doğan istekler temel alınarak yürütülür. Bir çalışma yerindeki gerek ve yüklenmeler, çalıştırılan kişinin performans sunusuna olabildiğince uygun düşmelidir. Üretim aracının iyi dengelenmesi işçilerin yüklerinin de iyi dengelendiği anlamına gelmez. Bazı hallerde, üretim aracının yanı sıra işçilerin yüklerini de ayrıca dengelemek gerekebilir. Özellikle basit makinelerin yer aldığı ve işçilik maliyetinin yüksek olduğu sistemlerde çeşitli yöntemlerle işçilik sürelerinin dengelenmesine çalışılır. İşçilik dengesinin tam olması, üretim hattındaki her işçiyi normal çalışma süresinin tamamında çalıştırmak anlamına gelir. Hiç kuşkusuz bu ideal durumdur ve pratikte amaç ideale mümkün olduğu kadar yaklaşmaktır (Acar, 1998; Starr, 1989; Güner, 2001).

5.1.2 İşletmenin planlanması

Bir konfeksiyon işletmesi ambarına gelen kumaş, yardımcı malzeme ve aksesuarların bir ürün halini almasına kadar işletmenin değişik bölümlerinde sırasıyla bir takım işlem kademelerinden geçmesi söz konusudur. İşletme bölümleri kesim, dikim, ütü, kalite kontrol, ambalaj, şeklinde sıralanmakta ürün toplam üretim süresinin büyük bir kısmını dikim bölümünde geçirmektedir. Dolayısıyla işletme personelinin büyük bir çoğunluğu da (%60 – 70) bu bölümde çalışmaktadır. İşletme üretimi iyi bir şekilde organize ve kontrol edebildiği, süreyi işletmenin teknik imkanlarının el verdiği ölçüde kısaltabildiği ve yoğun personeli iyi bir şekilde yönetebildiği takdirde kar elde edebilme şansını yakalar, tersi durumda zarar etme ihtimali yüksektir. Bu nedenle konfeksiyon işletmelerinde üretim şeması ve hat dengeleme çalışmaları genellikle dikimhane bolümü için yapılan çalışmalardır (Acar, 1998; Starr, 1989; Güner, 2001).

5.2 Dikim Hattının Dengelenmesi

Hat dengelemede ayrıntılı bir çalışma yaparak akıcı bir üretim için boş zamanların ortadan kaldırılması ve işin çalışma noktalarına hemen hemen eşit olarak taksim edilmesi amaçlanır. Konfeksiyon işletmelerinde personelin büyük

bir çoğunluğu dikim bölümünde çalışmaktadır. Bu nedenle hat dengeleme çalışmaları genellikle dikim bölümü için yapılır.

Hat dengelemede her işlem kademesi için gerekli sürelerin tespit edilmesi, bu işlemleri çok yakından tanıyan bir kişi tarafından yapılmalıdır. Bir işlemin süresi işlemin başlaması ve bitmesi arasında geçecek toplam zamandır ve standart birim süre olarak ifade edilir. Hat dengeleme yapabilmek için öncelikle belirlenmesi gereken işlemler şunlardır.

1- Ürünün işlem kademeleri (Üretim kademelerinin net olarak düzenlenmesi işlem sırasının net olarak görülmesi açısından faydalıdır).

2- Her işlem kademesinin standart birim süresi

3- Her işlem kademesinde kullanılması gereken makineler

4- Günlük çalışma süresi

5- Üretilecek modelin günlük üretim miktarı

Dikim hattının dengelenmesinde işlemler direkt olarak işçilere verilmemektedir. Bir işlemin standart zamanı ile o anki üretim miktarının çarpımı sonucu, o işlemin o üretim miktarı için günlük gerekli süresi belirlenir. Bir işlemin günlük gerekli süresinin, bir işçinin günlük çalışma süresinden büyük olması dengeleme işleminde olası bir durumdur. Bundan dolayı hat dengelemede işlemleri işçi gruplarına atamak daha makul bir yaklaşımdır. Herhangi bir işlemden istenen miktarda üretebilmek için günlük gerekli sürenin, bir grubun günlük çalışma süresini doldurmadığı durumda (boş zamanları olduğu takdirde), bir işçilik, iki işçilik veya üç işçilik guruplara, üretim şemasının akışına paralel olarak uygun işlemler eklenebilir.

Anlaşıldığı gibi hat dengelemede tüm hesaplar işlem kademelerinin standart birim süreleri üzerine oturtulmuş durumdadır. Standart birim süreler standart tempo (%100) hesabına göre tespit edildiğinden işçilerin %100 tempo ile çalıştığı varsayılmaktadır. Eğer işçi kendisine tanınan ve işçi niteliğine göre %5 ile %15 arasında değişen zaman paylarını kullanır ve/veya çalışma temposunu %100 olarak tanımlanan standart temposunun üstüne çıkarabilirse kendisinden beklenen sayıdan daha fazla üretim gerçekleştirebilir. Bütün bunların tersi olduğu durumlarda

beklenen sayısının altında bir üretim de çıkabilir. Bir diğer olasılık da işten kaynaklanan bir takım nedenlerle (hatalı kesim, iş beslemesinin aksaması, makine arızaları, işçinin yapması gereken işlemden bir önceki kademedeki elemanın ağır çalışması vs.) planlanan üretimin gerçekleşmemesi ihtimalinde elemanın boş kalması daha sonra zincirleme olarak diğer elemanlarında boş kalması söz konusu olacaktır. Bu tür problemlerin yaşanmasının önlenmesi için işletmedeki mühendis, tekniker, ustabaşı, usta, ayakçı vs. elemanların işçilerini iyi tanımaları, doğru görevlendirme yapmaları, sıkı takip, kontrol ve acil önlemler almaları, kumaş ve malzeme beslemesini aksatmamaları gerekir. Dikkatli ve önceden gerçekleşmiş birtakım fiili değerler göz önüne alınarak uygulamaya yönelik hazırlanmış bir hat dengeleme yürütücüler açısından büyük bir yol göstericidir (Acar, 1998; Starr, 1989; Güner, 2001; Scholl, 1999; Salveson, 1955).

5.3 Yalın Üretim

Yalın üretim, üretime yük getiren tüm israflardan arınmayı hedef alan bir yaklaşımdır. Diğer bir deyişle, yalın üretim, yapısında hiçbir gereksiz unsur taşımayan ve hata, maliyet, stok, işçilik, geliştirme süreci, üretim alanı, fire, müşteri memnuniyetsizliği gibi unsurların en aza indirgendiği üretim sistemi olarak da tanımlanabilir. Yalın üretimin ana stratejisi hızı artırıp, akış süresini azaltarak kalite, maliyet ve teslimat performansını aynı anda iyileştirmektir.

Süreç içinde değişik operasyonları kapsayan tüm üretimlerde, operasyon sürelerinin farklı olması, üretim akışı içinde gerek işçilerin, gerek de bitmiş ve işlenmekte olan parçaların beklemeyle vakit kaybetmelerine neden olabilir. İşlenmekte olan parçaların beklemesi demek, bir parçanın bir işlenme aşamasından diğerine hemen geçmemesi demektir, stoklu çalışmada işler zorunlu olarak bu şekilde yürümektedir. Yalın üretim stok tutma gereksinimini ortadan kaldırıp, düşük maliyetli yüksek kalitede üretimi mümkün kılmayı hedefler. Bu noktada, yalın üretim modelinin istenilen sonuçları vermesi tam zamanlı üretim için yalın üretim unsurlarına bağlıdır (Holweg, 2007).

5.3.1 Tam zamanlı üretim

Tam zamanlı üretim, üretimin herhangi bir aşaması için gerekli olan girdilerin, bir önceki üretim noktasından veya dışarıdan ihtiyaç duyulduğu anda, gerekli miktarda ve kalitede sağlanmasıdır. Yalın üretimde girdilerin böyle bir yapı içerisinde tedarik edilmesi, stokla çalışma ihtiyacını ortadan kaldırırken stok

maliyetini ve beraberinde işçilik maliyetini düşürmektedir. Ayrıca tam zamanlı üretim yöntemi, girdilerin kullanılma zamanı ve miktarı konusunda esneklik sağladığı için üretimin değişen talep miktarına göre düzenlenebilmesine de olanak sağlamaktadır (Güner, 2001; Kanat and Güner, 2006).

5.3.2 Yalın üretim unsurları

Düşük maliyetli, yüksek kaliteli, tam zamanlı üretimin gerçekleştirilebilmesi yalın üretim unsurlarının işletme ve tedarikçiler tarafından uygulanmasına bağlıdır. Bu unsurlardan biri olan hat dengelemenin yine yalın üretimin ilgili unsurları dikkate alınarak düzenlenmesi gerekir. Bu unsurlardan bazıları; iş gören, modüler üretim, makineler ve atölyeler arası senkronizasyondur (Holweg, 2007).

5.3.2.1 İş gören

Yalın üretim modelinde işçi sahip olduğu vasıf, bilgi, tecrübe ve sorumluluk ile üretim sürecinin gerçek yöneticisi olarak kabul edilmektedir. Yalın üretim, pazardan gelebilecek hedefleri anında karşılayabilmek için tepe yönetimden işçisine ve yan sanayicisine kadar herkesin çalışmasını bir bütün olarak birleştirir. Yalın üretimde işçi ve parça beklemelerinin neden olduğu, stoklu çalışmayı engellemek için işçiler ardı ardına gelen operasyonları yapabilir ve otomasyon düzeyi yüksek makineleri kullanabilir vasıftadır. Böylelikle işçinin değişik nitelik taşıyan işler yapması (işin zenginleştirilmesi) bekleme nedeniyle iş yapmadan geçen zamanının azaltılmasına, işçinin zihinsel ve bedensel yeteneklerinin kullanılmasına yol açmaktadır (Güner, 2001).

5.3.2.2 Modüler üretim

Yalın üretimde, modül üretimi uygulanarak, genel amaçlı makinelerle ürün farklılaştırılması yapılabilmekte, aynı makinelerle farklı mallar üretilerek değişen talebe uyum sağlanabilmektedir. Konfeksiyonda bir modülde bir model üretimi yapılabildiği gibi, fazla sayıda operasyonu olan modeller için belirli prensipler ile bazı operasyonları içeren modüller hazırlanabilir. Bu durumda tüm modüller senkronize çalışmalıdır (Güner, 2001).

5.3.2.3 Makineler ve atölyeler arası senkronizasyon

Yalın üretim stoklu çalışmayı ortadan kaldırmayı amaçlar. Buna yönelik olarak herhangi bir atölye içinde bir parçanın nihai halini alması için gereken tüm makinelerin, parçaların işlenme akışına dayanarak birbiri ardı sıra yerleştirilmeleri ve parçanın bir önceki süreçte kullanılan makineden bir sonraki süreçte kullanılacak makineye hiç beklemeden geçmesi gerekir. Makinelerin bu şekilde yerleştirilmelerine "süreç-bazlı yerleşim" ya da "süreç-bazlı hat", ve parçaların süreçler arasında beklemeden teker teker aktarılmalarına da "tek-parça akışı" denilmektedir (Güner, 2001).

Tek-parça akışının gerçekleştiği süreç-bazlı hat, stoğun sıfırlanması ya da mümkün olduğunca küçük miktarda tutulması için geliştirilmiş en etkin sistemlerden biridir. Süreç-bazlı hatların gerçekten etkin olabilmeleri için, aynı hattı oluşturan makinelerin çalışma tempoları ya da kapasitelerinin, yani bir işlemi tamamlamaları için gereken sürelerin de denkleştirilmeleri gerekir. Örneğin, hattaki bir önceki makinenin parçayı işleme süresi 1 dakika, sonrakinin süresi 4 dakika ise, bir sonrakinin tek bir parçayı işleme süresinde, bir önceki 4 parça birden işleyecek ve eğer makineler durmadan çalışırlarsa, sonraki makinenin yanında öncekinden gelen parçalar giderek artan miktarlarda birikmeye başlayacaklardır. Bu durumda beklemesiz üretim olan tek-parça akışı gerçekleşemeyecektir. Yalın üretimde bu sorun, hattaki makineleri birbirine senkronize ederek, yani tüm makinelerin aynı süre içinde aynı miktarda parça işlemeleri sağlanarak çözülmüştür (Güner, 2001).

Parçaların hat ya da makine yanı stokta beklememesinden elde edilecek kazanç, aslında makinelerin tam kapasite çalışmalarından elde edilecek kazançtan daha büyüktür. Yalın üretimde parçaların beklemesi, yani stoklu çalışma, olabilecek en büyük israftır ve sistem neredeyse tümüyle israfın önlenmesi üzerine kuruludur (Güner, 2001).

5.4 Hat Dengeleme Problemi Modelleri

HDP nin materyali için konfeksiyon işletmesi dikim bölümü seçilmiştir. Konfeksiyon işletmesi seçilmesinin, nedeni modaya bağlı olan bu sektörde model değişiminin çok fazla ve ürünlerin sezonluk olması nedeni ile tam zamanında üretimin önemli olmasıdır. Dikim bölümünün seçilme nedeni ise, bu bölümde işin niteliği ile ilgili olarak çok fazla personel çalışması sebebiyle motivasyonun önemli olması ve dengeli bir iş dağılımının yapılması, boş zamanların mümkün olduğunca az olması zorunluluğudur. HDP'de her farklı üretim miktarı için farklı bir planlama yapılmalıdır. Ancak farklı durumların sayısı çok fazla olabilir.

BPP, planlama problemlerinin ters çevrilmiş şeklidir. Belirli bir sayıdaki işçinin, verilen bir işi, bazı sınırlamalar uyarınca, en çabuk nasıl tamamlayacağı yerine; bir işi, belirli bir sürede bitirmek için en az kaç kişinin gerektiği sorusu sorulmaktadır. Bu soruda işçiler bidonların, iş öğeleri de paketlerin yerini alır.

HDP'nin çözümü için BPP'nin çözüm ilkeleri ele alınmıştır. Problemde belirli sayıdaki üretim miktarında operasyonların bazı sınırlamalar uyarınca belirli bir sürede bitirilmesi için en az kaç kişinin gerekli olduğu, çalışanlara hangi operasyonların verileceği ve sonuçta ne kadar boş süre kalacağı veya ne kadar süre fazla çalışması gerektiği sorusu sorulmaktadır (Gürsoy, 2009).

5.4.1 Esnek yalın üretim hat dengeleme problemi

Varsayalım ki, bütün iş s modülden oluşsun $\mathrm{A}=\{\mathrm{A}_1,\mathrm{A}_2,...,\mathrm{A}_s\}$ ve bu modüller kendi aralarında kısmi sıralı olsun.

Her bir A_q modülü $n^{(q)}$ sayıda operasyondan oluşur, yani $\mathrm{A}_q=\{a_1^{(q)},a_2^{(q)},...,a_{n^{(q)}}^{(q)}\}$. Bu operasyonlar arasında da kısmi sıralama vardır, yani;

$$a_1^{(q)} \prec a_2^{(q)} \prec ... \prec a_k^{(q)}, a_{k+1}^{(q)} \prec a_{k+2}^{(q)} \prec ... \prec a_r^{(q)}, a_{r+1}^{(q)}, a_{r+2}^{(q)} \prec a_{r+3}^{(q)} \prec ... \prec a_{n^{(q)}}^{(q)} \quad (5.4.1.1)$$

Bu ifadeyi şu şekilde yorumlayabiliriz. İlk k operasyon için normal sıralama vardır, yani her bir önce gelen operasyon sonra gelenlerden önce yapılmalıdır. $a_{k+1}^{(q)}$ operasyonu ise ondan sonra gelen operasyonlardan önce yapılmalıdır (sola doğru esnek), ama ondan önce gelen operasyonlar arasında hiçbir sıralama yoktur, yani $a_{k+1}^{(q)}$ operasyonu $a_1^{(q)},a_2^{(q)},...,a_k^{(q)}$ operasyonlarının her

birinden önce yapılabilir. $a_{k+2}^{(q)}, a_{k+3}^{(q)}, ..., a_r^{(q)}$ operasyonları da normal sıralıdır, $a_{r+1}^{(q)}$ operasyonu ise tam serbesttir, yani yukarıdaki (5.4.1.1) sıralamasında herhangi bir yerde yapılabilir. $a_{r+2}^{(q)}, a_{r+3}^{(q)}, ..., a_{n^{(q)}}^{(q)}$ operasyonları ise normal sıralıdır.

Böylece sıralamada her hangi bir operasyondan önce (veya sonra) gelen virgül, bu operasyonun ondan önce (veya sonra) gelen operasyonlara göre esnek olduğunu bildirir. Örneğin, yukarıdaki (5.4.1.1) sıralamasında $a_{k+1}^{(q)}$ operasyonu ondan önce gelen k operasyonuna göre esnektir, sonra gelenlere göre ise esnek değil, ama $a_{r+1}^{(q)}$ operasyonu hem ondan önce, hem de ondan sonra gelen operasyonlara göre esnektir.

$t_j^{(q)}$ ile $a_j^{(q)}$ operasyonunun yapılması için gereken *zamanı* gösterelim, yani $\left\{t_1^{(q)}, t_2^{(q)}, ..., t_{n^{(q)}}^{(q)}\right\}$'ler $\left\{a_1^{(q)}, a_2^{(q)}, ..., a_{n^{(q)}}^{(q)}\right\}$ operasyonlarının yapılma süreleri olsun.

Operasyonlar belirli gruplara atanmalıdır. Bu gruplardaki işçilerin sayısı $1,2,...,l$ tane olabilir. Burada $l \le 3$ olarak ele alınacaktır ve amaç operasyonları gruplar arasında öyle paylaştırmaktır ki, işçilerin toplam boş zamanı en küçük olsun. Her bir işçi günde en fazla $(T + ot)$ birim süre çalışmalıdır. Burada T mesai süresini, ot ise fazla mesai süresini temsil etmektedir. Üründen p sayıda üretilmesi gerekiyorsa $a_j^{(q)}$ operasyonu $p \cdot t_j^{(q)}$ zaman alacaktır.

Bu problemi matematiksel olarak BPP gibi ele alabiliriz. Burada işlemler eşyalara, gruplar ise bidonlara uygun gelir. Ama BPP'de bidonların kapasiteleri aynı olmasına rağmen burada farklı kapasiteli bidonları ele alacağız. Şöyle ki; eğer grup 1 kişiden oluşuyorsa ona uygun bidonun kapasitesi $(T + ot)$ olacak, grup $l \quad (l > 1)$ kişiden oluşuyorsa kapasite $l \cdot (T + ot)$ olacaktır, yani;

$$(l-1)(T+ot) < p \cdot t_j^{(i)} \le l\,(T+ot) \quad \Rightarrow \quad l \quad \text{operatör} \qquad (5.4.1.2)$$

BPP'de amaç bidonların sayısını en küçük yapmaktır. Burada ise tüm bidonlardaki toplam boş yeri en küçük yapmaktır. Tabi ki buradan bidonların sayısının en küçük olması anlamı da çıkacaktır.

Bilinen BPP'de eşyaların hiçbir önceliği yoktur, ele aldığımız problemde ise operasyonların (5.4.1.1) sıralamasına göre kısmi önceliği vardır. İncelediğimiz problemin bir başka özelliği de bidonların kapasitelerinin t'den dolayı esnek olmasıdır.

Aşağıdaki gibi $x_{ij}^{(q)}$ değişkenlerini tanımlayalım:

$$x_{ij}^{(q)} = \begin{cases} 1, \text{ eğer } a_j^{(q)} \text{ operasyonu } i^{(q)}. \text{ gruba atanırsa} \\ 0, \text{ aksi halde} \end{cases} \tag{5.4.1.3}$$

$$X_i^{(q)} = \left(x_{i1}^{(q)}, x_{i2}^{(q)}, ..., x_{in^{(q)}}^{(q)}\right),\ X^{(q)} = \left(X_1^{(q)}, X_2^{(q)}, ..., X_{m^{(q)}}^{(q)}\right),\ X = \left(X^{(1)}, X^{(2)}, ..., X^{(s)}\right)$$

Varsayalım ki; algoritmanın çalışması sonucu q. modül için $m^{(q)}\left(X^{(q)}\right)$ sayıda grup oluşturulmuş olsun ve $i^{(q)}$. gruptaki bidonların sayısını $l_i^{(q)}\left(X_i^{(q)}\right)$ işaretleyelim. Yazılışı kolaylaştırmak bundan sonra $m^{(q)}(X^{(q)})$ ve $l_i^{(q)}\left(X_i^{(q)}\right)$'yi $m^{(q)}$ ve $l_i^{(q)}$ olarak göstereceğiz.

(5.4.1.4) q. modül için $i^{(q)}$. gruptaki çalışma süresini, (5.4.1.5) $i^{(q)}$. gruptaki boş zamanı, (5.4.1.6) q. modül için boş zamanı ve (5.4.1.7) toplam boş zamanı gösterir. (5.4.1.8) q. modüldeki en yoğun çalışan grubun ortalama çalışma zamanı ve (5.4.1.9) tüm modüller arasında en yoğun olan grubun çalışma zamanıdır.

$$T_i^{(q)}\left(X_i^{(q)}\right) = \sum_{j=1}^{n^{(q)}} t_{ij}^{(q)} x_{ij}^{(q)}, \tag{5.4.1.4}$$

$$\Delta T_i^{(q)}\left(X_i^{(q)}\right) = (T + ot) l_i^{(q)} - T_i^{(q)}\left(X_i^{(q)}\right) \tag{5.4.1.5}$$

$$\Delta T^{(q)}\left(X^{(q)}\right) = \sum_{i=1}^{m^{(q)}} \Delta T_i^{(q)}\left(X_i^{(q)}\right) \tag{5.4.1.6}$$

$$\Delta T(X) = \sum_{q=1}^{s} \Delta T^{(q)}\left(X^{(q)}\right) \tag{5.4.1.7}$$

$$MT^{(q)}\left(X^{(q)}\right) = \max_{i=\overline{1,m^{(q)}}} \left\{T_i^{(q)}\left(X_i^{(q)}\right) / l_i^{(q)}\left(X_i^{(q)}\right)\right\} \tag{5.4.1.8}$$

$$MT(X) = \max_{q=\overline{1,s}} \left\{MT^{(q)}\left(X^{(q)}\right)\right\} \tag{5.4.1.9}$$

Bu notasyonu kullanarak problemin modelini aşağıdaki gibi yazabiliriz:

$$\sum_{j=1}^{n^{(q)}} t_j^{(q)} x_{ij}^{(q)} \le (T+ot) l_i^{(q)},\ i=1,2,...,m^{(q)},\ q=1,2,...,s \qquad (5.4.1.10)$$

$$\sum_{i=1}^{m^{(q)}} x_{ij}^{(q)} = 1,\ j=1,2,...,n^{(q)};\ q=1,2,...,s \qquad (5.4.1.11)$$

$$\sum_{j=1}^{n^{(q)}} x_{ij}^{(q)} \ge 1,\ i=1,2,...,m^{(q)},\ q=1,2,...,s \qquad (5.4.1.12)$$

$$\sum_{i=1}^{m^{(q)}} \sum_{j=1}^{n^{(q)}} x_{ij}^{(q)} = n^{(q)} \qquad (5.4.1.13)$$

$$x_{ij}^{(q)} = 0 \vee 1 \qquad i=1,2,...,m^{(q)},\ j=1,2,...,n^{(q)},\ q=1,2,...,s \qquad (5.4.1.14)$$

Bu kısıtlar altında tüm gruplardaki boş zamanların en küçüklenmesi istenir, yani;

$$\underset{X}{Enk}\ \ \Delta T_0 = \Delta T(X) \qquad (5.4.1.15)$$

Bu modelde, (5.4.1.10) her bir gruba atanan operasyonların o grubun zaman kapasitesini aşmayacağını, (5.4.1.11) her bir operasyonun yalnız bir gruba atanabileceğini, (5.4.1.12) her grupta en az bir operasyonun yapılacağını ve (5.4.1.13) açıkta operasyon bırakılmayacağını kontrol eder. Böylece problemin matematiksel modeli (5.4.1.10 - 5.4.1.15) şeklinde olur (Nuriyev et al., 2008, 2009; Gürsoy, 2009).

Farklı modüller arasında senkronizasyon yaratmak için verilen P^* planına göre her bir modül p'nin $P^* - \tilde{p} < p < P^* + \tilde{p}$ aralığındaki değerler için (burada $\tilde{p}$ esneklik sabitidir) çözülerek bütün modüller için en uygun olan $\hat{P}$ değeri karar verici tarafından seçilir ve buna uygun olan $MT(X)$ belirlenir. Bu değer (5.4.1.5) ve (5.4.1.10) formüllerinde, $(T+ot)$'nin yerine yazılarak, belirlenmiş atama için toplam boş zaman kesinleştirilir (Nuriyev et al., 2008, 2009; Gürsoy, 2009).

Bu şekilde tanımladığımız BPP'yi çözmek için, First Fit algoritmasının ilkelerini aşağıdaki şekilde ayarlayabiliriz (Gürsoy, 2009):

1) Bidon paketleme işlemine öncelikle sıralamanın önemli olduğu operasyonlardan başlanır.

2) Paketleme esnasında, kullanılan bidonun kapasitesi dolduğunda yeni ve boş bir bidondan paketlemeye devam edilir.

3) Eğer tek başına, bir bidonun kapasitesini aşan paketlerle karşılaşılırsa paketin büyüklüğüne göre bidonun kapasitesi l $(l > 1)$ katına çıkarılır. Kapasitesi arttırılan bidonlar için paketleme işlemi, bidonun yeni kapasitesi üzerinden ve operasyonların sırası değişmeyecek şekilde devam eder. Kapasitesi arttırılmış bidon dolduktan sonra, kapasitesi başlangıçtaki gibi $(T + ot)$ olan yeni ve boş bir bidona geçilir.

4) Sıralamanın önemli olduğu operasyonların paketlenmesi bittikten sonra sıralamanın önemsiz olduğu (esnek) operasyonlar için paketlemeye geçilir.

5) İlk bidondan son bidona kadar, bidonlar sırayla kontrol edilir ve sıradaki esnek operasyon uygun ilk bidona yerleştirilir.

6) Eğer sıradaki esnek operasyon mevcut hiç bir bidona yerleştirilemez ise ilgili operasyonun boyutuna bağlı olarak yeni bir bidon oluşturulur.

7) Eğer esnek olan tüm operasyonlar da paketlendiyse bidon paketleme işlemi bitirilir.

5.4.2 Esnek yalın üretim hat dengeleme problemi – yeni bir model ve sezgisel algoritma

Esnek yalın üretim hat dengeleme probleminde değişken piyasa taleplerine hızla cevap verebilecek yeni bir yaklaşım oluşturulmuştur.

Dikim bölümünde hat dengelemesi yapılacak ürünün operasyonlarından bazıları esnek atanabilme özellikleri içerebilmektedir, bazıları ise mevcut sıralamalarının dışına çıkamazlar. Hattı dengelenecek üründeki operasyon bilgileri doğrultusunda, esnek bir üretim aralığı kullanarak operatör başına düşen en az boş zamanı arayan bir tamsayılı matematiksel model geliştirilmiştir (Nuriyev et al., 2009). Bu model geliştirilirken BPP terminolojisi kullanılmıştır. Üretim hattındaki operasyonlar esneklikleri dikkate alınarak öyle atanmalıdır ki operatör başına kalan zaman en küçük olmalıdır.

Bunun yanında, HDP'nin NP-Zor sınıfta olmasından dolayı, piyasa taleplerine anında cevap verebilecek, polinomiyal karmaşıklığa sahip, minimum operatör sayısını belirleyen ve operatör başına minimum boş zamanı bulan ve BPP tabanlı yeni bir sezgisel algoritma tasarlanmıştır. Tasarlanan yeni algoritma ile endüstride kullanılabilecek C# dilinde bir yazılım programlanmış ve bu

yazılım yardımıyla örnek bir model için elde edilen yüksek verimlilikli dengeleme sonuçları sunulmuştur (Gürsoy, 2012).

5.4.2.1 Esnek yalın üretim hat dengeleme problemi için tamsayılı matematiksel model

Farz edelim ki, üretim hattında esnek ve esnek olmayan toplam *n* operasyon içeren bir model olsun. Esnek olmayan operasyonlar ard ardına, esnekler ise herhangi bir noktaya yerleştirilebilir olsun.

Üretim sürecinde t_j, j. operasyonun standart birim süresini, $T + ot$ bir operatörün günlük en fazla çalışma süresini göstersin. Burada, T günlük çalışma süresi, ot günlük fazla mesai süresidir. Üretim sürecinde, j. operasyon p adet ürün için $p \cdot t_j$ dakika zaman sürmektedir.

Tablo 5.1 Esnek yalın üretim hat dengeleme problemi için örnek bir modelin operasyon detayları

No	Operasyon adı	Birim zaman (cdk.)	Makine adı	Esneklik
1	Sason Dikme	88	Düz dikiş	Yok
2	Cep yeri tela yapıştırma	10	Ütü	Yok
3	Flatoya tela yapıştırma	8	Ütü	Var
4	Flato ve karşılığı overlok	38	3 iplik overlok	Var
5	Flato dikme	90	Düz dikiş	Yok
6	Flato çentik atma	56	Elde	Yok
7	Flato uç tutturma ve gaze	130	Düz dikiş	Yok
8	Arka cep bitirme	78	Düz dikiş	Yok
9	Arka cep torbalama	24	5 iplik overlok	Yok
10	Arka orta kapama overlok	37	5 iplik overlok	Yok
11	Arka orta gaze	39	Düz dikiş	Yok

Üretim sürecindeki tüm operasyonlar iş istasyonlarında yapılır ve iş istasyonları operatörler tarafından işletilir. Bazı üretim miktarlarında, operasyonların üretim süreleri bir operatörün günlük çalışma süresinden fazla olabilmektedir. Bu tip operayonları tamamlayabilmek için ikinci veya gerekli ise üçüncü operatör ilgili operasyonların tamamlanmasında görevlendirilebilir. Bu

nedenle, (5.4.1.2) eşitsizliğinden yararlanarak hat dengeleme sürecinde operatör ifadesi yerine, bir, iki veya üç operatör içerebilen işçi grupları kullanılacaktır.

(5.4.1.2) eşitsizliği, j. operasyon için en az kaç operatöre ihtiyaç duyulacağını belirler ve piyasa koşullarına uygun olması için üç işçi ile üstten sınırlandırılmıştır (Nuriyev et al., 2009).

$X=\left(X_1,X_2,...,X_m\right)$ için m işçi gruplarının sayısını, $i=\overline{1,m}$ olmak üzere $X_i=\left(x_{i_1},x_{i_2},...,x_{i_n}\right)$ için n operasyon sayısını ve s de esnek olmayan operasyonların sayısını göstersin.

Girdi parametreleri:

$t_j\in\mathbb{Z}^+$ ($j=\overline{1,n}$), j. operasyonun standart birim süresi,

$l_i\in\mathbb{Z}^+\cup\{0\}$ ($i=\overline{1,m}$), i. işçi grubundaki operatör sayısı,

$e_j\in\mathbb{Z}^+$ ($j=\overline{1,s}$), esnek olmayan j. operasyonun indisi,

$T\in\mathbb{Z}^+$, günlük çalışma süresi ve

$ot\in\mathbb{Z}^+\cup\{0\}$, günlük fazla mesai süresidir.

Karar değişkeni:

$$x_{ij}=\begin{cases}1, & \text{eğer } j\text{. operasyon } i\text{. işçi grubuna atandıysa}\\ 0, & \text{aksi halde.}\end{cases},\quad i=\overline{1,m},\ j=\overline{1,n}$$

Bu notasyonu kullanarak ve genellikten uzaklaşmadan $j=\overline{1,n}$ için $p\cdot t_j\le 3\cdot(T+ot)$ ve $i=\overline{1,m}$ için $0\le l_i\le 3$ kabul edelim

$$C_i=\sum_{j=1}^{n}p\cdot t_j\cdot x_{ij},\qquad i=\overline{1,m} \tag{5.4.2.1}$$

$$\Delta C_i=l_i\cdot(T+ot)-C_i,\quad i=\overline{1,m} \tag{5.4.2.2}$$

$$\Delta C=\sum_{i=1}^{m}\Delta C_i \tag{5.4.2.3}$$

$$\Delta ID = \Delta C \Big/ \sum_{i=1}^{m} l_i \tag{5.4.2.4}$$

Yukarıda, (5.4.2.1) *i*. işçi grubunun iş yükünü, (5.4.2.2) *i*. işçi grubunun boş zamanını, (5.4.2.3) üretim sürecindeki toplam boş zamanı ve (5.4.2.4) üretim sürecindeki operatör başına boş zamanı verir. Bu ifadeleri kullanarak, esnek operasyonlar içerebilen Esnek Yalın Üretim HDP'nin yeni bir Tamsayılı Matematiksel Modeli aşağıdaki gibi geliştirilmiştir (Gürsoy, 2012):

$$\underset{X}{Enk}\ \Delta ID \tag{5.4.2.5}$$

Kısıtlar:

$$\sum_{j=1}^{n} p \cdot t_j \cdot x_{ij} \le l_i \cdot (T + ot), \qquad i = \overline{1,m} \tag{5.4.2.6}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = \overline{1,n} \tag{5.4.2.7}$$

$$\sum_{j=1}^{n} x_{ij} \ge 1, \qquad i = \overline{1,m} \tag{5.4.2.8}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = n \tag{5.4.2.9}$$

$$l_i = \left\lceil \underset{j}{enb} \left\{ p \cdot t_j \cdot x_{ij} \Big/ (T + ot) \right\} \right\rceil, \quad i = \overline{1,m} \tag{5.4.2.10}$$

$$i \cdot x_{ie_j} - k \cdot x_{ke_{j-1}} > 0, \qquad i = \overline{1,m-1}, k = \overline{i+1,m}, j = \overline{2,s}, x_{ie_j} = 1 \tag{5.4.2.11}$$

$$x_{ij} = 0 \vee 1, \qquad i = \overline{1,m}, j = \overline{1,n} \tag{5.4.2.12}$$

$$l_i \in \mathbb{Z}^+ \cup \{0\}, \quad i = \overline{1,m} \tag{5.4.2.13}$$

Bu modelde, (5.4.2.6) kısıtı gruba atanmış operasyonların grubun çalışma zamanını geçemeyeceği, (5.4.2.7) kısıtı her operasyonun sadece tek bir grupta yapılabileceğini, (5.4.2.8) kısıtı sürece dahil olan her grubun en azından bir operasyonu işletmesini, (5.4.2.9) kısıtı dengeleme sürecindeki tüm operasyonların

tamamlanmasını kontrol eder. (5.4.2.10) kısıtı *i*. gruptaki işçi sayısını belirlerken (5.4.2.11) kısıtı esnek olmayan kısıtların doğru sıralamada işletilmesini denetler. (5.4.2.6 - 5.4.2.13) kısıtları altında, amaç (5.4.2.5) operatör başına boş zamanı en aza indirmektir.

En verimli üretim miktarını bulmak için modelin $\tilde{p}$ esnek üretim miktarı sabiti olmak üzere $\left[p-\tilde{p}, p+\tilde{p} \right]$ aralığında çalıştırılması gereklidir (Gürsoy, 2012).

5.4.2.2 Esnek yalın üretim hat dengeleme probleminin NP-tamlığı

Teorem 5.4.2.1: Esnek yalın üretim hat dengeleme problemi NP-tam'dır.

İspat: Esnek yalın üretim hat dengeleme probleminin NP-tamlığını ispatlayabilmek için bir NP-tam problem olan BPP'ye (Bölüm 4.2) kısıtlayacağız.

Esnek yalın üretim hat dengeleme probleminin tanıma versiyonu aşağıdaki gibidir.

Koşul: *n* elemanlı operasyonlar kümesi *U*, *m* elemanlı işçi grubu kümesi *V*, kendi aralarındaki sıralamaya göre yerleştirilmesi gereken esnek olmayan operasyonlar kümesi $E \subseteq U$, üretim miktarı *p*, $t_j \in \mathbb{Z}^+$ $(j=\overline{1,n})$ *j*. operasyonun standart zamanı, $0 \le l_i \le 3$ $(i=\overline{1,m})$ *i*. işçi grubunun operatör sayısı, günlük çalışma süresi *T*, günlük fazla mesai süresi ot ve $K \in \mathbb{Z}^+$ $(K \le m)$ verilsin.

Soru: Operasyonlar kümesi *U*'nun öyle *K* parçalanması var mıdır ki $i=\overline{1,K}$ için her gruba atanan operasyonların süresi grubun çalışma süresini aşmamış olsun?

Genellikten uzaklaşmadan $l_i = 0 \vee 1$ $(i=\overline{1,m})$ ve tüm operasyonları esnek kabul edelim, yani $E=\varnothing$. Bu kabulden dolayı tüm işçi grupları en fazla bir işçiden oluşabilecek ve günlük çalışma zamanları $T+ot$ olacaktır.

Bu durumda bidon kapasitesi olarak $c=T+ot$ ve eşya ağırlığı olarak $w_j = p \cdot t_j$ $(j=\overline{1,n})$ dikkate alınırsa problem BPP'yr kısıtlanmış olacaktır. Böylece Esnek Yalın Üretim Hat Dengeleme Problemi NP-tam sınıftan olduğu ispatlanmıştır.

5.4.2.3 Esnek yalın üretim hat dengeleme problemi için yeni bir sezgisel algoritma

m işçi gruplarının ve *n* de operasyonların sayısı olsun. hat dengeleme problemini optimal olarak çözmek için sayımlama yöntemiyle kontrol edilmesi gereken n^m olası durum vardır ve şimdiye kadar HDP problemi için optimal çözümü bulabilecek polinomiyal zamanlı bir algortima önerilmemiştir.

Sanayide, yöneticiler talepleri karşılamak için makul zaman sınırları çerçevesinde uygun, kabul edilebilir ve optimale yakın bir çözüm bulmak zorundadır. Bu amaçla 2 ana evreden oluşan bir sezgisel algoritma yaratılmıştır.

İlk olarak, esnek olmayan operasyonlar işçi gruplarına yerleştirilir. P1 olarak adlandırılan bu adım Next-Fit benzerin yapıda oluşturulmuştur. P1, esnek olmayan operasyonları yalın üretim mantığında aralarındaki sıraya göre yerleştirir. P1 algoritmasının karmaşıklığı $O(n)$ dir (Garey and Johnson, 1979; Martello and Toth, 1990; Nuriyev et al., 2008, 2009; Gürsoy, 2012).

P1 Algoritması:

1. $p, n, T, ot, (t_1, t_2, \dots, t_n)$ *öğren.*
2. $i = 1, j = 0, l_i = 1, C_i = 0, U = l_i \cdot (T + ot)$
3. *for* $j = 1$ *to* n *do*
4. *if j esnek değilse*
5. $pt = p \cdot t_j$
6. *if* $pt \geq U$ *then (5.4.1.2) eşitsizliğini kullanarak* l_i *yi belirle ve* $U = l_i \cdot (T + ot)$ *olarak güncelle*
7. *if* $C_i + pt > U$ *then* $i = i + 1,\ l_i = 1,\ C_i = 0,\ U = l_i \cdot (T + ot)$ *ve GOTO 6*
8. $C_i = C_i + pt$
9. *Print* $m, (C_1, C_2, \dots, C_m)$ *and* $(l_1, l_2, \dots, l_m)$

İkinci olarak, P2 adımı ile esnek operasyonlar uygun işçi grubuna atanmalıdır. P2, First-fit benzeri bir algoritmadır ve $O(n \cdot \log n)$ karmaşıklığındadır (Garey and Johnson, 1979; Martello and Toth, 1990; Gürsoy, 2012).

P2 Algoritması:

1. $p, n, m, T, ot, (t_1, t_2, \dots, t_n), (C_1, C_2, \dots, C_m), (l_1, l_2, \dots, l_m)$ *öğren.*
2. *for* $j = 1$ *to* n *do*
3. *if j esnek ise*
4. $pt = p \cdot t_j$
5. *for* $i = 1$ *to* m *do*
6. *if* $C_i + pt \leq l_i \cdot (T + ot)$ *then* $C_i = C_i + pt$ *ve GOTO 2*
7. *else break*
8. $m = m + 1$, *(5.4.1.2) eşitsizliğini kullanarak* l_m *i belirle,* $C_m = C_m + pt$
9. *Print* $m, (C_1, C_2, \dots, C_m)$ *and* $(l_1, l_2, \dots, l_m)$.

Tamsayılı Esnek Yalın Üretim Hat Dengeleme Problemi çözmek için P1 ve P2 algoritmalarını kullanan ana algoritma aşağıdaki gibidir. Bu algoritma, uygun üretim miktarı için minimum ortalama boş zamanı verir:

1. $p, \tilde{p}, n, T, ot, (t_1, t_2, \dots, t_n)$ *öğren.*
2. $\hat{p} \leftarrow p$
3. *For* $p \leftarrow \hat{p} - \tilde{p}$ *to* $\hat{p} + \tilde{p}$ *do*
4. *Call* $\boldsymbol{P1}(p, n, T, ot, (t_1, t_2, \dots, t_n))$
5. $m \leftarrow i$
6. *Call* $\boldsymbol{P2}(p, n, m, T, ot, (t_1, t_2, \dots, t_n), (C_1, C_2, \dots, C_m), (l_1, l_2, \dots, l_m))$
7. $id = \sum_{i=1}^{m} \left((T + ot) \cdot l_i - C_i \right) \Big/ \sum_{i=1}^{m} l_i$ *olarak operatör başına boş zamanı hesapla.*
8. *if yeni minimum bulunduysa then* id_{min} *ve* p_{min} *i belirle*
9. *Print* id_{min}, p_{min}

Bu algoritma $\left[p-\tilde{p}, p+\tilde{p}\right]$ aralığındaki en iyi üretim miktarını arar ve $O\left(\tilde{p}\cdot n\cdot \log n\right)=\left(2\tilde{p}+1\right)\left[O\left(n\cdot \log n\right)+O\left(n\right)\right]$ karmaşıklığa sahiptir (Gürsoy, 2012).

5.4.2.4 Esnek yalın üretim hat dengeleme problemi için yazılım ve hesaplama denemeleri

Bu yazılım yukarıda sunulan sezgisel algoritmayı temel alarak C# programlama dilinde kodlanmıştır ve verilen üretim miktarı aralığındaki tüm çözümleri listeler. Tablo 5.1'deki gibi örnek bir model operasyon bilgileri ve ardından üretim miktarı alt ve üst sınırı (*P_low* ve *P_up*), günlük ortak çalışma süresi (*T*), günlük fazla çalışma süresi (*ot*) ve alt verimlilik sınırı (*e_low*) bilgileri girilerek dengeleme işlemi çalıştırılabilir (Şekil 1). Sonuç olarak, yazılım, uygun üretim miktarları (*p*) için toplam boş zamanı ($\sum id$), toplam çalışma zamanını ($\sum C_i$), toplam işçi sayısını ($\sum l_i$), operatör başına çalışma zamanını ($\sum C_i/\sum l_i$) veilgili verimlilik oranını (ef) listeler.

Örnek olarak, Tablo 5.3'teki en iyi verimlilik %97 ile 175 adetlik dengelemedir ve operatör başına ortalama 523 dk. çalışma süresi düşmektedir. Piyasa taleplerinin boyutundan dolayı, daha büyük üretim miktarları verimlilikler kontrol edilerek dikkatli bir şekilde seçilmelidir. Tablo 5.3'teki bir diğer verimli çözüm %91 verimliliğe sahip 739 adetlik üretimdir. *p=739* seçilerek yapılan dengeleme sonuç olarak Tablo 5.2'deki değerleri vermektedir. (Gürsoy, 2012)

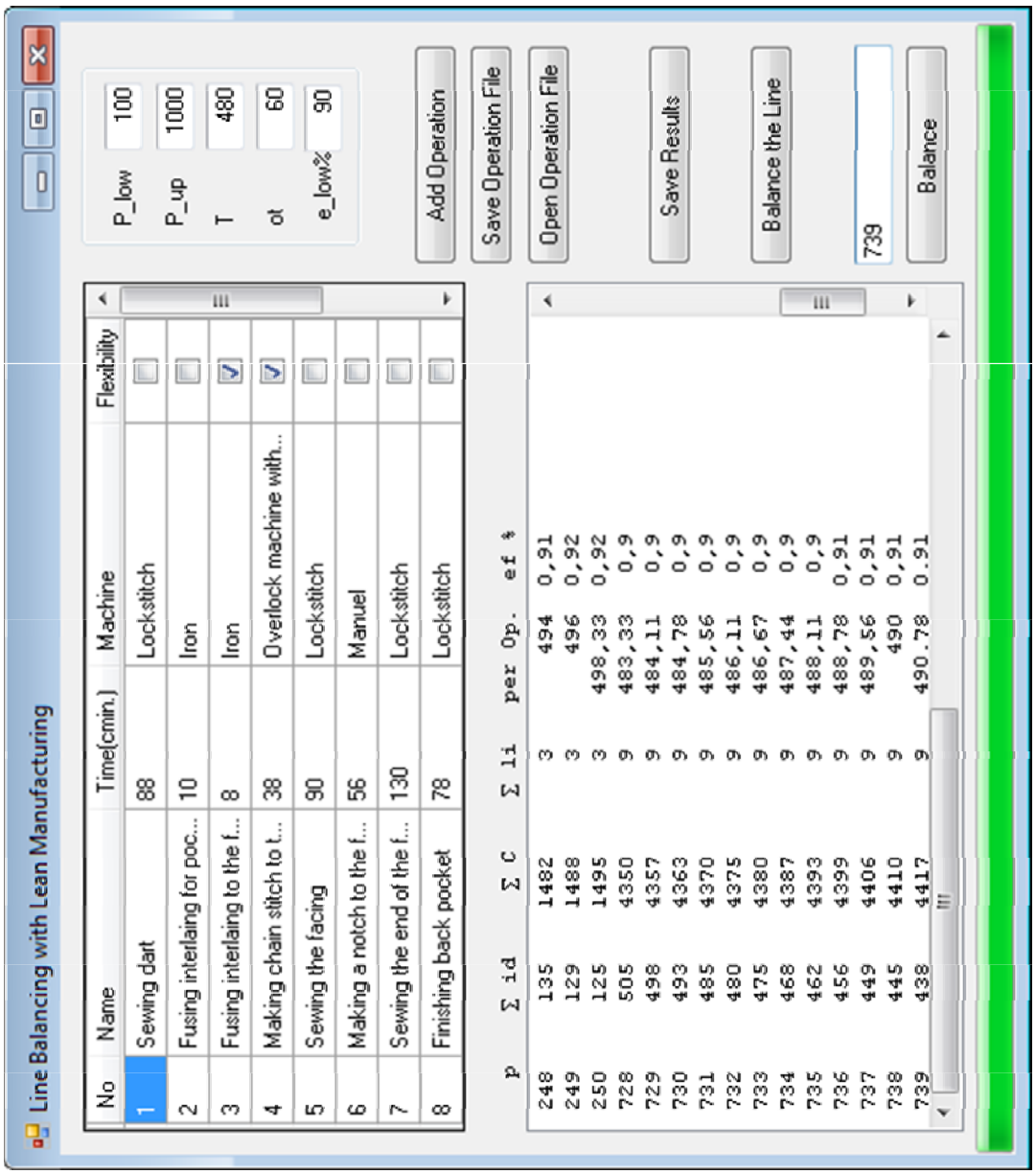
Tablo 5.2 Esnek yalın üretim hat dengeleme problemi için operasyon ataması ve verimlilik sonuçları

P	$\sum id$	$\sum C_i$	$\sum l_i$	$\sum C_i/\sum l_i$	ef %
739	438	4417	9	490,78	0,91

No	$\sum l_i$	Operations	$\sum C_i$
1	2	1; 2; 3; 4	1064
2	2	5; 6	1078
3	2	7	960
4	2	8; 9; 10	1027
5	1	11	288

Tablo 5.3 Esnek yalın üretim hat dengeleme problemi için örnek modelde %90 ve üstü verimlilikli çözümler

P	$\sum id$ (dk.)	$\sum C_i$ (dk.)	$\sum l_i$	$\sum C_i / \sum l_i$ (dk.)	ef %
162	111	967	2	483,5	0,9
163	104	974	2	487	0,9
164	98	980	2	490	0,91
165	92	986	2	493	0,91
166	86	992	2	496	0,92
167	80	998	2	499	0,92
168	74	1004	2	502	0,93
169	68	1010	2	505	0,94
170	63	1016	2	508	0,94
171	57	1021	2	510,5	0,95
172	51	1027	2	513,5	0,95
173	45	1033	2	516,5	0,96
174	39	1039	2	519,5	0,96
175	33	1046	2	523	0,97
243	166	1451	3	483,67	0,9
244	159	1458	3	486	0,9
245	153	1464	3	488	0,9
246	147	1470	3	490	0,91
247	141	1476	3	492	0,91
248	135	1482	3	494	0,91
249	129	1488	3	496	0,92
250	125	1495	3	498,33	0,92
728	505	4350	9	483,33	0,9
729	498	4357	9	484,11	0,9
730	493	4363	9	484,78	0,9
731	485	4370	9	485,56	0,9
732	480	4375	9	486,11	0,9
733	475	4380	9	486,67	0,9
734	468	4387	9	487,44	0,9
735	462	4393	9	488,11	0,9
736	456	4399	9	488,78	0,91
737	449	4406	9	489,56	0,91
738	445	4410	9	490	0,91
739	438	4417	9	490,78	0,91



Şekil 5.1 Esnek yalın üretim hat dengeleme problemi yazılımı

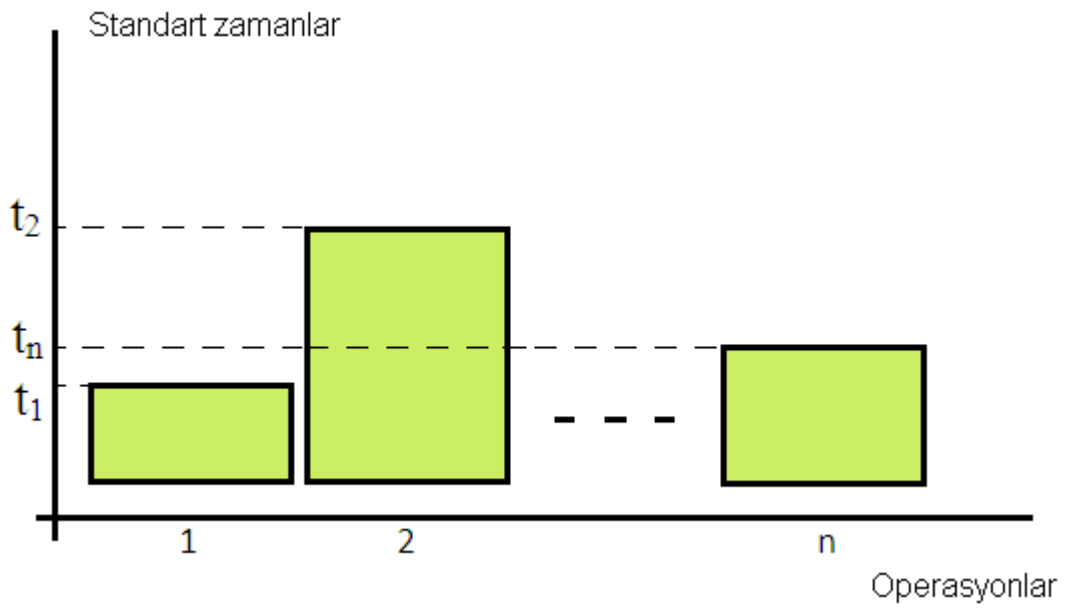
5.4.3 Performansa dayalı hat dengeleme problemi

Performansa dayalı hat dengeleme probleminde (PDHDP) işçilerin (operatörlerin) operasyonlar üzerindeki performanslarını dikkate alan ve BPP terminolojisi kullanan bir model oluşturulmuştur. Burada, bidonlar yerine operatörler, paketler yerine operasyonlar kullanılmaktadır.

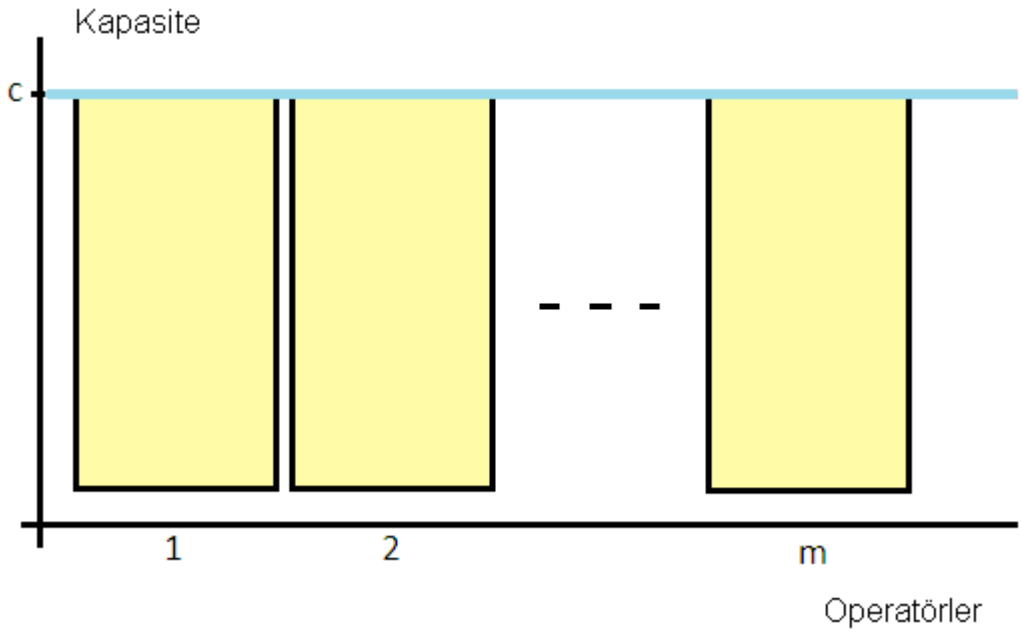
Her operatörün tüm operasyonlar üzerinde uzman olması istisnai bir durumdur ve operatörlerin uzmanlıkları çeşitlilik gösterebilir. Bu nedenle operatörlerin tüm operasyonlardaki performans oranları farklıdır. Performans oranlarındaki farklılık operasyonların standart zamanlarında farklılıklara neden olmaktadır. Her operatörün her operasyondaki performans oranı bir veri tabanında tutularak gerekli zamanlarda bu oranlar güncellenebilir (Gürsoy and Nuriyev, 2010a).

5.4.3.1 Performansa dayalı hat dengeleme probleminin matematiksel modeli

PDHDP modelinin amacı, operatöre göre faklı standart zamanlara sahip operasyonları öyle yerleştirmektir ki hat dengeleme sürecinde toplam boş zaman en az olsun. Farzedelim ki, *n* operasyon (Şekil 5.2), *m* operatör ve operatör başına günlük çalışma süresi *c* (Şekil 5.3) olsun.

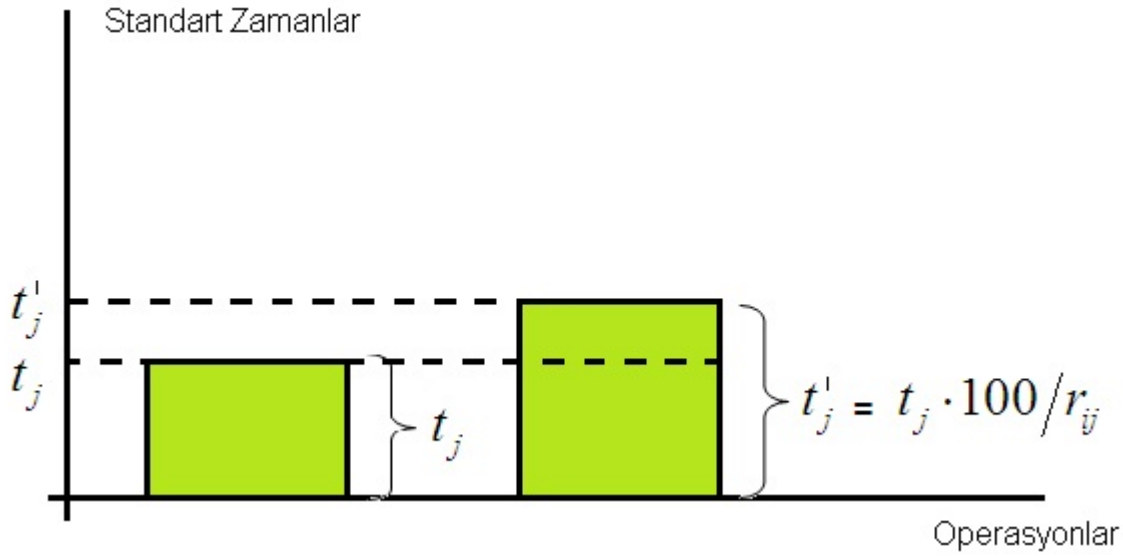


Şekil 5.2 Performansa dayalı hat dengeleme problemi için operasyonlar ve standart zamanları



Şekil 5.3 Performansa dayalı hat dengeleme problemi için operatörler ve günlük çalışma zamanları

t_j j. operasyonun standart zamanı (Şekil 5.4) ve r_{ij} i. operatörün j. operasyon üzerindeki performans oranı (Şekil 5.5) olsun. Buna göre, j. operasyon i. operatör tarafından işletilecek ise $t_j \cdot 100/r_{ij}$ zaman alır (Şekil 5.4). Operatörlerin operasyonlar üzerindeki performans oranları r_{ij} ($i = \overline{1,m}$, $j = \overline{1,n}$) $m \times n$ boyutlu matriste saklanır (Şekil 5.5).



Şekil 5.4 Performansa dayalı hat dengeleme problemi için operasyon standart zamanının operatördeki değişimi

Operator 1	r_{11}	r_{12}	...	r_{1n}
Operator 2	r_{21}	r_{22}	...	r_{2n}
	⋮	⋮	...	⋮
Operator m	r_{m1}	r_{m2}	...	r_{mn}

Şekil 5.5 Performansa dayalı hat dengeleme problemi için operatörlerin operasyonlar üzerindeki performans oranları matrisi

Girdi parametreleri:

r_{ij} : i. operatörün j. operasyon üzerindeki performans oranı (%)

t_j : j. operasyonun standart zamanı (sn.)

p : üretim miktarı

c : bir operatörün günlük çalışma zamanı (dk.)

m : operatör sayısı

n : operasyon sayısı

$i = 1,\ldots,m$

$j = 1,\ldots,n$

Karar değişkeni:

$x_{ij} = k,\ k \in \mathbb{Z}^+ \cup \{0\}$, eğer i. operatör j. operasyondan k sayıda yaparsa

Bu notasyonlar doğrultusunda PDHDP'nin tamsayılı matematiksel modeli aşağıdaki gibidir.

$$Enk \sum_{i=1}^{m}\left(c \cdot 60 - \sum_{j=1}^{n}\left(x_{ij} \cdot t_j \cdot 100/r_{ij} \right) \right) \tag{5.4.3.1}$$

Kısıtlar:

$$\sum_{i=1}^{m} x_{ij} = p, \qquad j = \overline{1,n} \tag{5.4.3.2}$$

$$\sum_{j=1}^{n} x_{ij} \cdot t_j \cdot 100/r_{ij} \le c \cdot 60, \qquad i = \overline{1,m} \tag{5.4.3.3}$$

$$x_{ij} \in \mathbb{Z}^+ \cup \{0\},\ i = \overline{1,m},\ j = \overline{1,n} \tag{5.4.3.4}$$

Burada, kısıt (5.4.3.2) her operasyonun p adet yapılmasını, kısıt (5.4.3.3) her operatöre günlük çalışma zamanının üzerinde atama yapılamayacağını kontrol eder. (5.4.3.2 - 5.4.3.4) kısıtları altında amaç (5.4.3.1) hat dengeleme işlemindeki operatör sayısını ve operatörlerdeki boş zamanı saniye biriminden en küçüklemektir. (Gürsoy and Nuriyev, 2010a)

5.4.3.2 Performansa dayalı hat dengeleme probleminin NP-tamlığı

Teorem 5.4.3.1: PDHDP NP-tam problemler sınıfındandır.

İspat: PDHDP'nin NP-tamlığını ispatlayabilmek için bir NP-tam problem olan BPP'ye kısıtlayacağız.

PDHDP tanıma versiyonu aşağıdaki gibidir.

Koşul: n elemanlı operasyonlar kümesi U, m elemanlı işçi kümesi V, üretim miktarı p, $t_j \in \mathbb{Z}^+$ $(j=\overline{1,n})$ j. operasyonun standart zamanı, r_{ij} $(i=\overline{1,m},\ j=\overline{1,n})$ i. operatörün j. operasyon üzerindeki performans oranı, günlük çalışma süresi c, ve $K \in \mathbb{Z}^+$ $(K \le m)$ verilsin.

Soru: Operasyonlar kümesi U'nun operatörler kümesi V üzerinde öyle bir K parçalanması var mıdır ki $i=\overline{1,K}$ için her operatöre yapılan atama (parçalanma) operatörün günlük çalışma süresinden fazla olmasın?

Genellikten uzaklaşmadan, her işçinin her operasyon üzerinde %100 performansla çalışabildiğini kabul edelim. Bu durumda bidon kapasitesi olarak c ve eşya ağırlığı olarak w_j $(j=\overline{1,n})$ (standart zaman ile işletilen operasyon sayısının çarpımı olarak) dikkate alınırsa problem BPP'ye kısıtlanmış olacaktır. Böylece PDHDP'nin NP-tam sınıftan olduğu ispatlanmıştır.

5.4.3.3 Performansa dayalı hat dengeleme problemi için sezgisel algoritma

Girdi parametreleri:

p , üretim miktarı,

c, bir operatörün günlük çalışma zamanı (dk),

m, operatör sayısı,

n, operasyon sayısı,

$r_{ij} \in \mathbb{Z}^+$ $(i = \overline{1,m},\ j = \overline{1,n})$, i. operatörün j. operasyon üzerindeki performans oranını gösteren $m \times n$ boyutlu matris,

$t_j \in \mathbb{Z}^+$ $(j = \overline{1,n})$ j. operasyonun standart zamanı (sn) gösteren n boyutlu vektördür.

Bu girdi parametreleri kullanılarak her operatörün her operasyondan kaç adet üreteceğini gösterecek x_{ij} matrisi belirlenmek istenir.

$x_{ij} \in \mathbb{Z}^+ \cup \{0\}$ $(i = \overline{1,m},\ j = \overline{1,n})$, i. operatörün j. operasyondan kaç adet yapacağını gösterir ve bu matriste sütun toplamları üretim miktarı p'ye eşittir.

x_{ij}	***1***	***2***	***…***	***n***
1				
2				
⋮				
m				

PDHDP greedy algoritması:

1. *p, m, n, c, n* boyutlu t_j standart zaman vektörünü, $m \times n$ boyutlu r_{ij} performans matrisini öğren.
2. $m \times n$ boyutlu x_{ij} üretim miktarı matrisini matrisini sıfırla.
3. *for j=1 to n do*
4. $p' = p$
5. while ($p' > 0$)
6. *j*. operasyon için uygun (boş zamanı olan) operatörlerden en yüksek performanslı operatörün indisini *k* olarak belirle. Birden fazla ise indis sırasına göre seç.
7. *k*. operatörün *j*. operasyon için yapabileceği en fazla üretim miktarı *s*'i belirle ($s \le p$ olmak üzere $s \cdot t_j \cdot 100 / r_{kj} \le c \cdot 60$ olmalıdır).
8. $p' = p' - s$
9. $x_{kj} = s$
10. *idle=0*
11. *for i=1 to m do*
12. $workload = 0$
13. *for j=1 to n do*
14. $workload = workload + x_{ij} \cdot t_j \cdot 100 / r_{ij}$
15. $idle = idle + c \cdot 60 - workload$
16. *Print idle*

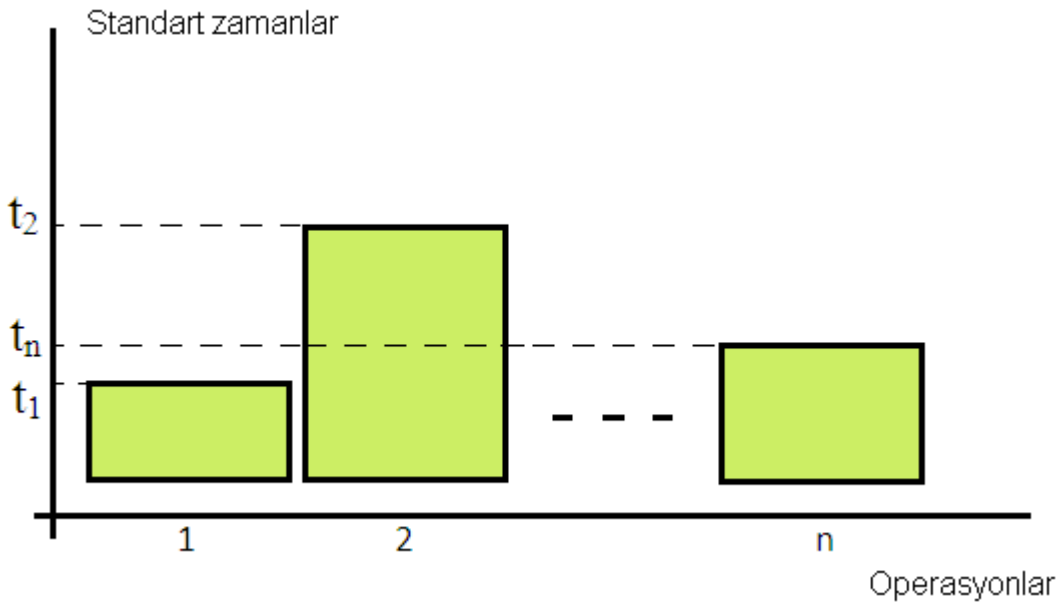
PDHDP için geliştirilmiş bu sezgisel algoritma her adımda yerel en iyi durumu değerlendirdiği için bir greedy algoritmadır. Bu algoritmada atama yapan 1-9 adımları en kötü durumda $O(n \cdot m)$ karmaşıklığındadır. Yapılan atama üzerine kalan boş zamanların hesaplanması (10-15 adımları) ise $O(n \cdot m)$

karmaşıklığındadır. Sonuç olarak PDHDP greedy algoritması $O(n \cdot m)$ karmaşıklıkta olup polinomiyal bir algoritmadır.

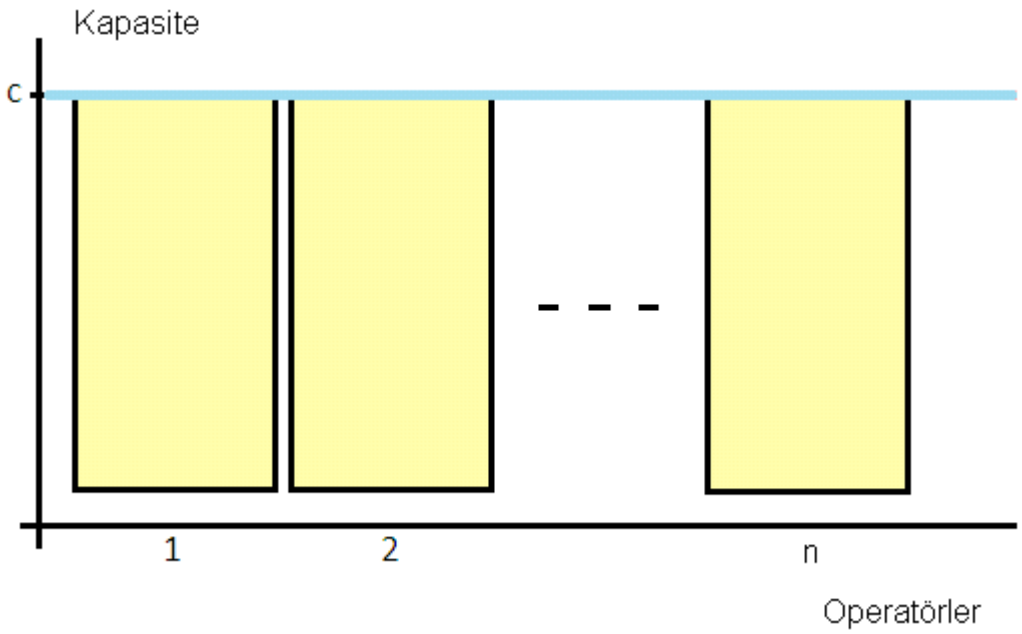
5.4.4 Esneklik kısıtlı yalın üretim hat dengeleme problemi

Esneklik kısıtlı yalın üretim hat dengeleme probleminde (EKYÜHDP) yalın üretim hat dengeleme problemi için operasyonların esneklikleri dikkate alınarak bir 0-1 tamsayılı matematiksel model oluşturulmuştur. Bu model BPP formülasyonu kullanılarak tanımlanmış (Martello ve Toth, 1990), bidonlar yerine operatörler, operasyonlar yerine paketler ve paket ağırlıkları yerine ise operasyonların standart zamanları kullanılmıştır (Gürsoy and Nuriyev, 2010b).

EKYÜHDP'de *n* operasyon, en çok *n* operatör var olduğu (Şekil 5.6 ve Şekil 5.7) kabul edilir. Tüm operatörler bir günde aynı çalışma zamanına (*c*) sahiptir ve her operasyonun günlük çalışma süresinin bir operatörün çalışma süresinden fazla olmadığı kabul edilir. Bunun yanında bazı operasyonlar birtakım önceliklere sahip olabilmektedir. Yani, *j*. operasyon s_j operasyonundan sonra ve o_j operasyonundan önce yapılmak şartıyla herhangi bir noktada yapılabilir $(s_j \prec j \prec o_j)$ (Nuriyev et al., 2008, 2009).



Şekil 5.6 Esneklik kısıtlı yalın üretim hat dengeleme problemi için operasyonlar ve standart zamanları



Şekil 5.7 Esneklik kısıtlı yalın üretim hat dengeleme problemi için operatörler ve çalışma zamanı

5.4.4.1 Esneklik kısıtlı yalın üretim hat dengeleme problemi için 0-1 tamsayılı matematiksel model

EKYÜHDP modelinin amacı, operasyon önceliklerini hesaba katarak maksimum çalışma zamanı ile operasyonları operatörlere atamaktır. Çalışacak minimum operatör sayısını bulmak da aynı amaca hizmet edecektir.

Girdi parametreleri:

t_j : j. operasyonun standart zamanı (sn.)

s_j : j. operasyonun esneklik başlangıcı

o_j : j. operasyonun esneklik bitişi

c : bir işçinin günlük çalışma zamanı (dk.)

p : üretim miktarı

n : operasyon ve operatör sayısı

$i = \overline{1,n},\ j = \overline{1,n}.$

Karar değişkenleri:

$$y_i = \begin{cases} 1, & i.\text{operatör kullanıldıysa} \\ 0, & \text{aksi halde} \end{cases} \qquad i = \overline{1,n}$$

$$x_{ij} = \begin{cases} 1 & , j.\text{operasyon } i.\text{operatöre atandıysa} \\ 0, & \text{aksi halde} \end{cases} \qquad i = \overline{1,n},\ j = \overline{1,n}$$

Girdi parametreleri ve karar değişkenleri yukarıdaki gibi tanımlanan EKYÜHDP'nin 0-1 tamsayılı matematiksel modeli aşağıdaki gibidir.

$$Enk \sum_{i=1}^{n} y_i \tag{5.4.4.1}$$

Kısıtlar:

$$\sum_{j=1}^{n} p \cdot t_j \cdot x_{ij} \le c \cdot y_i, \qquad i = \overline{1,n}, \tag{5.4.4.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = \overline{1,n}, \tag{5.4.4.3}$$

$$i \cdot x_{ij} - k \cdot x_{ks_j} \ge 0, \qquad i, j, k = \overline{1,n}, \tag{5.4.4.4}$$

$$k \cdot x_{ko_j} - i \cdot x_{ij} \ge 0, \qquad i, j, k = \overline{1,n}, \tag{5.4.4.5}$$

$$x_{ij} = 0 \vee 1, \qquad i, j = \overline{1,n}, \tag{5.4.4.6}$$

$$y_i = 0 \vee 1, \qquad i = \overline{1,n}, \tag{5.4.4.7}$$

Burada, kısıt (5.4.4.2) *i*. operatöre atanabilecek operasyonların günlük çalışma zamanından fazla olmamasını kontrol eder. Kısıt (5.4.4.3) her operasyonun sadece bir operatöre atanabileceğini garanti eder. (5.4.4.4 - 5.4.4.5) kısıtları operasyonların verilen esneklik aralığında uygun noktalara yerleştirilmesini kontrol eder. (5.4.4.2 – 5.4.4.7) kısıtları altında, amaç (5.4.4.1) operatör sayısını en küçüklemektir (Gürsoy and Nuriyev, 2010b).

5.4.4.2 Esneklik kısıtlı yalın üretim hat dengeleme probleminin NP-tamlığı

Teorem 5.4.4.1: Esneklik kısıtlı yalın üretim hat dengeleme problemi NP-tam problemler sınıfındandır.

İspat: Esneklik kısıtlı yalın üretim hat dengeleme probleminin NP-tamlığını ispatlayabilmek için bir NP-tam problem olan BPP'ye (Bölüm 4.2) kısıtlayacağız.

Esneklik kısıtlı yalın üretim hat dengeleme probleminin tanıma versiyonu aşağıdaki gibidir.

Koşul: n elemanlı operasyonlar kümesi U, m elemanlı operatörler kümesi V, üretim miktarı p, $j=\overline{1,n}$ için $t_j \in \mathbb{Z}^+$ j. operasyonun standart zamanı, $s_j \in \mathbb{Z}^+$ j. operasyonun esneklik başlangıcı, $o_j \in \mathbb{Z}^+$ j. operasyonun esneklik bitişi, günlük çalışma süresi c ve $K \in \mathbb{Z}^+$ $(K \le m)$ verilsin.

Soru: Operasyonlar kümesi U'nun operatörler kümesi V üzerinde K sayısını aşmayacak şekilde bir parçalanması var mıdır?

Genellikten uzaklaşmadan, ilk ve son operasyonu esnek olmayan operasyon, yani $s_1=1$, $o_1=1$, $s_n=n$, $o_n=n$, diğer tüm operasyonları $s_j=1$ ve $o_j=n$ olacak şekilde esnek, yani tüm hat dengeleme süresince $(1,n)$ açık aralığında herhangi bir sırada yapılabilen operasyonlar olarak kabul edelim. Burada bidon kapasitesi c ve eşya ağırlığı $w_j = p \cdot t_j$ $(j=\overline{1,n})$ olacaktır.

Bu durumda, Esneklik Kısıtlı Yalın Üretim Hat Dengeleme Problemi, 1. eşya 1. bidona ve n. eşya K. bidona yerleştirilmek üzere, K sayısını aşmayacak şekilde bidon paketleme yaparak BPP'ye kısıtlanmış olacaktır. Böylece Esneklik Kısıtlı Yalın Üretim Hat Dengeleme Problemi NP-tam sınıftan olduğu ispatlanmıştır.

5.4.4.3 Esneklik kısıtlı yalın üretim hat dengeleme problemi için genetik algoritma

Girdi parametreleri:

p, üretim miktarı,

c, bir operatörün günlük çalışma zamanı (dk),

n, operasyon ve operatör sayısı,

$\overline{\overline{n}}$, esnek olmayan operasyon sayısı ($1 \le \overline{\overline{n}} \le n$),

$\overline{n}$, esnek operasyon sayısı ($1 \le \overline{n} \le n$),

$t_j \in \mathbb{Z}^+$ ($j = \overline{1,n}$) *j*. operasyonun standart zamanı (sn) gösteren *n* boyutlu vektör,

s_j ($j = \overline{1,n}$), *j*. operasyonun esneklik başlangıcı, o_j ($j = \overline{1,n}$) *j*. operasyonun esneklik bitişi olsun.

Bu girdi parametreleri kullanılarak ve üretim sürecindeki tüm operasyonları tamamlamak koşuluyla kaç operatör gerektiğinin belirlenmesi istenir.

Birey oluştur alt algoritması:

1. Esnek olmayan operasyonları ardışık olarak sırala.
2. *for j=1 to* $\overline{n}$ *do*
3. j. esnek operasyonu $\left(s_j, o_j\right)$ açık aralığından rastgele bir noktada araya ekle.
4. Esneklik kısıtlarına göre oluşturulan dizilimin sırasında $p \cdot t_j$'leri hesapla.
5. Next-Fit algoritmasına göre atama yaparak kullanılan operatör sayısını belirle ve bunu fitness değeri olarak ata.

EKYÜHDP genetik algoritması:

1. *for j=1 to n do*
2. *Birey oluştur()*
3. Oluşturulan bireyi, fitness değerine göre populasyona azalmayan sırada ekle.
4. *crosn=0,9*n, mutn=0,1*n*
5. *while (*en iyi fitness değeri *n* defa tekrarlanıncaya kadar*)*
6. *for i=1 to crosn do*
7. Rulet tekerleğine göre populasyondan iki birey seç.
8. Seçilen bireyleri çaprazla.
9. Çaprazlama sonucu oluşan çocuk bireylerin ikisi birden esneklik kısıtlarını sağlamıyor ise A7'ye git
10. Esneklik kısıtlarını sağlayan çocuk bireyler için Next-Fit algoritmasını çalıştır, opreratörlere ata, fitness değerini belirle ve populasyona artmayan sırada ekle.
11. *for i=1 to mutn do*
12. Rulet tekerleğine göre populasyondan bir birey seç.
13. Seçilen bireyden rastgele bir esnek operasyon (gen) belirle.
14. Seçilen geni yeni rastgele ve esneklik kısıtı altında rastgele yer değiştir.
15. Mutasyon sonucu oluşan çocuk birey için Next-Fit algoritmasına göre opreratörlere yerleştir, fitness değerini belirle ve populasyona artmayan sırada ekle.
16. Populasyonun 1. elemanını yani fitness değeri en küçük bireyi yazdır.

Birey oluştur alt algoritması, BPP'nin Next-Fit algoritması bir alt prosedür olarak kullanılarak yeni bireyler oluşturmaktadır ve karmaşıklığı $O(n)$'dir. *Birey oluştur* alt algoritması aynı zamanda EKYÜHDP için bir sezgisel algoritma özelliği de taşımaktadır. Genetik algoritma ise elde edilen en iyi fitness değeri *n* defa tekrarlanıncaya karar nesiller üretmeye devam edecektir.

5.4.5 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi

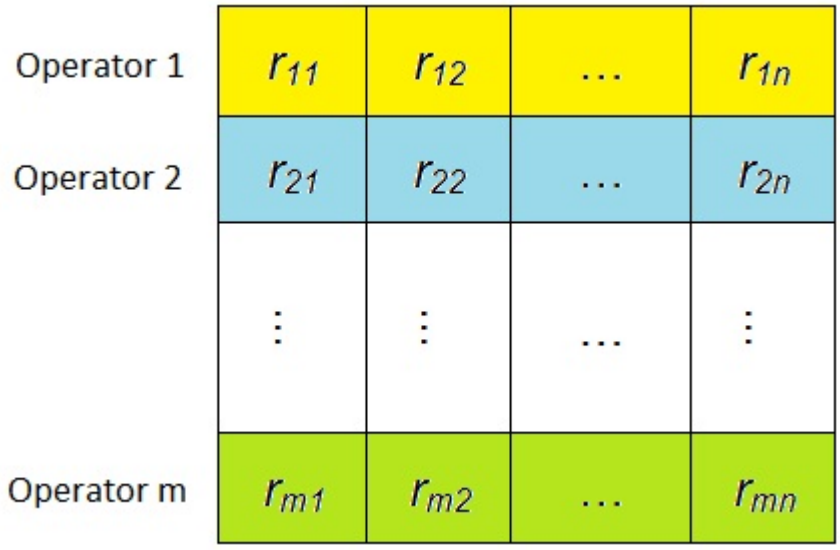
HDP, görevlerin öncelik ilişkileri de dikkate alınarak hat üzerindeki iş istasyonlarına atanması olarak tanımlanabilir. HDP için bugüne kadar yapılan çalışmalar mevcut operatör grubu içinden operasyonlara doğrudan atamalar yapmak üzerine yapılmıştır. Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme probleminde (PDEKYÜHDP) yalın üretim tekniği kullanılarak, esnek ve esnek olmayan operasyonlar içerebilen bir üretim hattı için, operasyonlar üzerindeki performansları önceki hat dengelemelere dayanılarak tespit edilmiş operatör grubundan en uygun seçimi yapmak ve operasyon önceliklerini gözeterek hattı optimum olarak dengelemek üzerine bir problem tanımlanmış ve tamsayılı matematiksel modeli oluşturulmuştur.

Esnek operasyon, üretim şemasında belirli bir aralıkta üretilme özelliği olan operasyondur. Yani, esnek bir operasyonun, mevcut üretim miktarına bağlı olarak esneklik aralığı bozulmadan üretim şemasındaki sırası değiştirilebilir. Tablo 5.4'deki 3, 4 ve 7 nolu operasyonlar esnektir.

Esnek olmayan operasyonlar, üretim şemasında kendi aralarında bir sıralama ilişkisi olan operasyonlardır ve üretim sürecinde bu sıra kesinlikle bozulamaz. Tablo 5.4'de 1, 2, 5, 6, 8, 9, 10 ve 11 nolu operasyonlar esnek değildir ve bu operasyonların sıralamaları bozulamaz. Yani operasyon 5 operasyon 2 bitirildikten sonra ve operasyon 6 başlamadan önce tamamlanmalıdır.

Tablo 5.4 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için örnek model operasyon detayları

No	Operasyon Adı	Standart zamanı	Makinası	Esnek mi?	Esneklik başlangıcı	Esneklik Bitişi
1	Sason Dikme	88	Düz dikiş	Hayır		
2	Cep yeri tela yapıştırma	10	Ütü	Hayır		
3	Flatoya tela yapıştırma	8	Ütü	Evet	1	6
4	Flato ve karşılığı overlok	38	3 iplik overlok	Evet	2	8
5	Flato dikme	90	Düz dikiş	Hayır		
6	Flato çentik atma	56	Elde	Hayır		
7	Flato uç tutturma ve gaze	130	Düz dikiş	Evet	3	10
8	Arka cep bitirme	78	Düz dikiş	Hayır		
9	Arka cep torbalama	24	5 iplik overlok	Hayır		
10	Arka orta kapama overlok	37	5 iplik overlok	Hayır		
11	Arka orta gaze	39	Düz dikiş	Hayır		



Şekil 5.8 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için operatörlerin operasyonlar üzerindeki performans oranları matrisi

5.4.5.1 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için 0-1 tamsayılı matematiksel model

Girdi parametreleri:

p, üretim miktarı

$M = \{1,2,...,m\}$, dikim sürecinde çalışabilecek operatörler kümesi

$N = \{1,2,...,n\}$, dikim sürecindeki operasyonlar kümesi

r_{ij}, i. operatörün j. operasyondaki performans oranı ($i = \overline{1,m}, j = \overline{1,n}$)

t_j , j. operasyonun standart zamanı (sn.) ($j = \overline{1,n}$)

T, günlük çalışma süresi (dk.),

ot, günlük fazla mesai süresi (dk.),

$N = F \cup F'$

$\overline{\overline{n}}$, esnek olmayan operasyon sayısı ($1 \le \overline{\overline{n}} \le n$)

$\overline{n}$, esnek operasyon sayısı ($1 \le \overline{n} \le n$)

$F' = \left(f_1', f_2', ..., f_{\overline{\overline{n}}}' \right)$, esnek olmayan operasyonlar kümesi ($f_j' \in N, 1 \le j \le \overline{\overline{n}}$)

Tüm esnek olmayan operasyonlar kendi arasında kısmi sıralı olmalıdır; $f_j' \prec f_{j+1}'$, yani f_j' operasyonu f_{j+1}' operasyonundan önce tamamlanmalıdır ($1 \le j \le \overline{\overline{n}}$)

$F = \left(f_1, f_2, ..., f_{\overline{n}} \right)$, esnek olmayan operasyonlar kümesi ($1 \le \overline{n} \le n$; $f_j \in N$, $1 \le j \le \overline{n}$)

b_j , f_j esnek operasyonu için esnekliğin başladığı operasyon numarası ($1 \le j \le \overline{n}$)

e_j , f_j esnek operasyonu için esnekliğin bittiği operasyon numarası ($1 \le j \le \overline{n}$)

Tüm esnek operasyonlar kendi esneklik aralığında tamamlanmalıdır: $b_j < f_j < e_j$, $1 \le j \le \overline{n}$

Karar değişkenleri:

$$x_{ij} = \begin{cases} 1, & \text{eğer } j\text{. operasyon } i\text{. işçi grubuna atandıysa} \\ 0, & \text{aksi halde} \end{cases} \qquad i = \overline{1,m},\ j = \overline{1,n}$$

$$y_{ik} = \begin{cases} 1, & i\text{. operatör } k\text{. sırada kullanıldıysa} \\ 0, & \text{aksi halde} \end{cases} \qquad i = k = \overline{1,m}$$

Öyle bir operasyon sıralaması ve bu sıralamaya uygun olacak operatör seçimi var mıdır ki p adet ürün için minimum boş zamanlı bir dengeleme oluşsun?

Yukarıdaki girdi parametreleri yardımıyla performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için 0-1 tamsayılı matematiksel modeli aşağıdaki şekildedir:

$$Enk \left[\left(\sum_{i=1}^{m} \sum_{k=1}^{m} y_{ik} \right) \cdot (T + ot) - \sum_{i=1}^{m} \sum_{j=1}^{n} p \cdot t_j \cdot x_{ij} \cdot 100 / r_{ij} \right] \Big/ \sum_{i=1}^{m} \sum_{k=1}^{m} y_{ik} \qquad (5.4.5.1)$$

Kısıtlar:

$$\sum_{j=1}^{n} p \cdot t_j \cdot x_{ij} \cdot 100 / r_{ij} \le T + ot, \quad i = \overline{1,m} \qquad (5.4.5.2)$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = \overline{1,n} \qquad (5.4.5.3)$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = n \qquad (5.4.5.4)$$

$$k \cdot x_{if_k'} - j \cdot x_{if_j'} > \sum_{l=1}^{n} x_{if_l'}, \ i = \overline{1,m}, j = \overline{\overline{1,n}}, k = \overline{\overline{j+1,n}}, x_{if_j'} = 1, x_{if_k'} = 1 \qquad (5.4.5.5)$$

$$l \cdot y_{il} \cdot x_{if_j} - s \cdot y_{ks} \cdot x_{kb_j} \ge 0, \ i = k = l = s = \overline{1,m}, j = \overline{\overline{1,n}}, x_{if_j} = 1, x_{ib_j} = 1 \qquad (5.4.5.6)$$

$$s \cdot y_{ks} \cdot x_{ke_j} - l \cdot y_{il} \cdot x_{if_j} \ge 0, \ i = k = l = s = \overline{1,m}, j = \overline{\overline{1,n}}, x_{if_j} = 1, x_{ie_j} = 1 \qquad (5.4.5.7)$$

$$x_{ij} = 0 \vee 1, \qquad i = \overline{1,m}, j = \overline{1,n} \tag{5.4.5.8}$$

$$y_{ik} = 0 \vee 1, \qquad i = k = \overline{1,m} \tag{5.4.5.9}$$

Yukarıdaki modelde, (5.4.5.2) her operatörün en fazla mesai süresi kadar çalışabileceğini, (5.4.5.3) her operasyonun sadece bir operatör tarafından işletilebileceğini, (5.4.5.4) üretim hattındaki tüm operasyonların işletilmesini, (5.4.5.5) esnek olmayan operasyonlar arasındaki sıralamayı kontrol eder. (5.4.5.6) ve (5.4.5.7) kısıtları ise esnek olmayan operasyonların esneklik aralığı içinde olmasını sağlar. (5.4.5.2 - 5.4.5.9) kısıtları altında bu Boolean tamsayılı matematiksel modelin amacı (5.4.5.1), üretim hattında çalışan operatör başına boş zamanı minimize etmektir.

5.4.5.2 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme probleminin NP-tamlığı

Teorem 5.4.5.1: Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi NP-tam'dır.

İspat: Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme probleminin NP-tamlığını ispatlayabilmek için bir NP-tam problem olan BPP'ye (Bölüm 4.2) kısıtlayacağız.

Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme probleminin tanıma versiyonu aşağıdaki gibidir.

Koşul:

n elemanlı operasyonlar kümesi N,

m elemanlı operatörler kümesi M,

$t_j \in \mathbb{Z}^+$, ($j = \overline{1,n}$) j. operasyonun standart zamanı,

p, üretim miktarı,

r_{ij}, i. operatörün j. operasyondaki performans oranı ($i = \overline{1,m}, j = \overline{1,n}$),

$\overline{\overline{n}}$, esnek olmayan operasyon sayısı ($1 \le \overline{\overline{n}} \le n$),

$\overline{n}$, esnek operasyon sayısı ($1 \le \overline{n} \le n$),

$F' = \left(f_1', f_2', ..., f_{\overline{\overline{n}}}' \right)$, esnek olmayan operasyonlar kümesi ($f_j' \in N$, $1 \le j \le \overline{\overline{n}}$),

$F = \left(f_1, f_2, ..., f_{\overline{n}} \right)$, esnek olmayan operasyonlar kümesi ($1 \le \overline{n} \le n$, $f_j \in N$, $1 \le j \le \overline{n}$),

b_j, f_j esnek operasyonu için esnekliğin başladığı operasyon numarası ($1 \le j \le \overline{n}$)

e_j, f_j esnek operasyonu için esnekliğin bittiği operasyon numarası ($1 \le j \le \overline{n}$)

ve $K \in \mathbb{Z}^+$ ($K \le m$) verilsin.

Soru: Tüm esnek olmayan operasyonlar kendi arasında kısmi sıralı olmak: $f_j' \prec f_{j+1}'$ ($1 \le j \le \overline{\overline{n}}$) ve tüm esnek operasyonlar kendi esneklik aralığında tamamlanmak: $b_j < f_j < e_j$, $1 \le j \le \overline{n}$ üzere, operasyonlar kümesi *N*'nin operatörler kümesi *M* üzerinde *K* sayısını aşmayacak şekilde bir parçalanması var mıdır?

Genellikten uzaklaşmadan, ilk ve son operasyonu esnek olmayan operasyon, yani $b_1 = 1$, $e_1 = 1$, $b_n = n$, $e_n = n$, diğer tüm operasyonları $b_j = 1$ ve $e_j = n$ olacak şekilde esnek, yani tüm hat dengeleme süresince (1,*n*) açık aralığında herhangi bir sırada yapılabilen operasyonlar olarak ve tüm operatörleri tüm operasyonlar üzerinde %100 performans ile çalıştığını, yani $r_{ij} = 100$, kabul edelim. Burada bidon kapasitesi $c = T + ot$ ve eşya ağırlığı $w_j = p \cdot t_j \cdot 100 / r_{ij}$ ($i = \overline{1,K}$, $j = \overline{1,n}$) olacaktır.

Bu durumda, Performansa Dayalı Esneklik Kısıtlı Yalın Üretim Hat Dengeleme Problemi, 1. eşya 1. bidona ve *n*. eşya *K*. bidona yerleştirilmek üzere, *K* sayısını aşmayacak şekilde bidon paketleme yaparak BPP'ye kısıtlanmış olacaktır. Böylece Esneklik Kısıtlı Yalın Üretim Hat Dengeleme Problemi NP-tam sınıftan olduğu ispatlanmıştır.

5.4.5.3 Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için sezgisel bir algoritma

Girdi parametreleri:

p, üretim miktarı, $T+ot$, bir operatörün günlük çalışma zamanı (dk), n elemanlı operasyonlar kümesi N, m elemanlı operatörler kümesi M, $\overline{\overline{n}}$, esnek olmayan operasyon sayısı ($1 \le \overline{\overline{n}} \le n$), $\overline{n}$, esnek operasyon sayısı ($1 \le \overline{n} \le n$),

$t_j \in \mathbb{Z}^+$ ($j = \overline{1,n}$) j. operasyonun standart zamanı (sn) gösteren n boyutlu vektör,

r_{ij} ($i = \overline{1,m}, j = \overline{1,n}$), i. operatörün j. operasyondaki performans oranını gösteren $m \times n$ boyutlu matris,

$F' = \left(f_1', f_2', ..., f_{\overline{\overline{n}}}' \right)$, esnek olmayan operasyonlar kümesi ($f_j' \in N$, $1 \le j \le \overline{\overline{n}}$),

$F = \left(f_1, f_2, ..., f_{\overline{n}} \right)$, esnek olmayan operasyonlar kümesi ($1 \le \overline{n} \le n$, $f_j \in N$, $1 \le j \le \overline{n}$),

b_j, f_j esnek operasyonu için esnekliğin başladığı operasyon numarası, e_j, f_j esnek operasyonu için esnekliğin bittiği operasyon numarası ($1 \le j \le \overline{n}$) olsun.

Bu girdi parametreleri kullanılarak ve üretim sürecindeki tüm operasyonları tamamlamak koşuluyla kaç operatör gerektiğinin belirlenmesi istenir.

Dizilim oluştur alt algoritması:

1. Esnek olmayan operasyonları ardışık olarak sırala.
2. *for j=1 to* $\overline{n}$ *do*
3. j. esnek operasyonu $\left(b_j, e_j \right)$ açık aralığından rastgele bir noktada araya ekle.

PDEKYÜHDP greedy algoritması:

1. *p, m, n, c, n* boyutlu t_j standart zaman vektörünü, $m \times n$ boyutlu r_{ij} performans matrisini ve dizilimi öğren.
2. y_{ik} ve x_{ij} matrislerini sıfırla.
3. *i=1, u=1*
4. *for j=1 to n do*
5. if $u = 1$ ise *j*. operasyon için uygun (boş zamanı olan) operatörlerden en yüksek performanslı operatörün indisini *k* olarak belirle. Birden fazla ise indis sırasına göre seç.
6. $u = 1$
7. *k*. operatöre *j*. operasyon için yapabileceği en fazla üretim miktarını ata ($s \le p$ olmak üzere $s \cdot t_j \cdot 100 / r_{kj} \le (T + ot) \cdot 60$).
8. $y_{ki} = 1$, $x_{kj} = 1$
9. *k*. operatör doldu ve *j*. operasyon tamamlandı ise $i = i + 1$.
10. *k*. operatör doldu ve *j*. operasyon tamamlanmadı ise $i = i + 1$, goto 4.
11. *k*. operatör dolmadı ise $u = 0$
12. $worker = 0$, $workload = 0$
13. *for i=1 to m do*
14. *for j=1 to n do*
15. $workload = workload + p \cdot t_j \cdot x_{ij} \cdot 100 / r_{ij}$
16. *for k=1 to m do*
17. $worker = worker + y_{ik}$
18. *Operatör başına düşen zamanı* $(worker \cdot (T + ot) - workload) / worker$ hesapla.

PDEKYÜHDP genetik algoritması:

1. *for j=1 to n do*

2. *Dizilim oluştur()*

3. *PDEKHDP greedy(dizilim)*

4. Operatör başına düşen boş zamanı oluşturulan bireyin, fitness değeri olarak ata ve fitness değerine göre populasyona azalmayan sırada ekle.

5. *crosn=0,9*n, mutn=0,1*n*

6. *while (*en iyi fitness değeri *n* defa tekrarlanıncaya kadar*)*

7. *for i=1 to crosn do*

8. Rulet tekerleğine göre populasyondan iki birey seç.

9. Seçilen bireyleri çaprazla.

10. Çaprazlama sonucu oluşan çocuk bireylerin ikisi birden esneklik kısıtlarını sağlamıyor ise A8'e git

11. Esneklik kısıtlarını sağlayan çocuk bireylerin dizilimleri için *PDEKHDP greedy()* alt prosedürünü çalıştır, fitness değerini belirle ve populasyona artmayan sırada ekle.

12. *for i=1 to mutn do*

13. Rulet tekerleğine göre populasyondan bir birey seç.

14. Seçilen bireyin diziliminden rastgele bir esnek operasyon (gen) belirle.

15. Seçilen geni rastgele ve esneklik kısıtı altında rastgele yer değiştir.

16. Mutasyon sonucu oluşan çocuk bireyin dizilimi için *PDEKHDP greedy()* alt prosedürünü çalıştır, fitness değerini belirle ve populasyona artmayan sırada ekle.

17. Populasyonun 1. elemanını yani fitness değeri en küçük bireyi yazdır.

Dizilim oluştur alt algoritması, hat dengeleme sürecindeki esnek ve esnek olmayan operasyonları dikkate alarak uygun bir operasyon sıralaması oluşturur. Öncelikle aralarında ardışıklık olan esnek olmayan operasyonlar sıralanır. Ardından, her j esnek operasyonu (b_j, e_j) açık aralığında rastgele seçilen bir konuma yerleştirilir. Dizilim oluştur alt algoritmasının karmaşıklığı $O(n)$'dir.

PDEKHDP greedy algoritması, rastgele ama kurallı oluşturulan bir dizilimde, karşılaştığı en yüksek performans oranlarını dikkate alarak dizilim sırasını bozmadan operasyonları operatörlere atar. PDEKHDP greedy algoritması $O(m \cdot n)$ karmaşıklığındadır.

Dizilim oluştur ve *PDEKHDP greedy algoritmasını* kullanan *PDEKHDP genetik algoritması* ise elde edilen en iyi fitness değeri n defa tekrarlanıncaya karar nesiller üretmeye devam eder ve çalışmayı durdurduğunda populasyon listesindeki birinci yani minimum ortalama boş zamanlı atamayı yazdırır.

6. SONUÇ

Bu tezde, literatürdeki esnek yalın üretim hat dengeleme problemi geliştirilmiş, performansa dayalı hat dengeleme problemi, esneklik kısıtlı yalın üretim hat dengeleme problemi ve performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi olmak üzere üç yeni hat dengeleme problemi tanımlanmıştır.

Bu hat dengeleme problemlerinden,

I. Esnek yalın üretim hat dengeleme problemi için

a. problemin yeni bir 0-1 tamsayılı matematiksel modeli oluşturulmuş,

b. problemin NP-tamlık ispatı yapılmış,

c. problemin polinomiyal karmaşıklıkta yeni bir sezgisel algoritması tasarlanmış,

d. bu algoritmaya dayalı C# dilinde kodlanmış bir paket program hazırlanmış,

e. örnek bir üretim modeli üzerinde hesaplama denemeleri yapılarak sonuçlar listelenmiştir.

II. Performansa dayalı hat dengeleme problemi için,

a. problem tanımlanarak literatüre kazandırılmış,

b. problemin tamsayılı matematiksel modeli tasarlanmış,

c. problemin NP-tamlık ispatı yapılmış,

d. problem için polinomiyal karmaşıklıkta bir greedy algoritma tasarlanmıştır.

III. Esneklik kısıtlı yalın üretim hat dengeleme problemi için,

a. problem tanımlanarak literatüre kazandırılmış,

b. problemin 0-1 tamsayılı matematiksel modeli tasarlanmış,

c. problemin NP-tamlık ispatı yapılmış,

d. problem için polinomiyal karmaşıklıkta bir greedy algoritma ve bu greedy ile desteklenmiş bir genetik algoritma tasarlanmıştır.

IV. Performansa dayalı esneklik kısıtlı yalın üretim hat dengeleme problemi için,

a. problem tanımlanarak literatüre kazandırılmış,

b. problemin 0-1 tamsayılı matematiksel modeli tasarlanmış,

c. problemin NP-tamlık ispatı yapılmış,

d. problem için polinomiyal karmaşıklıkta bir greedy algoritma ve bu greedy ile desteklenmiş bir genetik algoritma hazırlanmıştır.

KAYNAKLAR DİZİNİ

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

Acar, N., 1998, Üretim Planlaması Yöntem ve Uygulamaları, M. P. M. Yayınları No:280.

Baker, B.S. and Coffman, Jr. E.G., 1981, "A tight asymptotic bound for next-fit-decreasing bin packing", SIAM Journal on Algebraic and Discrete Methods, 2:147-152.

Bakır, M.A. ve Altunkaynak, B., 2003, Tamsayılı Programlama: Teori, Modeller ve Algoritmalar, Nobel Basımevi, Ankara, 630s.

Başkaya, Z., 2005, Tamsayılı Programlama Algoritmaları ve Bilgisayar Uygulamalı Problem Çözümleri, Ekin Kitabevi, Ankara, 236s.

Boysen, N., Fliedner, M., Scholl, A., 2008, Assembly line balancing: Which model to use when?, Int. J. Production Economics 111: 509–528.

Coffman, Jr. E.G., Garey, M.R. and Johnson, D.S., 1996, Approximation algorithms for bin-packing: An updated survey, In Hochbaum D. (ed.), "Approximation Algorithms for NP-Hard Problems", PWS Publishing, Boston, 46-93.

Coffman, Jr. E.G., Galambos, G., Martello, S. and Vigo, D., 1998, Bin packing approximation algorithms: Combinatorial analysis, In Du D.Z. and Pardalos P.M. (eds), "Handbook of Combinatorial Optimization", Kluwer Academic Publishers, Boston, MA.

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

Cura, T., 2008, Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayıncılık, İstanbul, 173s.

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

Gen, M., 2006, Genetic Algorithms and Their Applications, Springer Handbook of Engineering Statistic, Springer London, 749-773p.

Glover, F. and Laguna, M., 1997, Tabu Search, Kluwer Academic Publishers, Boston, 182p.

KAYNAKLAR DİZİNİ (devam)

Gu, X.D., Chen, G.L., Gu, J., Huang, L.S. and Jung, Y.J., 2002, Performance analysis and improvement for some linear on-line bin packing algorithms, Journal of Combinatorial Optimization, 6:455-471.

Güner, M., 2001, Konfeksiyon İşletmelerinde Dikim Hattının Dengelenmesi, Konfeksiyon ve Teknik, Temmuz.

Gürsoy, A., 2009, Bidon Paketleme Problemi ve Uygulamaları, Yüksek Lisans Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 38s.

Gürsoy, A., Nuriyev, U.G., 2010, Mathematical Model For Performance-Based Line Balancing Problem, Proceeding of the First International Symposium on Computing in Science & Engineering (ISCSE 2010), Gediz University, p.1286-1290, Kusadasi, Aydin, Turkey.

Gürsoy, A., Nuriyev, U.G., 2010, A Mathematical Model For Assembly Line Balancing Problem With Lean Manufacturing, Proceeding of the third International Conference "Problems of Cybernetics and Informatics, PCI' 2010 ", Vol. III, p. 156 - 159, Baku, Azerbaijan, September 6 - 8.

Gürsoy, A., 2012, An integer model and a heuristic algorithm for the flexible line balancing problem, Tekstil ve Konfeksiyon, Vol. 22(1), pp. 58-63.

Holweg, M., 2007, The genealogy of lean production, Journal of Operations Management, Vol: 25(2), pp: 420–437.

Johnson, D.S., 1973, "Near-optimal bin packing algorithms", Technical Report MAC TR-109, Project MAC, Massachusetts Institute of Technology, Cambridge, MA.

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R. and Graham, R.L., 1974, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM Journal of Computing, 3:299-325.

Kanat, S., Güner, M., 2006, Just in time production system and its feasibility to textile and apparel sector, Tekstil ve Konfeksiyon, Vol: 16(4), pp: 274-278.

Karaboğa, D., 2004, Yapay Zeka Optimizasyon Algoritmaları, Atlas Yayın Dağıtım, İstanbul, 199s.

Kellerer, H., Ulrich, P. and Pisinger, D., 2004, Knapsack Problems, Springer-Verlag, Berlin.

KAYNAKLAR DİZİNİ (devam)

Kenyon, C., 1996, “Best-fit bin-packing with random order”, In Proc. of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 359-364, SIAM.

Korte, B. and Vygen, J., 2008, Combinatorial Optimization: Theory and Algorithms, Springer-Verlag, Berlin.

Martello, S. and Toth, P., 1990, Knapsack Problems, Algorithms and Computer implementations, England, Chichester: John Wiley & Sons.

Malkevitch J., 2012, “Bin Packing” From American Mathematical Society, http://www.ams.org/featurecolumn/archive/bins1.html.

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

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

Nemhauser, G. L. and Woolsey, L. A., 1998, Integer and Combinatorial Optimization, Wiley, New York, 763p.

Nuriyev, U.G., Güner, M., Berberler, M.E. and Gürsoy, A., 2008, Optimum Line Balancing of a Complex Production Process with Consecutive and Parallel Operations in Textile, International Conference on Control and Optimization with Industrial Applications, p.142, Baku, Azerbaijan, 2-4 June.

Nuriyev, U., Guner, M. and Gursoy, A., 2009, A model for determination of optimal production quantity for minimum idle time: a case study in agarment industry, Tekstil, 58, p214-220.

Papadimitriou, C. H., Steiglitz, K., 1998, Combinatorial Optimization: Algorithms and Complexity, Dover Publications Inc., New York, 496p.

Salveson, M.E., 1955. The assembly line balancing problem. The Journal of Industrial Engineering 6 (3), 18–25.

Scholl, A., 1999, Balancing and sequencing assembly lines, second ed. Physica, Heidelberg.

Skiena, S.S., 1997, The Algorithm Design Manual, Springer-Verlag, New York.

Starr, M.K., 1989, Managing Production and Operations, Prentice Hall, New Jersey.

KAYNAKLAR DİZİNİ (devam)

Suri, S., 2012, “Bin Packing Algorithms”, From University of California, http://www.cs.ucsb.edu/~suri/cs130b/BinPacking.txt.

Woolsey, L. A., 1998, Integer programming. Wiley, New York, 264p.

ÖZGEÇMİŞ

17.11.1983 tarihinde Edirne'nin Havsa ilçesinde dünyaya gelen Arif GÜRSOY, İlk ve orta öğrenimini sırasıyla İzmir Anafartalar İlkokulu, İzmir Alsancak Ortaokulu ve İzmir Karataş Lisesi'nde tamamladı. Ardından 2000 yılında Ege Üniversitesi Fen Fakültesi Matematik Bölümünü kazandı. Bir yıl süren yabancı dil hazırlık eğitimiyle birlikte 2005 yılında Matematik Bölümü - Bilgisayar Bilimleri Opsiyonundan mezun oldu. 2005-2009 yılları arasında Ege Üniversitesi Fen Bilimleri Enstitüsü Matematik Anabilim Dalı Bilgisayar Bilimleri Bilim Dalında "Bidon Paketleme Problemi ve Uygulamaları" isimli tez çalışmasıyla Yüksek Lisans eğitimini başarılı bir şekilde tamamladı. 2009 yılında yine Ege Üniversitesi Fen Bilimleri Enstitüsü Matematik Anabilim Dalı Bilgisayar Bilimleri Bilim Dalı'nda Doktora eğitimine başladı ve 2010 yılında doktora yeterliliğini tamamlayarak "Bazı Ayrık Optimizasyon Problemlerinin Modellenmesi ve Çözüm Yöntemleri Üzerine" isimli tez çalışmasına başladı. 2006 yılından beri Ege Üniversitesi Fen Fakültesi Matematik Bölümde olarak görev yapmaktadır.

YAYINLAR

Makaleler

A) Uluslararası Science Citation Index ve Sosial Science Citation Indexce taranan dergilerde yayınlanmış makaleler

1. **Gürsoy, A.**, 2012, "An integer model and a heuristic algorithm for the flexible line balancing problem", Tekstil ve Konfeksiyon, Vol. 22(1), pp. 58-63.
2. Nuriyev U., Güner M., **Gürsoy A.**, 2009, "A model for determination of optimal production quantity for minimum idle time: a case study in agarment industry", Tekstil, Vol. 58(5), pp. 214-220.

B) Diğer bilimsel dergilerde yayınlanmış makaleler

1. **Gürsoy A.**, Nuriyev, U.G., 2012, "Genetic Algorithm for Multi Bandpass Problem and Library of Problems", Proceeding of the fourth International Conference "Problems of Cybernetics and Informatics", Section I, " Information and communication technologies", p. 33 - 37, Baku, Azerbaijan, 12-14 September.

2. **Gürsoy A.**, Nuriyev U.G., 2010, "A Mathematical Model For Assembly Line Balancing Problem With Lean Manufacturing", *Proceeding of the third International Conference "Problems of Cybernetics and Informatics, PCI' 2010"*, Vol. III, p. 156 - 159, Baku, Azerbaijan, September 6 - 8.
3. Berberler M. E., **Gürsoy A.**, 2010, "Genetic Algorithm Approach for Bandpass Problem", Proceeding of the 24-th Mini Euro Conference on Continuous Optimization and Information-Based Technologies in the Financial Sector (MEC EurOPT 2010), Selected papers, VGTU Publishing House "Technika", pp 201-206.
4. **Gürsoy A.**, Nuriyev U.G., 2010, "Mathematical Model For Performance-Based Line Balancing Problem", Proceeding of the First International Symposium on Computing in Science &Engineering (ISCSE 2010), Gediz University, p.1286-1290, Kusadasi, Aydin, Turkey.
5. Nuriyev U.G., **Gürsoy A.**, 2006, Derman C., "En Kısa Uzunluklu Hamilton Yolu Problemi Üzerine", *XIX. Ulusal Matematik Sempozyumu Bildiriler Kitabı,* s. 666 – 672, 22-25 Ağustos, Kütahya, Dumlupınar Üniversitesi.
6. Güner M., **Gürsoy A.**, Öksüz D., Çukur R., 2006, "Süreç Planlama ve Kontrol için Takt Zamana Göre AHP ve GANTT Planı'nın Kombineli Kullanımı ile Hazırlanan Bir Yazılımın Geliştirilmesi", *Yöneylem Araştırması ve Endüstri mühendisliği XXVI Ulusal kongresi*, *Bildiriler kitabı*, s. 441-444, İzmit, Kocaeli Üniversitesi.
7. Nuriyev U.G., Berberler M. E., **Gürsoy A.**, 2006, "Computer Assisted Teaching of Multi-Focus Curves", *Proceeding of the International scientific conference "Information Technologies and Telecommunications in Education and Science" (IT & T ES'2006)*, p.189-191, Antalya, Turkey.
8. Nuriyev U.G., Berberler M.E., **Gürsoy A.**, 2006, "An Approach to The Elliptical Orbit With The Lines Of Equal Length", *Proceeding of the International scientific conference "Information Technologies and Telecommunications in Education and Science" (IT & T ES'2006)*, p. 192-194, Antalya, Turkey.

Kongreler

A) Uluslararası bilimsel toplantılarda yayınlanmış bildiriler:

1. **Gürsoy A.**, Nuriyev, U.G., "Genetic Algorithm for Multi Bandpass Problem and Library of Problems", *Proceeding of the fourth International Conference "Problems of Cybernetics and Informatics"*, *Section I, " Information and communication technologies"* , p. 33 - 37, Baku, Azerbaijan, September 12-14, 2012.
2. **Gürsoy A.**, Kurt M., Berberler M.E., Nuriyev U G., "On the solution approaches of the combinatorial bandpass problem in telecommunication", *25th Conference of European Chapter on Combinatorial Optimization, (ECCO XXV)*, Antalya, Turkey, April 26 – 28, 2012.
3. Ordin B., **Gürsoy A.**, "Genetic incremental clustering algorithm", *25th Conference of European Chapter on Combinatorial Optimization, (ECCO XXV)*, Antalya, Turkey, April 26 – 28, 2012.

4. **Gürsoy A.**, Nuriyev U.G., "A Mathematical Model For Assembly Line Balancing Problem With Lean Manufacturing", Proceeding of the third International Conference "Problems of Cybernetics and Informatics", V.III, p. 156 - 159, Baku, Azerbaijan, September 6-8, 2010.
5. Berberler M. E., **Gürsoy A.**, "Genetic Algorithm Approach for Bandpass Problem", Proceeding of the 24-th Mini Euro Conference on Continuous Optimization and Information-Based Technologies in the Financial Sector (MEC EurOPT 2010), June 23–26, - Izmir, TURKEY, pp 201-206, 2010.
6. **Gürsoy A.**, Nuriyev U.G., "Mathematical Model For Performance-Based Line Balancing Problem", Proceeding of the First International Symposium on Computing in Science &Engineering (ISCSE 2010), Gediz University, p.1286-1290, Kusadasi, Aydin, Turkey, June 3-5, 2010.
7. Babayev, D.A., Nuriyev, U.G., Bell, G.I., Berberler, M.E., Kutucu, H., **Gürsoy, A.**, Kurt, M. "Mathematical modelling of a telecommunication packing problem and a heuristic approach for finding a solution", *International Conference on Control and Optimization with Industrial Applications*, p.44, Baku, Azerbaijan, June 2-4, 2008.
8. Nuriyev, U.G., Güner, M., Berberler, M.E., **Gürsoy, A.** "Optimum Line Balancing of a Complex Production Process with Consecutive and Parallel Operations in Textile", *International Conference on Control and Optimization with Industrial Applications*, p.142, Baku, Azerbaijan, June 2-4, 2008.
9. Nuriyev U.G., **Gürsoy A.**, Dinler Y. "Multi-Focused Curves", *First International Computer and Educational Technology Symposium*, p. 91, 16-18 May 2007, Çanakkale, 2007.
10. Nuriyev U.G., Berberler M. E., **Gürsoy A.** "Computer Assisted Teaching of Multi-Focus Curves", *International scientific conference "Information Technologies and Telecommunications in Education and Science" IT & T ES'2006)*, p.189-191, Antalya, Turkey, May 19-26, 2006.
11. Nuriyev U., Berberler M.E., **Gürsoy A.** "An Approach to The Elliptical Orbit with the Lines of Equal Length", *International scientific conference "Information Technologies and Telecommunications in Education and Science" (IT & T ES'2006)*, p.192-194, Antalya, Turkey, May 19-26, 2006.

B) Ulusal bilimsel toplantılarda yayınlanmış bildirileri:

1. Nuriyev, U.G., Berberler, M.E., Kurt, M., **Gürsoy, A.**, Kutucu, H., "Bir Telekomünikasyon Probleminin Matematiksel Modellenmesi Üzerine", XVI. Türkiye'de ˙Internet Konferansı, inet-tr'11, Ege Üniversitesi, İzmir, Türkiye, 30 Kasım-02 Aralık, 2011
2. **Gürsoy A.**, Nuriyev U., Güner M., Ünal C., Solaklar F., "Tekstil İşletmelerinde Departman Yerleşimi Problemi", YAEM 2009 - *Yöneylem Araştırması ve Endüstri mühendisliği 29. Ulusal Kongresi,* Ankara, Bilkent Üniversitesi, 22–24 Haziran 2009.
3. Nuriyev U.G., **Gürsoy A.**, Berberler M.E. "Bantgeçişi (Bandpass) Problemini Çözmek İçin Heuristik Bir Algoritma", *Yöneylem Araştırması ve Endüstri mühendisliği XXVII Ulusal Kongresi*, İzmir, Dokuz Eylül Üniversitesi, 2– 4 Temmuz, 2007.

4. İlleez A, Nuriyev U.G., Güner M., **Gürsoy A.** "Mathematical Methods Used for Process Scheduling in Apparel Sector", *Second Annual SUNY-YÖK Collaboration Symposium "Scientific Collaboration for Sustainable Development"* 23-25 May 2007, Çukurova University Balcalı/ADANA, 2007.
5. Nuriyev U.G., **Gürsoy A.**, 2006, Derman C., "En Kısa Uzunluklu Hamilton Yolu Problemi Üzerine", *XIX. Ulusal Matematik Sempozyumu Bildiriler Kitabı,* s. 666 – 672, 22-25 Ağustos, Kütahya, Dumlupınar Üniversitesi.
6. Nuriyev U.G., Güner M., Berberler M.E., **Gürsoy A.**, Vatansever B. "Yalın Üretim Tekniği Unsurları Dikkate Alınarak Hat Dengelemede Kayıp Zamanların En Az Olduğu Üretim Miktarının Belirlenmesi", *Yöneylem Araştırması ve Endüstri Mühendisliği XXV. Ulusal kongresi*, s.19-20, İstanbul, Koç Üniversitesi, 4 – 6 Temmuz, 2005.

Projeler

1. Nuriyev U.G., Güner M., **Gürsoy A.,** Berberler M.E., Tekstilde Ardışık ve Paralel İşlemleri Karmaşık Bir Üretim Sürecinin Optimal Hat Dengelenmesi: Matematiksel olarak Modelleme ve Yeni Bir Yazılım Modülü Geliştirme, **TÜBİTAK, Bilimsel ve Teknolojik Araştırma Projelerini Destekleme Programı (1001) Projesi**, Proje No: **107T543**, 2007-2009, Bursiyer.
2. Nuriyev U.G., Berberler M.E., **Gürsoy A.** "Ulaşım Sektöründe Bir Boyutlu Uygun Yer Seçimi Problemi İçin Yazılım Geliştirme", ***Proje pazarı 2008***, Düzenleyen Kuruluş: İzmir Yüksek Teknoloji Enstitüsü - Endüstriyel İlişkiler Yönetim Birimi, İzmir; 18, 25 Haziran 2008.
3. Nuriyev U.G., Güner M., Berberler M.E., **Gürsoy A.** "Yalın Üretim Hat Dengeleme, Model Değişim Süreçlerinin Optimizasyonu, Üretim Sürecinde Planlama Problemlerinin Çözümü, Yazılım Geliştirilmesi ve Tekstil Sektöründe Uygulanması", ***Proje pazarı 2008***, Düzenleyen Kuruluş: İzmir Yüksek Teknoloji Enstitüsü - Endüstriyel İlişkiler Yönetim Birimi, İzmir; 18, 25 Haziran 2008.
4. Nuriyev U.G., Berberler M.E., **Gürsoy A.** "Bir Boyutlu Kesme Problemi İçin Bir Paket Program Hazırlanması Ve PVC, Metal Ve Kamu Sektörlerinde Uygulanması", ***Proje pazarı 2008***, Düzenleyen Kuruluş: İzmir Yüksek Teknoloji Enstitüsü - Endüstriyel İlişkiler Yönetim Birimi, İzmir; 18, 25 Haziran 2008.
5. Güner M., Nuriyev U., Berberler M. E., **Gürsoy A.** "Tekstilde Ardışık ve Paralel İşlemli Karmaşık Bir Üretim Sürecinin Optimal Hat Dengelemesi: Matematiksel Olarak Modelleme ve Yeni Bir Yazılım Modülü Geliştirme", **Türkiye Tekstil Teknoloji Platformu (TTTP) - Bilgi Paylaşımı ve Proje Ortaklıkları Oluşturma Toplantısı**, Düzenleyen Kuruluş: İstanbul Tekstil ve Konfeksiyon İhracatçı Birlikleri, İstanbul, 21 Mayis 2008.
6. Nuriyev U.G., Berberler M.E., **Gürsoy A.**, Kaya O. "Ulaşım Sektöründe Bir Boyutlu Uygun Yer Seçimi Problemi İçin Yazılım Geliştirme", ***"Inno-Venture 2007" Proje pazarı***, Düzenleyen Kuruluşlar: TÜBİTAK, EBİLTEM, TURKCELL (Proje No BIT_206), İzmir, 13-14 Aralık 2007.
7. Nuriyev U.G., Berberler M.E., **Gürsoy A.**, Kaya O. "Yalın Üretim Hat Dengeleme, Model Değişim Süreçlerinin Optimizasyonu, Üretim Sürecinde Planlama Problemlerinin Çözümü, Yazılım Geliştirilmesi ve Tekstil Sektöründe Uygulanması", ***"Inno-Venture 2007" Proje pazarı***, Düzenleyen Kuruluşlar: TÜBİTAK, EBİLTEM, TURKCELL (Proje No BIT_207), İzmir, 13-14 Aralık 2007.

8. Nuriyev U.G., Berberler M.E., **Gürsoy A.**, Kaya O., "Bir Boyutlu Kesme Problemi İçin Bir Paket Program Hazırlanması ve PVC, Metal Ve Kamu Sektörlerinde Uygulanması", ***"Inno-Venture 2007" Proje pazarı***, Düzenleyen Kuruluşlar: TÜBİTAK, EBİLTEM, TURKCELL (Proje No BIT_208), İzmir, 13-14 Aralık 2007.

9. Güner M., **Gürsoy A.**, İlleez A.A., Nuriyev U.G. "Dikim Ünitelerinde Hazırlık Sürelerinin Optimizasyonu İçin İş Çizelgeleme Programı", ***"Inno-Venture 2007" Proje pazarı***, Düzenleyen Kuruluşlar: TÜBİTAK, EBİLTEM, TURKCELL (Proje No BIT_210), İzmir, 13-14 Aralık 2007.

10. Bidon Paketleme Problemi ve Uygulamaları, **Ege Üniversitesi Araştırma Fonu Projesi**, 2007/Fen/007, **Proje Araştırmacısı**, 2007

11. Nuriyev U.G., Güner M., Berberler M. E., **Gürsoy A.**, Vatansever B. "Yalın Üretim Unsurları ve Minimum Kayıp Zaman Hedefi ile İdeal Üretim Değerlerini Belirleyen Modüler Hat Dengeleme Paket Programı", ***Teknoloji, Yenilik & Girişim Sermayesi Günleri ve Proje Pazarı***, Düzenleyen Kuruluşlar: ESBAS, TÜBİTAK, EBİLTEM (Proje No 409), İzmir, 1-2 Aralık 2005.

Ek – Esnek Yalın Üretim Hat Dengeleme Programının C# Programlama Dili Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace WindowsFormsApplication1_yuhad
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        public void ekle(object[] ob) {
            dataGridView1.Rows.Add(ob);
            this.Text = dataGridView1.RowCount.ToString();
        }
        private void button1_Click(object sender, EventArgs
e)//VeriGirişi
        {   VeriGiris f_vg1;
            f_vg1 = new VeriGiris(this);
            f_vg1.MaximizeBox = false;
            f_vg1.ShowDialog();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            label1.Text = "p".PadLeft(4) + "∑ id".PadLeft(8) + "∑
C".PadLeft(8) + "∑ li".PadLeft(8) +
                "per Op.".PadLeft(10) + "ef %".PadLeft(6);
            dataGridView1.AutoResizeColumns();
        }

        private void button2_Click(object sender, EventArgs
e)//Kaydet/
        {
            saveFileDialog1.Filter = "Operasyon
Dosyaları(*.opd)|*.opd|" +  "Bütün dosyalar |*.*";
            saveFileDialog1.DefaultExt = "opd";
            saveFileDialog1.Title = "Kaydedilicek dosya";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                StreamWriter d = new
StreamWriter(saveFileDialog1.FileName);

                int i, j;
```

```
                string str_tut;
                for (i = 0; i < dataGridView1.RowCount - 1; i++)
                {
                    str_tut = "";
                    for (j = 0; j < 7; j++)
                    {
                        str_tut += dataGridView1[j,
i].Value.ToString();
                        if (j < 6) str_tut += ";";
                    }
                    d.WriteLine(str_tut);
                }
                d.Close();
            }

        }

        private void button3_Click(object sender, EventArgs
e)//DosyaAç
        {
            dataGridView1.Rows.Clear();

            openFileDialog1.Filter = "Operasyon
Dosyaları(.opd)|*.opd|" +  "Bütün dosyalar |*.*";
            openFileDialog1.Title = "Açılacak dosya";
            openFileDialog1.FileName = "Bir dosya seçin!";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                using (StreamReader reader = new
StreamReader(openFileDialog1.FileName))
                {
                    string line;
                    while ((line = reader.ReadLine()) != null)
                    {
                        object[] parts = line.Split(';');
                        dataGridView1.Rows.Add(parts);
                    }
                }
            }
            dataGridView1.AutoResizeColumns();


        }

        class Operasyon {
            public object[] bilgi; //0 no; 1 ad; 2 sure; 3 mak; 4
es; 5 es_bas; 6 es_bit; 7 grup;
            public Operasyon(DataGridViewRow dgvr) {
                this.bilgi = new object[] {
                    dgvr.Cells[0].Value, //no
                    dgvr.Cells[1].Value,//ad
                    dgvr.Cells[2].Value,//sure
                    dgvr.Cells[3].Value,//mak
                    dgvr.Cells[4].Value,//es
                    dgvr.Cells[5].Value,//es_bas
                    dgvr.Cells[6].Value,//es_bit
                    -1                  //grup
```

```
                };
            }
            public string birlestir() {
                string str_tumu = ""; int i;
                int[] p_dz = new int[] { 3, 30, 4, 20, 6, 4, 4, 4
};
                for (i = 0; i < 7; i++) {
                    str_tumu +=
this.bilgi[i].ToString().PadRight(p_dz[i]);
                }
                return str_tumu;
            }
            public void Temizle() { this.bilgi[7] = -1; }
        }
        class Grup {
            public int l;//atanan işçi sayısı
            public int C;//atanan operasyonların yükü
            public ArrayList lst = new ArrayList();//atanan
operasyonlar
            public Grup() {
                this.l = 1;
                this.C = 0;
                this.lst.Clear();
            }
            public void resize_wg(int num) { this.l = num;
}//resize work group
            public string yaz_lst()
            {
                string str_tumu = "";
                int i;
                for (i = 0; i < this.lst.Count; i++)
                    str_tumu += this.lst[i].ToString() + "; ";
                return str_tumu.PadLeft(20);
            }
            public bool varmi(int no) {
                int i;
                for (i = 0; i < this.lst.Count; i++) {
                    if (no == Convert.ToInt32(this.lst[i]))
return true;
                }
                return false;
            }
            public string yaz_C() {
                return Math.Round(Convert.ToDouble(this.C / 100),
2).ToString().PadLeft(8);
            }
        }
        class Sonuc {
            public int t_l,p;
            public double t_id, t_yzd_ef, t_C;
            public Sonuc() {
                this.t_l= 0;//total operators
                this.t_id = 0.0;//total idle time
                this.t_C = 0.0;//total work time
            }
            public Sonuc(int p, int t_l, double t_id, double t_C,
double t_yzd_ef) {
                this.p = p;
```

```
                this.t_l = t_l;
                this.t_id = t_id;
                this.t_C = t_C;
                this.t_yzd_ef = t_yzd_ef;
            }
            public string yaz() {
                return (this.p.ToString().PadLeft(4) +
this.t_id.ToString().PadLeft(8) + this.t_C.ToString().PadLeft(8) +
this.t_l.ToString().PadLeft(8) +
                    Math.Round(this.t_C / this.t_l,
2).ToString().PadLeft(10) + this.t_yzd_ef.ToString().PadLeft(6));
            }
            public string yaz2()
            {
                return (this.p.ToString() + "\t" +
this.t_id.ToString() + "\t" + this.t_C.ToString() + "\t" +
this.t_l.ToString() + "\t" +
                    Math.Round(this.t_C / this.t_l, 2).ToString()
+ "\t" + this.t_yzd_ef.ToString());
            }

        }
        ArrayList urun = new ArrayList();//tüm operasyonlar
        ArrayList urun_e = new ArrayList();//urun esnek:esnek
operasyon
        Operasyon opr_j;

        ArrayList ekip = new ArrayList();
        Grup gr_i;

        public int p_alt, p_ust, p, U;//üretim mik,mesai(cdk),
mesai+fazla mesai(cdk).
        public double ef;
        public int T, ot;//mesai(cdk), fazla mesai(cdk)
        //p, product number;
        //T, common daily work time;
        //ot, daily overtime;
        //l, number of operators of work group i;
        //U, upper work time of current work group;
        //C, workload of the work group i;
        //i, group index;
        //j, operation index;
        //pt, current operation’s production time (p·t_j); (cdk)

        //n, number of operations;
        //m, number of work groups after running procedure P1;
        //t_j, unit time of operation j;
        //id, idle time.
        private void Balancing_bounds() {
            try { p_alt = Convert.ToInt32(textBox1.Text); }
            catch (Exception ex)
            {
                MessageBox.Show("Üretim Alt Miktarı - Error: " +
ex.Message);
                textBox1.Focus();
                textBox1.SelectAll();
                return;
            }
```

```
            try { p_ust = Convert.ToInt32(textBox2.Text); }
            catch (Exception ex)
            {
                MessageBox.Show("Üretim Üst Miktarı - Error: " +
ex.Message);
                textBox2.Focus();
                textBox2.SelectAll();
                return;
            }

            try { T = Convert.ToInt32(textBox3.Text); }
            catch (Exception ex)
            {
                MessageBox.Show("Mesai Süresi - Hata:" +
ex.Message);
                textBox3.Focus();
                textBox3.SelectAll(); return;
            }

            try { ot = Convert.ToInt32(textBox4.Text); }
            catch (Exception ex)
            {
                MessageBox.Show("Fazla Mesai Süresi - Hata:" +
ex.Message);
                textBox4.Focus();
                textBox4.SelectAll(); return;
            }
            try { ef = Convert.ToDouble(textBox5.Text) / 100; }
            catch (Exception ex)
            {
                MessageBox.Show("Efficiency lower limit - Error:"
+ ex.Message);
                textBox5.Focus();
                textBox5.SelectAll(); return;
            }
        }

        private void button4_Click(object sender, EventArgs
e)//HatDengele
        {
            Balancing_bounds();

            T *= 100;
            ot *= 100;
            int j, i, cev;
            urun.Clear();
            urun_e.Clear();
            listBox1.Items.Clear();
            listBox1.Items.Add("no".PadRight(3) +
"ad".PadRight(30) + "t".PadRight(4) + "mak".PadRight(20) +
"es".PadRight(6) + "bas".PadRight(4) + "bit".PadRight(4) +
"grup".PadRight(4));
            for (j = 0; j < dataGridView1.RowCount - 1; j++)
            {
                urun.Add(new Operasyon(dataGridView1.Rows[j]));
                listBox1.Items.Add((urun[j] as
Operasyon).birlestir());
```

```
                if
(Convert.ToBoolean(dataGridView1.Rows[j].Cells[4].Value) == true)
urun_e.Add(new Operasyon(dataGridView1.Rows[j]));
            }
            string duzelt = Es_Kontrol();
            if (duzelt != null)
                    MessageBox.Show("Esnek operasyonlar esneklik
Baş. ve Bitişi için sabitlere referans verebilir!\r\n" + duzelt);
            progressBar1.Value = 0;
            progressBar1.Maximum = p_ust - p_alt+1;

            Sonuc s1, s = new Sonuc();
            double id;

            listBox1.Items.Add(" ");
            listBox1.Items.Add("p".PadLeft(4) + "∑ id".PadLeft(8)
+ "∑ C".PadLeft(8) + "∑ li".PadLeft(8) +
                "per Op.".PadLeft(10) + "ef %".PadLeft(6));
            for (p = p_alt; p <= p_ust; p++) {
                progressBar1.Value = progressBar1.Value + 1;
                U = (T + ot);
                for (j = 0; j < urun.Count; j++) {
                    opr_j = urun[j] as Operasyon;
                    opr_j.Temizle();
                }
                for (j = 0; j < urun_e.Count; j++) {
                    opr_j = urun_e[j] as Operasyon;
                    opr_j.Temizle();
                }

                ekip.Clear();//Eski grup atamalarını sıfırlar

                cev = Yerlestir(0);//İlk yerleştirme
                if (cev != -1) {
                    MessageBox.Show("İlk Yerleştirme
başarıSIZ!:/n/rp= " + p.ToString() + "; opr info=" + (urun[cev] as
Operasyon).birlestir()); ;
                    return;
                }

                cev = Es_Yerlestir(0);//2. yerleştirme
                if (cev != -1)
                {
                    MessageBox.Show("2. Yerleştirme
başarıSIZ!:/n/rp= " + p.ToString() + "; opr info=" + (urun_e[cev]
as Operasyon).birlestir());
                    return;
                }

                s1 = new Sonuc();

                for (i = 0; i < ekip.Count; i++)
                {
                    gr_i = (ekip[i] as Grup);
                    id = Math.Round(Convert.ToDouble((gr_i.l * (T
+ ot) - gr_i.C) / 100), 2);
                    s1.t_l += gr_i.l;
                    s1.t_id += id;
```

```
                    s1.t_C += Math.Round(Convert.ToDouble(gr_i.C /
100), 2);//min
                }
                s1.t_yzd_ef = Math.Round(s1.t_C / (s1.t_l * (T+ot)
/ 100),2);
                if (s1.t_yzd_ef < ef) continue;
                s1.p = p;
                if (s1.t_yzd_ef >= s.t_yzd_ef) s = s1;
                listBox1.Items.Add(s1.yaz());

            }
            listBox1.Items.Add("-");
            listBox1.Items.Add("Best:");
            listBox1.Items.Add(s.yaz2());


        }//HatDengele

        private string Es_Kontrol() {
            //esnek operasyonlar bas ve bit olarak sabitleri
işaret etmeli
            int j, tut;
            string duzelt = null;
            Operasyon opr_j, opr_j1;
            for (j = 0; j < urun_e.Count; j++) {
                opr_j = urun_e[j] as Operasyon;//5 6

                tut = Convert.ToInt32(opr_j.bilgi[5]);
                opr_j1 = urun[tut-1] as Operasyon;
                if (Convert.ToBoolean(opr_j1.bilgi[4]) == true)
duzelt += opr_j.bilgi[5].ToString() + "; ";

                tut = Convert.ToInt32(opr_j.bilgi[6]);
                opr_j1 = urun[tut-1] as Operasyon;
                if (Convert.ToBoolean(opr_j1.bilgi[4]) == true)
duzelt += opr_j.bilgi[6].ToString() + "; ";
            }
            return duzelt;
        }//Es_kontrol

        private int Es_Yerlestir(int bas)
        {
            int j,i;
            bool ypld;//operasyon yerleştirilebildi mi?
            int pt;//operasyon üretim süresi: p*st (cdk)

            for (j = bas; j < urun_e.Count; j++)
            {
                opr_j = (urun_e[j] as Operasyon);
                pt = p * Convert.ToInt32(opr_j.bilgi[2]);
                if (pt > 3 * (T + ot))
                {
                    MessageBox.Show("3 kişiden fazla aynı
operasyon ataması!" +
                        "\n\rÜrün Mik: " + p.ToString() +
                        "\n\roperasyon üretim süresi: " +
pt.ToString() +
                        "\n\r" + opr_j.birlestir());
```

```
                    return j;
                }

                ypld = false;
                for (i = 0; i < ekip.Count; i++) {
                    gr_i= (ekip[i] as Grup);
                    if (gr_i.C + pt <= gr_i.l * U)
                    {
                        gr_i.C += pt;
                        gr_i.lst.Add(opr_j.bilgi[0]);//Gruba
operasyon atandı
                        opr_j.bilgi[7] = i;//Operasyona grup
atandı
                        ypld = true;
                        break;
                    }
                }
                if (ypld) continue;

                gr_i = new Grup(); //yeni grup oluşturuldu

                if (T + ot < pt && pt <= 2 * (T + ot))
gr_i.resize_wg(2);
                else if (2 * (T + ot) < pt && pt <= 3 * (T + ot))
gr_i.resize_wg(3);
                gr_i.C += pt;
                gr_i.lst.Add(opr_j.bilgi[0]);//Gruba operasyon
atandı
                opr_j.bilgi[7] = ekip.Count;//Operasyona grup
atandı
                ekip.Add(gr_i);//yeni grubu ekibe ekle
            }//for j:esnek operasyonları yerleştir
            return -1;
        }//Es_yerleştir

        private int Yerlestir(int bas) {
            int j;
            int pt;//operasyon üretim süresi: p*st

            gr_i = new Grup(); //ekip.Add(gr_i);//Başlangıç grubu

            for (j = bas; j < urun.Count; j++)
            {
                opr_j = (urun[j] as Operasyon);
                pt = p * Convert.ToInt32(opr_j.bilgi[2]);
                if (pt > 3 * (T + ot))
                {
                    MessageBox.Show("3 kişiden fazla aynı
operasyon ataması!" +
                        "\n\rÜrün Mik: " + p.ToString() +
                        "\n\roperasyon üretim süresi: " +
pt.ToString() +
                        "\n\r" + opr_j.birlestir());
                    return j;
                }

                if (Convert.ToBoolean(opr_j.bilgi[4]) == true)
continue;//Esnek operasyonlar dikkate alınmıyor
```

```
            Yeniden:

                if (pt <= T + ot)
                {
                    if (gr_i.C + pt <= U)
                    {
                        gr_i.C += pt;
                        gr_i.lst.Add(opr_j.bilgi[0]);//Gruba
operasyon atandı
                        opr_j.bilgi[7] = ekip.Count;//Operasyona
grup atandı
                    }
                    else//operasyon gruptan taştı.
                    {
                        ekip.Add(gr_i); gr_i = new Grup(); U = (T
+ ot); goto Yeniden;
                    }
                }
                else if (pt <= 2 * (T + ot))
                {
                    U = 2 * (T + ot);//diğer ifleri etkiler,
değiştirme
                    if (gr_i.C + pt <= U)
                    {
                        gr_i.resize_wg(2);//Grubun işçi sayısı 2
oldu
                        gr_i.C += pt;
                        gr_i.lst.Add(opr_j.bilgi[0]);//Gruba
operasyon atandı
                        opr_j.bilgi[7] = ekip.Count;//Operasyona
grup atandı
                    }
                    else//operasyon gruptan taştı.
                    {
                        ekip.Add(gr_i); gr_i = new Grup(); U = (T
+ ot); goto Yeniden;
                    }
                }
                else if (pt <= 3 * (T + ot))
                {
                    U = 3 * (T + ot);//diğer ifleri etkiler,
değiştirme
                    if (gr_i.C + pt <= U)
                    {
                        gr_i.resize_wg(3);//Grubun işçi sayısı 3
oldu
                        gr_i.C += pt;
                        gr_i.lst.Add(opr_j.bilgi[0]);//Gruba
operasyon atandı
                        opr_j.bilgi[7] = ekip.Count;//Operasyona
grup atandı
                    }
                    else//operasyon gruptan taştı.
                    {
                        ekip.Add(gr_i); gr_i = new Grup(); U = (T
+ ot); goto Yeniden;
                    }
```

```
                }
            }//for i:tüm operasyonları yerleştir
            ekip.Add(gr_i);
            return -1;//dengeleme başarılı
        }//yerleştir

        private void button5_Click(object sender, EventArgs
e)//SonuçKaydet
        {
            saveFileDialog1.Filter = "*.xls|*.xls|" + "Bütün
dosyalar |*.*";
            saveFileDialog1.DefaultExt = "xls";
            saveFileDialog1.Title = "Kaydedilicek Çıktı Dosyası";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                StreamWriter d = new
StreamWriter(saveFileDialog1.FileName);
                int i;
                for (i = 0; i < listBox1.Items.Count; i++)
                    d.WriteLine((listBox1.Items[i] as string));
                d.Close();
            }
        }

        private void listBox1_MouseClick(object sender,
MouseEventArgs e)
        {
            try
            {
                textBox6.Text = (listBox1.SelectedItem as
string).Substring(0, 4);
                textBox6.Text.Trim();
            }
            catch
            {
            }

        }

        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                int i, j, cev;
                p = Convert.ToInt32(textBox6.Text);
                U = (T + ot);
                for (j = 0; j < urun.Count; j++)
                {
                    opr_j = urun[j] as Operasyon;
                    opr_j.Temizle();
                }
                for (j = 0; j < urun_e.Count; j++)
                {
                    opr_j = urun_e[j] as Operasyon;
                    opr_j.Temizle();
                }

                ekip.Clear();//Eski grup atamalarını sıfırlar
```

```
                cev = Yerlestir(0);//İlk yerleştirme
                cev = Es_Yerlestir(0);//2. yerleştirme
                double id;
                Sonuc s1 = new Sonuc();
                s1.p = p;
                listBox1.Items.AddRange(new string[] { " ", "-",
"No".PadLeft(3) + "li".PadLeft(3) + "Operations".PadLeft(20) +
"C".PadLeft(8) });
                for (i = 0; i < ekip.Count; i++)
                {
                    gr_i = (ekip[i] as Grup);
                    id = Math.Round(Convert.ToDouble((gr_i.l * (T
+ ot) - gr_i.C) / 100), 2);
                    s1.t_l += gr_i.l;
                    s1.t_id += id;
                    s1.t_C += Math.Round(Convert.ToDouble(gr_i.C /
100), 2);//min
                    listBox1.Items.Add((i +
1).ToString().PadLeft(3) + gr_i.l.ToString().PadLeft(3) +
gr_i.yaz_lst() + gr_i.yaz_C());
                }
                s1.t_yzd_ef = Math.Round(s1.t_C / (s1.t_l * (T +
ot) / 100), 2);
                listBox1.Items.Add(" ");
                listBox1.Items.Add("p".PadLeft(4) + "∑
id".PadLeft(8) + "∑ C".PadLeft(8) + "∑ li".PadLeft(8) +
                    "per Op.".PadLeft(10) + "ef %".PadLeft(6));
                listBox1.Items.Add(s1.yaz());
            }
            catch
            {
            }
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1_yuhad
{
    public partial class VeriGiris : Form
    {

        public VeriGiris(Form1 _f1) {
            InitializeComponent();
            form1 = _f1;
        }
        Form1 form1;

        private void checkBox1_CheckedChanged(object sender,
EventArgs e)
        {
            if (checkBox1.Checked)
            {
                groupBox1.Visible = true;
                textBox4.Focus();
            }
            else
            {
                groupBox1.Visible = false;
                textBox4.Text = "";
                textBox5.Text = "";
            }
        }

        public int op_say = 0;

        object[] ob = new object[7];
        private void button1_Click(object sender, EventArgs e)
        {//ekle
            ob[0] = ++op_say;
            ob[1] = textBox1.Text;
            ob[2] = textBox2.Text;
            ob[3] = textBox3.Text;
            ob[4] = checkBox1.Checked;
            ob[5] = textBox4.Text;
            ob[6] = textBox5.Text;
            form1.ekle(ob);
            for (int i = 0; i < 3;i++ )
                str_autofill[i].Add(ob[i+1].ToString());
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            checkBox1.Checked = false;
            form1.dataGridView1.AutoResizeColumns();
            this.Text = (op_say).ToString() + ". operasyon
bilgileri";
```

```
            textBox1.Focus();
        }

        Dictionary<string, int> d;
        Control[] cnt;
        AutoCompleteStringCollection[] str_autofill = new
AutoCompleteStringCollection[3];
        private void Form2_Load(object sender, EventArgs e)
        {
            this.Text = (op_say + 1).ToString() + ". operasyon
bilgileri";
            cnt = new Control[] { textBox1, textBox2, textBox3,
checkBox1};
            d = new Dictionary<string, int>()
            {
                {"textBox1",0}, {"textBox2",1}, {"textBox3",2},
{"checkbox1",3}
            };
            int i;
            for (i = 0; i < 3; i++)
            {
                str_autofill[i] = new
AutoCompleteStringCollection();
                ((TextBox)cnt[i]).AutoCompleteCustomSource =
str_autofill[i];
                ((TextBox)cnt[i]).AutoCompleteMode =
AutoCompleteMode.SuggestAppend;
                ((TextBox)cnt[i]).AutoCompleteSource =
AutoCompleteSource.CustomSource;
            }

            textBox1.KeyDown += new
System.Windows.Forms.KeyEventHandler(enterabasildi);
            textBox2.KeyDown += new
System.Windows.Forms.KeyEventHandler(enterabasildi);
            textBox3.KeyDown += new
System.Windows.Forms.KeyEventHandler(enterabasildi);
        }

        private void enterabasildi(object sender, KeyEventArgs e)
        {
            if (e.KeyValue == 13) cnt[d[((Control)sender).Name] +
1].Focus();
        }

        private void checkBox1_KeyDown(object sender, KeyEventArgs
e)
        {
            if (e.KeyValue == 13) button1.Focus();
        }

        private void textBox4_KeyDown(object sender, KeyEventArgs
e)
        {
            if (e.KeyValue == 13) textBox5.Focus();
        }
```

```
        private void textBox5_KeyDown(object sender, KeyEventArgs
e)
        {
            if (e.KeyValue == 13) button1.Focus();
        }
    }
}
```