

**T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ**

**ÇAĞRI MERKEZLERİNDE VARDİYA VE
PERSONEL ÇİZELGELEMESİ İÇİN KISIT
PROGRAMLAMA TEMELLİ BİR MODEL ÖNERİSİ**

DOKTORA TEZİ

Turgay TÜRKER

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Tez Danışmanı : Prof. Dr. Ayhan DEMİRİZ

Ocak 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

**ÇAĞRI MERKEZLERİNDE VARDİYA VE
PERSONEL ÇİZELGELEMESİ İÇİN KISIT
PROGRAMLAMA TEMELLİ BİR MODEL ÖNERİSİ**

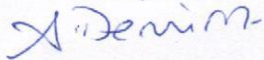
DOKTORA TEZİ

Turgay TÜRKER

Enstitü Anabilim Dalı : ENDÜSTRİ MÜHENDİSLİĞİ

Bu tez 03/01/2019 tarihinde aşağıdaki jüri tarafından oybirliği/oyçokluğu ile kabul edilmiştir.

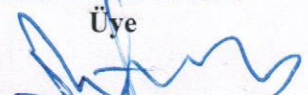
Prof. Dr.
Ayhan DEMİRİZ
Jüri Başkanı



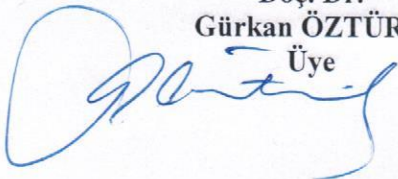
Prof. Dr.
Bayram TOPAL
Üye



Prof. Dr.
Emin GÜNDOĞAR
Üye



Doç. Dr.
Gürkan ÖZTÜRK
Üye

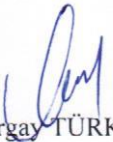


Doç. Dr.
Semra BORAN
Üye



BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.


Turgay TÜRKER
03.01.2019

TEŞEKKÜR

Doktora eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Prof. Dr. Ayhan DEMİRİZ'e teşekkürü bir borç biliyor ve şükranlarımı sunuyorum.

Çalışmam boyunca desteklerini esirgemeyen, yapıcı yaklaşımlarını/ yönlendirmelerini her daim yanımda hissettiğim tez izleme komitemde yer alan hocalarım Prof. Dr. Emin GÜNDOĞAR ve Prof. Dr. Bayram TOPAL'a teşekkürlerimi sunarım.

Tez çalışmamın düzeltme kısmında değerli fikirleriyle destek olan Doç. Dr. Gürkan ÖZTÜRK ve Doç. Dr. Semra BORAN hocalarıma çok teşekkür ederim.

Tez çalışmasının yürütülmesinde gerekli olan verileri tarafıma sağlayan çağrı merkezi çalışanlarına ve çalışmalarım sırasında beni maddi açıdan destekleyen TÜBİTAK'a ayrıca teşekkür ederim.

Tezimi, çalışma süresince tüm zorlukları benimle göğüsleyen ve hayatımın her evresinde bana destek olan değerli eşim Yasemin ALTUN TÜRKER'e ve dünyalar tatlısı biricik kızım Selma Ahsen'ime ithaf ediyorum.

İÇİNDEKİLER

TEŞEKKÜR.....	i
İÇİNDEKİLER....	ii
SİMGELER VE KISALTMALAR LİSTESİ.....	iv
ŞEKİLLER LİSTESİ	v
TABLolar LİSTESİ	vi
ÖZET.....	vii
SUMMARY.....	viii
BÖLÜM 1.	
GİRİŞ.....	1
BÖLÜM 2.	
LİTERATÜR ÖZETİ	8
2.1. İşgücü Çizelgeleme Alt Problemleri Açısından Literatür	11
2.1.1. İş yükü tahmini	11
2.1.2. Personel seviyesi belirleme	12
2.1.3. Vardiya çizelgeleme	14
2.1.4. Personel çizelgeleme	16
2.2. Kısıt Programlama Literatürü.....	19
BÖLÜM 3.	
METODOLOJİ... ..	23
3.1. Vardiya Çizelgeleme Problemi İçin Geliştirilen Model (Model 1)....	23
3.2. Personel Çizelgeleme Problemi İçin Geliştirilen Modeller.....	26
3.2.1. Tam sayılı programlama modeli (Model 2a).....	26

3.2.2. Kısıt programlama, IBM ILOG CP ve geliştirilen KP modeli (Model 2b).....	29
3.2.2.1. Kısıt programlama.....	29
3.2.2.2. IBM ILOG CPLEX Optimization Studio ve KP	35
3.2.2.3. Kısıt programlama model (Model 2b).....	46
3.3. Kıyaslama Test Örnekleri Modelleri	53
3.3.1. Kıyaslama test örneklerinde kullanılan matematiksel model (Model 3a).....	53
3.3.2. Kıyaslama test örnekleri için geliştirilen KP temelli model (Model 3b).....	57
 BÖLÜM 4.	
UYGULAMA ÇALIŞMASI	64
4.1. Gerçek Yaşam Problemi	64
4.1.1. Problem tanımı	64
4.1.2. Uygulama ve hesaplamalı sonuçlar	69
4.2. Deneysel Çalışmalar.....	77
4.2.1. Simüle test örnekleri ve elde edilen sonuçlar	77
4.2.2. Kıyaslama test örnekleri ve elde edilen sonuçlar	83
 BÖLÜM 5.	
TARTIŞMA VE SONUÇ	87
KAYNAKLAR.....	90
EKLER.....	98
ÖZGEÇMİŞ.....	148

SİMGELER VE KISALTMALAR LİSTESİ

AHS	: Analitik hiyerarşi süreci
ARIMA	: Autoregressive integrated moving average
BT	: Bayes tahmini
DP	: Doğrusal programlama
HW	: Holt-winters
İÇP	: İşgücü çizelgeleme problemi
KOP	: Kısıt optimizasyon problemleri
KP	: Kısıt programlama
KSP	: Kısıt sağlama problemleri
KTSP	: Karma tam sayılı programlama
MS	: Microsoft
OPD	: Optimizasyon programlama dili
RM	: Regresyon modelleri
SA	: Sinir ağları
SP	: Stokastik programlama
TL	: Türk lirası
TSP	: Tam sayılı programlama
VZA	: Veri zarflama analizi

ŞEKİLLER LİSTESİ

Şekil 1.1. İş gücü çizelgeleme alt problemleri	2
Şekil 1.2. Çalışmada ele alınan alt problemler.....	4
Şekil 1.3. Çalışmada geliştirilen modeller	5
Şekil 1.4. Kıyaslama test örnekleri ve kıyaslanan yaklaşımlar	6
Şekil 3.1. Bir harita boyama probleminin KSP olarak gösterimi.....	31
Şekil 3.2. KSP'nin genel bir algoritması.....	32
Şekil 3.3. ILOG CPLEX Optimization Studio programı genel ekran görüntüsü..	36
Şekil 3.4. Bir aralık (interval) değişken.	40
Şekil 3.5. Örnek bir sıralama karar değişkeni çözümü gösterimi	41
Şekil 3.6. pulse (a,h) ve pulse (u,v,h) fonksiyonları örneği.	44
Şekil 3.7. Kümülatif fonksiyon örneği	44
Şekil 3.8. noOverlap'sız ve noOverlap ile sıralama karar değişkeni çözüm örneği	45
Şekil 3.9. Görevlerin veri yapısı.	48
Şekil 4.1. Ele alınan vardiya düzenleri.....	66
Şekil 4.2. Mola gösterimli vardiya düzeninden bir kesit	66
Şekil 4.3. Planlama döneminde periyot başına ihtiyaç duyulan operatör miktarı.	68
Şekil 4.4. Mevcut durumun Model 1 ile çözümünü gösteren ekran görüntüsü	70
Şekil 4.5. Önerilen durumun Model 1 ile çözümünü gösteren ekran görüntüsü...	70
Şekil 4.6. Model 2a'ya ait özet değerler (1. Gün).....	76
Şekil 4.7. Model 2b'ya ait özet değerler (1. Gün).....	76
Şekil 4.8. Periyot başına molada olan operatör sayıları (1. Gün)	77
Şekil 4.9. MS Excel'de örnek bir mevcut durum değerleri.....	78
Şekil 4.10. MS Excel'de mevcut değerlerden Örnek1'in oluşturulması.....	78

TABLolar LİSTESİ

Tablo 2.1. İÇP'nin bazı özellikleri açısından literatür özeti	9
Tablo 2.2. İÇP'nin modellenmesi açısından literatür özeti.....	10
Tablo 3.1. Modelde kullanılan OPD sözdizimleri ve tanımları.	43
Tablo 3.2. Kıyaslama için geliştirilen KP modelinin özet kısıt yapısı.....	63
Tablo 4.1. Firmaya ait vardiya şablonları	65
Tablo 4.2. Model 1 için diğer girdi verileri.....	68
Tablo 4.3. Model 1'in çözüm sonuçlarına ait detaylar.....	69
Tablo 4.4. Günlere göre vardiyalara atanan operatör sayıları (G_{ds}).....	70
Tablo 4.5. KP çözücüsünde kullanılan parametre ayarları	71
Tablo 4.6. Model 2a ve Model 2b'ye ait çözüm detayları	73
Tablo 4.7. IBM ILOG CPLEX'ten alınabilecek bazı çözüm çıktıları	74
Tablo 4.8. %22'lik artış oranına göre güncellenmiş G_{ds} değerleri.....	75
Tablo 4.9. Test örneklerini oluşturmada kullanılan parametreler	79
Tablo 4.10. Test örnekleri ve Model 1'e ait detaylar	79
Tablo 4.11. Oluşturulan test örnekleri için çözüm sonuçları	81
Tablo 4.12. Kıyaslama test örneklerine ait bazı değerler	84
Tablo 4.13. KP temelli model ile diğer modellerin karşılaştırma sonuçları	86

ÖZET

Anahtar kelimeler: Çağrı merkezleri, vardiya çizelgeleme, personel çizelgeleme, tam sayılı programlama, kısıt programlama

Kısa zaman dilimlerinde oldukça belirsiz ve yoğun talep altında faaliyet gösteren çağrı merkezlerinde en uygun vardiya programına ulaşmak çok zor olabilmektedir. Zamansal olarak örtüşen vardiya sistemleri genellikle bu tür durumlar için tasarlanmıştır. Bu çalışma genel olarak zorlu vardiya sistemlerine sahip çağrı merkezlerindeki vardiya çizelgeleme ve personel çizelgeleme problemlerini entegre olarak ele almaktadır. Vardiya çizelgeleme problemi için bir tam sayılı programlama modeli önerilmiştir. Bu model ile planlama döneminde hangi vardiyaların açılacağı ve açılan vardiyalara kaç operatör atanacağına karar verilmektedir. Personel çizelgeleme problemi için ise bir tam sayılı programlama ve bir kısıt programlama modeli geliştirilmiştir. Bu modeller ile operatörlerin açılan vardiyalara atanması, haftalık izin günlerinin belirlenmesi ve gün içerisindeki operatör-mola (yemek ve dinlenme) eşleşmeleri sağlanmaktadır. Önerilen modeller birden çok firmaya dış kaynak sağlayan bir çağrı merkezi firmasındaki gerçek veriler ile çalıştırılmış, kabul edilebilir bir hesaplama zamanında optimal çözümler elde edilmiştir. Vardiya çizelgeleme problemi için önerilen model ile mevcut duruma göre %15'e yakın iyileştirme sağlanmıştır. Personel çizelgeleme problemi için ise önerilen modellerin hesaplama performans seviyeleri bir firmadan elde edilen veriler ve bazı benzetilmiş test örnekleri kullanılarak karşılaştırılmıştır. Hesaplama sonuçları kısıt programlama modelinin tam sayılı programlama modelinden daha iyi performans gösterdiğini ortaya koymaktadır. Ayrıca, kısıt programlama yaklaşımının literatürdeki mevcut modellerle karşılaştırması için kıyaslama veri setleri (benchmark data sets) kullanılmıştır. Yaklaşımın kıyaslama örneklerinin test edildiği diğer yöntemlerle yeterince rekabetçi olduğu ortaya konulmuştur.

Bu çalışmanın özgünlüğü iki başlıkta toplanabilir: (1) vardiya çizelgeleme problemi için bir model ve personel çizelgeleme problemi için iki farklı model ayrıntılı olarak sunulmakta, gerçek veriler ve standart veri setleri kullanılarak karşılaştırılmaktadır, (2) personel çizelgeleme problemi standart çizelgeleme mantığıyla görev-kaynak eşleşmesi şeklinde düşünülerek kısıt programlama ile etkin olarak modellenmiş ve bu model ile oldukça kısa süreli hesaplama zamanında uygun çözümler üretilebilmiştir.

A MODEL PROPOSAL FOR SHIFT SCHEDULING AND ROSTERING IN CALL CENTERS BASED ON CONSTRAINT PROGRAMMING

SUMMARY

Keywords: Call centers, shift scheduling, staff scheduling, rostering, integer programming, constraint programming

It may be very difficult to achieve the optimal shift schedule in call centers which have highly uncertain and peaked demand during short time periods. Overlapping shift systems are usually designed for such cases. This paper studies shift scheduling and rostering problems for inbound call centers where overlapping shift systems are used. An integer programming model that determines which shifts to be opened and how many operators to be assigned to these shifts is proposed for the shift scheduling problem. For the rostering problem both integer programming and constraint programming models are developed to determine assignments of operators to all shifts, weekly days-off and meal and relief break times of the operators. The proposed models are tested on real data supplied by an outsource call center and optimal results are found in an acceptable computation time. An improvement of 15% in the objective function compared to the current situation is observed with the proposed model for the shift scheduling problem. The computational performances of the proposed integer and constraint programming models for the rostering problem are compared using real data observed at a call center and simulated test instances. In addition, benchmark instances are used to compare our constraint programming approach with the existing models. The results of the comprehensive computational study indicate that the constraint programming model runs more efficiently than the integer programming model for the rostering problem. In addition, our constraint programming approach has been shown to be sufficiently competitive with other methods in which benchmarking instances are tested.

The originality of this research can be attributed to two contributions: (a) a model for shift scheduling problem and two models for rostering problem are presented in detail, compared using real and benchmarking data, (b) the rostering problem is considered as a task-resource allocation and considerably shorter computation times are obtained by modeling this new problem via constraint programming.

BÖLÜM 1. GİRİŞ

Günümüzde işletmeler, rekabet edebilirliklerini arttırmaya, globalleşmeyle birlikte dünyayı saran ekonomik krizlerin etkilerini bir ölçüde hafifletebilmeye ve müşteri memnuniyetini en üst düzeyde tutarken işgücü maliyetlerini de en aza indirebilmeye odaklanmaktadır. Bununla beraber çalışan personelin memnuniyeti de işletmeler tarafından önemle takip edilmektedir. Ayrıca her ülkede, çalışan personelin çalışma koşulları ile ilgili olarak yasaların ve sendikaların belirlediği birtakım kurallar ve düzenlemeler mevcuttur.

Diğer sektörlerle nazaran özellikle hizmet sektöründe insan faktörünün daha çok kullanılması nedeniyle işgücü maliyetleri maliyet kalemleri içerisinde önemli bir unsur olarak yer almaktadır. Bu maliyetin azaltılmasına yönelik yapılan çalışmalar toplam maliyetlerde önemli derecede azalmalara neden olmaktadır. Ayrıca başarılı bir şekilde ortaya konulan işgücü çizelgeleri hem firma açısından hem müşteri açısından ve hatta personel açısından karşılıklı memnuniyeti sağlayacaktır. Bu bağlamda, işletmeler açısından en önemli ve pahalı kaynak olan işgücünün etkin kullanımının oldukça önemli olduğu ortadadır. Van den Bergh ve ark. [1] çalışmalarında geçen yüzyıldan bu yana işgücü çizelgeleme problemi (İÇP) ile ilgili çok sayıda çalışmada işgücü maliyetlerinin azaltılarak firmalara fayda sağlandığına işaret etmektedirler.

Hizmet sektörünün emek yoğun bir kolu olan çağrı merkezlerinde onlarca personel (operatör) çalışmaktadır. Bu merkezlerde günde yüzlerce çağrıya yanıt verilmekte ya da dışa dönük çağrılar yapılmaktadır. Çağrı merkezlerinde müşteri ve personel memnuniyetinin artırılması noktasında işgücünün etkin kullanımı oldukça önem arz etmektedir.

Çağrı merkezlerinde toplam işgücü maliyeti toplam maliyetin yaklaşık olarak %70'ine denk gelmektedir [2]. Açıkçası bu yüksek oran operasyonların ekonomik verimliliğinin, temel olarak işgücü planlama sürecinin kalitesi ile belirleneceğini göz önüne sermektedir [3]. Çağrı merkezlerinin her gün yüzlerce çağrı ile baş başa kaldığı göz önüne alındığında da verimlilik artışının kazançlara etkisi yadsınamaz derecede olacaktır.

Çağrı merkezleri işgücü planlaması araştırmalarının birçoğunda hem istenen servis düzeylerini hem de bazı kanun ve yönetmeliklerde belirtilen hususları sağlayan işgücü çizelgeleri oluşturulması hedeflenmektedir. Bu çizelgeler oluşturulurken çalışanların tercihlerini dikkate almak, maliyetleri en aza indirmek, tüm işyeri kısıtlamalarını karşılamak gibi son derece kısıtlı ve karmaşık problemlere karşı optimum, hatta iyi çözümler üretmek de son derece zordur [4].

Bir çağrı merkezi için İÇP dört alt problem olarak ele alınabilir [5]:

1. İş yükü tahmini (Workload prediction),
2. Personel seviyesi belirleme (Staffing),
3. Vardiya çizelgeleme (Shift scheduling),
4. Personel çizelgeleme (Rostering).

Bu alt problemler Şekil 1.1.'de gösterilmiştir. İlk alt problem (iş yükünün tahmin edilmesi) belirli süreler için (genellikle 30 dk. ya da 15 dk. [6]) çağrı merkezine gelen çağrıların tahmin edilmesine dayanmaktadır. Planlama dönemi boyunca çağrı merkezine hangi zaman aralıklarında ne kadar çağrı geleceği bu alt problemin ana konusudur.



Şekil 1.1. İş gücü çizelgeleme alt problemleri.

İkinci alt problem olan personel seviyesi belirleme ile temel olarak gelen çağrı hızına bağlı olarak gerekli minimum operatör sayısı hesaplanmaktadır. Başka bir ifadeyle tahmin edilen çağrı adetlerinin istenen servis düzeyini karşılayacak şekilde periyodik düzeyde operatör ihtiyacına dönüştürülmesidir.

Üçüncü alt problem olan vardiya çizelgeleme ise yukarıda sözü edilen operatör ihtiyaçlarını en iyi şekilde karşılayacak olan vardiyaların seçilmesidir/açılmasıdır. Bu alt problemin çözümü ile hangi operatörün hangi vardiyaya atandığından ziyade hangi vardiyaların açılacağı belirlenmektedir. Dördüncü ve son alt problem olan personel çizelgeleme ise genel olarak mevcut operatörlerin açılan vardiyalara atanmasını içermektedir. Ayrıca her bir operatörün gün içi mola ve/veya genellikle haftalık izin zamanlarının belirlenmesi ile ilgilidir.

Bu tezdeki amacımız yukarıda sözü edilen son iki alt problemin ele alınması suretiyle bir çağrı merkezi için vardiya ve personel çizelgelemesi yapacak etkin bir modelleme yaklaşımı ortaya koymaktır. Daha açık bir ifadeyle bu tez yukarıda sözü edilen 3. ve 4. alt problemleri kapsamaktadır. Yani bu tezde çağrı varış oranları ile ilgili yakın geçmişteki verilerin, öngörülebilir bir geleceği temsil ettiği varsayılmıştır. Bu yüzden çalışmada ilk iki alt problem ele alınmamıştır. Ancak gerekmesi halinde, diğer alt problemlerin çalışmaya kolaylıkla entegre edilebileceğini de vurgulamak gerekir.

Vardiya çizelgelemede planlama dönemi boyunca uygulanması düşünülen vardiya düzeni ortaya konulmaktadır. Firmaların vardiya düzeni genel olarak zamansal anlamda birbiri ile örtüşmeyen (çakışmayan) vardiyalardan oluşmaktadır ki böylesi bir durumda her vardiyadaki personel ihtiyacı birbirinden bağımsız ve belirlidir. Bu tarz modellerde tahsis problemini (allocation problem) çözmek kolaydır.

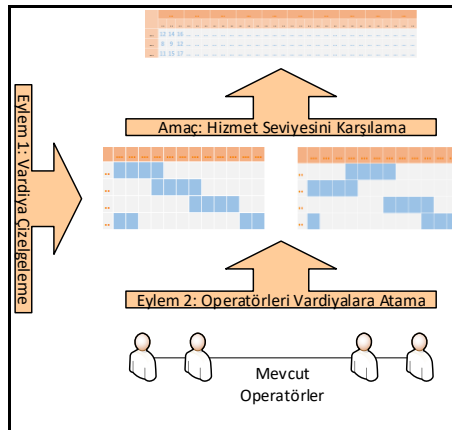
Çağrı merkezlerindeki gibi talep ihtiyacının kısa periyot seviyelerinde belirleniyor olması halinde yukarıda sözü edilen konfigürasyon artık kullanışlı olmamakta ve sonuç olarak birbiri ile zamansal olarak örtüşen/çakışan vardiyaların (overlapping shifts) kullanımı söz konusu olmaktadır. Bu durumda periyodik işgücü ihtiyacı farklı

vardiyalardaki çalışanlar tarafından karşılanmaktadır. Çağrı merkezi vardiya düzeni bu anlamda zorludur.

Personel çizelgeleme probleminde de öncelikle belirli kısıtlamalar dahilinde operatör-vardiya eşleşmeleri yapılmaktadır. Ele alınan problemde eğer gereklilikler (kısıtlamalar) kapsamında var ise her bir operatörün izin ve mola (yemek, dinlenme vb.) zamanları da belirlenmektedir.

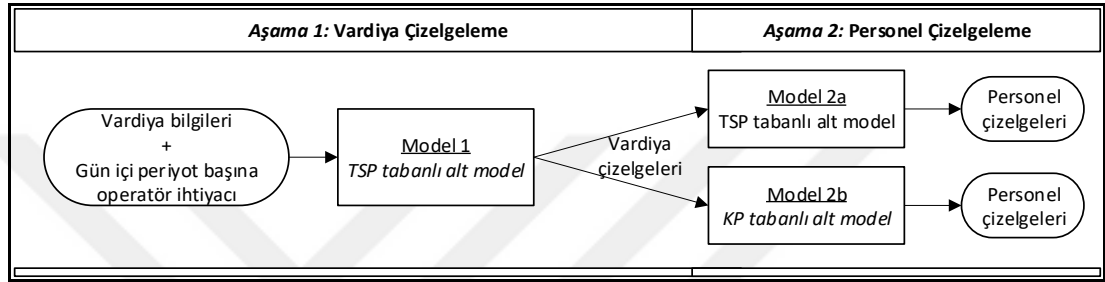
Burada özellikle mola zamanlarının belirlenmesi konusuna dikkat çekilecek olursa: Bilindiği üzere bir operatör bir günde yaklaşık 7 ile 9 saat arası çalışmakta ve bu süre zarfında molalara ihtiyaç duymaktadır. Sungur ve ark. [7] çalışmalarında mola zamanlarının anlık (real time) olarak belirlenmesinin servis seviyesinin düşmesi veya çalışan memnuniyetinin azalması gibi riskleri olduğunu, bu anlamda operatör-mola atamalarının çizelgeleme esnasında ele alınması gerektiğini vurgulamışlardır. Ayrıca yapılan çalışmaların çoğunda sadece yemek molalarının ele alındığını, buna ek olarak dinlenme molalarının (relief breaks) da göz önüne alınması gerektiğini belirtmişlerdir.

Çalışmanın genel çerçevesi Şekil 1.2.'de resmedilmiştir. Çalışmada öncelikle işgücü ihtiyaçlarını karşılayacak vardiyalar ve bu vardiyalara atanacak işgücü sayısı belirlenecek (Eylem 1), daha sonra operatörlerin bu vardiyalara bazı kısıtlar göz önüne alınarak atanması, izin ve mola zamanlarının belirlenmesi sağlanacaktır (Eylem 2).



Şekil 1.2. Çalışmada ele alınan alt problemler.

Diğer bir ifadeyle Şekil 1.3.'te de gösterildiği gibi öncelikle vardiya çizelgeleme problemini çözmeye yönelik olarak operatör ihtiyacını ve vardiya bilgilerini girdi olarak alan bir Tam Sayılı Programlama (TSP) modeli ortaya konulacak (Model 1) ve çözülecektir. Burada sözü edilen operatör ihtiyacı, planlama döneminde her bir zaman aralığında servis düzeyini garanti edecek en az operatör sayısını yansıttığını vurgulamak gerekir. Daha sonra ilk problemin çözümü neticesinde ortaya konulan vardiyalar için personel çizelgeleme problemi çözülecektir.



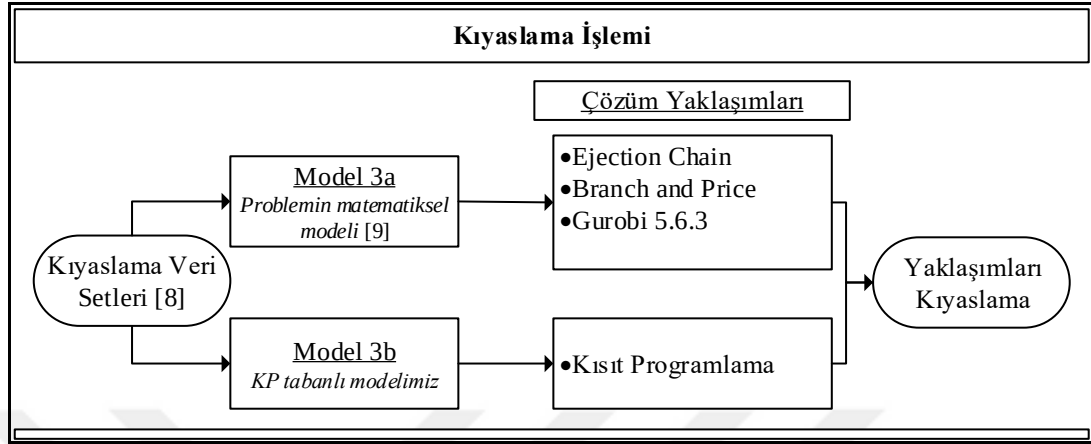
Şekil 1.3. Çalışmada geliştirilen modeller

Çalışmada Şekil 1.3.'te de görüldüğü üzere personel çizelgeleme problemi için birbirini ile kıyaslama adına iki ayrı model önerilmiştir. Bu modellerden ilki klasik TSP modelleme temelli yaklaşım (Model 2a), diğeri ise Kısıt Programlama (KP) temelli yaklaşımdır (Model 2b). KP temelli yaklaşım ile bu problem klasik çizelgeleme problemlerindeki gibi bir görev-kaynak paylaşımı olarak ele alınmış ve bu doğrultuda modellenmiştir. Aslında bu çalışmanın ana katkısı da yeni bir bakış açısına dayanarak, çağrı merkezlerindeki personel çizelgeleme probleminin çözümüne yönelik etkin bir KP modeli ortaya koymasındır.

Geliştirilen modellerin standart veri setleri üzerinde doğrulanması çalışmaların güvenilirliği açısından büyük önem arz etmektedir. Daha açık ifadeyle, ortaya konulan modellerin standart veri setleri üzerindeki performansının belirlenerek aynı konuda geliştirilmiş olan diğer modellerle kıyaslanması gerekmektedir. Bu tezdeki kıyaslama süreci Şekil 1.4.'te gösterilmiştir.

Kıyaslama işlemi için www.schedulingbenchmarks.org adlı internet sitesinde detayları var olan kıyaslama veri setleri kullanılmıştır [8]. Kıyaslama işlemi, geliştirilen KP temelli yaklaşım (Model 3b) ile Curtois ve Qu'nun çalışmalarında [9]

yer verdiği Şekil 1.4.'te de görülen 3 farklı yöntem arasında yapılmış ve karşılaştırmalı test sonuçları ortaya konulmuştur.



Şekil 1.4. Kıyaslama test örnekleri ve kıyaslanan yaklaşımlar.

Model 3b, Model 3a ile belirtilen matematiksel modelin KP ile modellenen halidir. Daha açık ifadeyle Curtois ve Qu'nun çalışmasında yer verilen matematiksel model (Model 3a) KP yaklaşımı ile modellenerek test örnekleri üzerine uygulanmıştır. Çıkan sonuçlar yine aynı çalışmada [9] yer alan 3 farklı yöntem sonuçları ile kıyaslanarak yorumlanmıştır.

Tezde özetle KP'nin bu tarz problemler üzerinde etkin çözümler üretebildiği gerek gerçek veri kullanımı ile gerekse standart kıyaslama test örnekleri kullanımı ile gösterilmiştir. Araştırma sonuçları literatürdeki mevcut çözümler ile karşılaştırılarak geliştirilen modellerin eksik ve güçlü yönleri ortaya konulmuştur.

Tez beş bölümden oluşmaktadır. Birinci bölümde; tez çalışmasının konusu, önemi, amaçları ve içeriğine ilişkin genel bilgiler verilmiştir. İkinci bölümde, işgücü çizelgeleme problemleri ile ilgili kapsamlı bir literatür özeti yer almaktadır.

Üçüncü bölümde, vardiya çizelgeleme problemi için önerilen TSP tabanlı model, personel çizelgeleme problemi için önerilen TSP ve KP tabanlı modeller ve kıyaslama test modelleri hakkında detaylı bilgiler verilmiştir. Ayrıca KP yaklaşımı ve modellerin geliştirildiği IBM ILOG CPLEX Optimization Studio adlı paket program hakkında genel bilgiler verilmiştir.

Çalışmanın dördüncü bölümü olan uygulama bölümünde ise hâlihazırda faaliyet gösteren bir çağrı merkezinden elde edilen gerçek verilere, simültane oluşturulan test verilerine ve personel çizelgeleme problemi için oluşturulmuş standart kıyaslama test verilerine dayanan deneysel çalışmalara yer verilmiştir. Elde edilen sonuçlar detaylıca incelenmiştir.

Çalışmanın nihai sonuçlarının açıklandığı beşinci ve son bölümde elde edilen bulgular ortaya konulmuştur. Ayrıca gelecek çalışmalarda diğer araştırmacılara ışık tutacak önerilerde bulunulmuştur.



BÖLÜM 2. LİTERATÜR ÖZETİ

Literatürde işgücü çizelgeleme için yapılan modellerin uzun bir geçmişi vardır [10]. İlk çalışmalardan olan Edie'nin çalışmasında [11] deneysel bir çalışma yapılarak gişelerdeki trafik gecikmeleri incelenmiş ve gişelerde ihtiyaç duyulan personelin belirlmesi amacıyla sabit kuyruk sistemleri formüle edilmiştir.

Dantzig [12] ise, Edie [11]'nin yaptığı çalışmaya dayanarak bu probleme uygun tam sayılı doğrusal programlama modelinin nasıl oluşturacağını göstermiştir. Fakat İÇP'ler ilk ortaya atıldığı zamandan bu zamana farklı açılardan birçok değişiklik göstermiştir.

Van den Bergh ve ark. [1], Ernst ve ark. [4, 13] ile Brucker ve ark. [10]'nın yapmış oldukları çalışmalar literatürdeki İÇP ile ilgili çalışmaları detaylıca ortaya koymaları açısından önemlidir. Bu çalışmalarda İÇP'lerin uygulama alanları, çözüm yöntemleri ve modelleri hakkında kapsamlı bir literatür araştırmasına yer vermişlerdir.

Bir önceki bölümde belirtildiği üzere çağrı merkezleri açısından İÇP 4 alt problem halinde incelenmektedir. Bu bölümde sırasıyla başlıklar halinde bu alt problemler hakkında literatür incelemesine yer verilecek ve daha sonra İÇP açısından KP literatürüne değinilecektir. İncelemede çoğunlukla çağrı merkezi kapsamında yapılan çalışmalar göz önüne alınmıştır. Bu tezde periyodik personel gereksinimi ve vardiya bilgileri girdi olarak alınıp vardiya ve personel çizelgelemesi yapıldığı için ilk iki alt problem kapsamındaki literatür incelemesi daha kısa (özet) şekilde yapılmıştır.

Bu başlıklara geçmeden önce çağrı merkezi işgücü çizelgeleme çalışmalarının 2003-2018 yıllarını kapsayan literatür taramasının özeti Tablo 2.1. ve Tablo 2.2.'de verilmiştir. Çalışmalar yazar isimlerine göre alfabetik olarak listelenmiştir.

Tablo 2.1. İÇP'nin bazı özellikleri açısından literatür özeti

İncelenen Literatür	Yıl	Ele Alınan Alt Problem Tipi				Mola(lar)	
		İş Yüğü Tah.	Personel Sev. Bel.	Vardiya Çiz.	Personel Çiz.	Y	D
Alfares [14]	2007		✓		✓	✓	
Ásgeirsson ve Sigurðardóttir [15]	2014		✓		✓		
Atlason ve ark. [16]	2008		✓	✓			
Atlason ve ark. [17]	2004		✓				
Avradimis ve ark. [18]	2008		✓	✓			
Avramidis ve ark. [19]	2009		✓				
Bhulai ve ark. [5]	2008		✓	✓			
Canon [20]	2007		✓	✓	✓	✓	
Castillo ve ark. [21]	2009		✓	✓		✓	✓
Cezik ve L'Ecuyer [22]	2008		✓				
Defraeye ve Van Nieuwenhuyse [23]	2016			✓			
Dietz [3]	2011		✓		✓	✓	
Dudin ve ark. [24]	2013		✓				
Ertogral ve Bamuqabel [25]	2008		✓		✓		
Excoffier ve ark. [26]	2016			✓			
Gong ve ark. [27]	2015		✓				
Green ve ark. [28]	2007		✓				
Helber ve Henken [29]	2010		✓	✓			
Hojati [30]	2010				✓	✓	
Ibrahim ve L'Ecuyer [31]	2013	✓					
Ingolfsson ve ark. [32]	2010		✓		✓	✓	✓
Koole ve van der Sluis [33]	2003		✓	✓			
Kyngäs ve ark. [34]	2012				✓		
Liao ve ark. [35]	2009		✓				
Mattia ve ark. [36]	2018		✓	✓			
Millán-Ruiz ve Hidalgo [37]	2013	✓					
Nah ve Kim [38]	2013		✓		✓	✓	✓
Örmeci ve ark. [39]	2014				✓		
Pot ve ark. [40]	2008		✓				
Restrepo ve ark. [41]	2017				✓	✓	✓
Robbins ve Harrison [42]	2008		✓	✓			
Robbins ve Harrison [43]	2010		✓				
Taskiran ve Zhang [44]	2017				✓	✓	
Wallace ve Whitt [45]	2005		✓				
Weinberg ve ark. [46]	2007	✓					
Bu çalışma				✓	✓	✓	✓

✓=Var, Y=Yemek, D=Dinlenme

Tablo 2.2. İÇP'nin modellenmesi açısından literatür özeti

İncelenen Literatür	DP	TSP	KTSP	Kuy.	Yap.	Diğ.	Sim.	KP	Diğ.
					Sez.	Sez.			
Alfares [14]		✓		✓					
Ásgeirsson ve Sigurðardóttir [15]			✓						
Atlason ve ark. [16]			✓			✓	✓		
Atlason ve ark. [17]			✓				✓		
Avradimis ve ark. [18]			✓				✓		
Avramidis ve ark. [19]			✓			✓	✓		
Bhulai ve ark. [5]			✓			✓	✓		
Canon [20]			✓			✓		✓	
Castillo ve ark. [21]				✓	✓		✓		VZA
Cezik ve L'Ecuyer [22]	✓					✓	✓		
Defraeye ve Van Nieuwenhuyse [23]					✓				
Dietz [3]				✓					
Dudin ve ark. [24]				✓					
Ertogral ve Bamuqabel [25]		✓		✓			✓		AHS
Excoffier ve ark. [26]			✓						SP
Gong ve ark. [27]				✓					
Green ve ark. [28]				✓		✓	✓		
Helber ve Henken [29]			✓				✓		
Hojati [30]					✓				
Ibrahim ve L'Ecuyer [31]									RM
Ingolfsson ve ark. [32]			✓	✓		✓			
Koole ve van der Sluis [33]				✓	✓				
Kyngäs ve ark. [34]						✓			
Liao ve ark. [35]		✓					✓		SP
Mattia ve ark. [36]		✓			✓				
Millán-Ruiz ve Hidalgo [37]									SA
Nah ve Kim [38]			✓						
Örmeci ve ark. [39]			✓						
Pot ve ark. [40]			✓			✓			
Restrepo ve ark. [41]									SP
Robbins ve Harrison [42]					✓	✓	✓		
Robbins ve Harrison [43]			✓			✓	✓		SP
Taskiran ve Zhang [44]		✓							
Wallace ve Whitt [45]				✓			✓		
Weinberg ve ark. [46]									BT
Bu çalışma		✓						✓	

✓=Var, DP=Doğrusal Programlama, KTSP=Karma Tam Sayılı Programlama, Kuy.= Kuyruk Modelleri, Yap. Sez.=Yapısal Sezgiseller, Diğ. Sez.=Diğer Sezgiseller, Sim.=Simülasyon, Diğ.=Diğerleri, VZA= Veri Zarflama Analizi, AHS= Analitik Hiyerarşi Süreci, SP= Stokastik Programlama, SA=Sinir Ağları, RM=Regresyon Modelleri, BT=Bayes Tahmini.

Tablo 2.1. ve Tablo 2.2., çağrı merkezleri için İÇP'nin bazı özelliklerini ve modelleme yaklaşımlarını göstermektedir. Ayrıca alt problemlerin iki veya üçünü kapsayan çalışmaların varlığını da ortaya koymaktadır. Her iki tablonun son satırına literatürdeki çalışmalar ile benzerliklerini/farklılıklarını daha iyi göstermek adına bu çalışmaya ait özellikler konulmuştur.

Literatürdeki çalışmalar incelendiğinde aşağıda gösterilen noktalarda literatür eksiklikleri olduğu gözlenmiştir:

1. Vardiya ve personel çizelgeleme problemlerini birlikte ele alan çalışmaların azlığı,
2. KP'nin yok derecede az kullanımı,
3. Zor bir kısıt olan “Molaları Belirleme” kısıtının az sayıda kullanımı (özellikle dinlenme molaları).

Bu çalışmanın özgünlüğü, yukarıda bahsi geçen literatür eksikliklerini gidermeye (açıkları kapatmaya) yönelik olmasıdır.

Burada bir nokta daha vurgulanmalıdır. Literatürde bu alanda yapılan önemli çalışmalar Tablo 2.1. ve Tablo 2.2.'de listelenmiştir ve belirtilmelidir ki yapılan bu çalışmaların hemen hemen tümünde geliştirilen modeller doğrulanmak üzere standart kıyaslama örnekleri ile test edilmemiştir. Yazarların model yaklaşımlarının standart veri setleri üzerindeki davranışı ortaya konulmamıştır.

2.1. İşgücü Çizelgeleme Alt Problemleri Açısından Literatür

2.1.1. İş yükü tahmini

Önceki bölümde de belirtildiği üzere bir çağrı merkezi işgücü çizelgelemesi probleminde ilk alt problem belirli süreler için (genellikle 15-, 30-, veya 60- dakikalık periyotlarla) gelen çağrılar tahmin edilmesidir [5].

Çağrı merkezlerinde yapılan iki tip çağrı vardır. Bunlardan biri müşteri temsilcisinin hizmet satmak veya bilgilendirmek için müşteriyi aradığı dış aramalardır. Diğerisi ise müşterinin hizmet veya bilgi almak ya da sorun iletmek için müşteri temsilcisini aradığı çağrı merkezleri açısından gelen çağrılardır.

Dış aramaların planlanması zor olmamaktadır. Gelen çağrılar açısından ise müşterilerden gelen çağrıların sayısı ve sıklığı konusunda belirsizliğin olması ve belirli bir servis kalitesi düzeyinin sağlanması gerekliliği literatürdeki çalışmaların bu yönde gelişmesini zorunlu kılmıştır [20].

Literatürdeki ilk çalışmaların çoğu Hareketli Ortalamalar (Moving Averages) ve Düzeltme Yöntemleri (Smoothing Methods) üzerine odaklanmıştır. Bianchi ve ark. [47], yaptıkları çalışmada çağrı tahmininde Holt-Winters (HW) adı verilen Üstel Ağırlıklı Hareketli Ortalama Düzleştirme Modeli ile Box-Jenkins (ARIMA) adı verilen Otoregresif Bütünleşik Hareketli Ortalama Modeli'ni karşılaştırmışlardır.

Daha basit sistemler için HW'nin doğruluk payının yüksek olduğu sonucuna ulaşmalarına karşın, daha kompleks sistemler açısından ARIMA'nın daha gerçekçi sonuçlar verdiğini gözlemişlerdir. Taylor [48] çalışmasında, HW düzleştirme metodunun bir versiyonunu da içine alan farklı zaman serisi modellerinin tahmin doğruluğunu karşılaştırmış ve basit tahmin tekniklerinin zamansal olarak büyük olan süreçlerin tahmininde çokta başarılı olmadığı sonucuna varmıştır.

Yine literatürde çağrı tahmini ile ilgili çalışmalarda regresyon modellerinin [31], kuyruk teorisinin [49], simülasyon temelli algoritmaların [50], stokastik modellerin [51, 52], Bayes yaklaşımının [46, 53] ve sinirsel ağların [37] kullanımı göze çarpmaktadır.

2.1.2. Personel seviyesi belirleme

Çağrı merkezlerinde çağrı varış hızları zamansal bağlamda büyük ölçüde değişkendir. Müşteri tatmininin belirli bir limitin altına düşmemesi de ancak ve

ancak personel çizelgelerinin etkin bir şekilde planlanması ile mümkün olacaktır. Tablo 2.1.'de görüldüğü üzere literatürde çağrı merkezlerinde gelen çağrı hızına bağlı olarak gerekli minimum personel sayısını bulmak için yapılmış olan birçok çalışma bulunmaktadır.

Yapılan çalışmalardaki genel görüş çağrı merkezi sisteminin bir kuyruk sistemi olarak kabul edilebileceğidir [49]. Bu nedenle çalışmalar çoğunlukla kuyruk modellerine dayandırılmakta ve bu modellerden özellikle üstel, hiperüstel, Erlang, aşamalı, sabit ve genel servis zamanı dağılımlı modellerin kullanıldığı göze çarpmaktadır.

Analitik çözümün ve kapalı formun elde edilebilmesi nedeniyle yaygın olarak kullanılan en basit çağrı merkezi kuyruk modeli $M/M/S$, diğer bir deyişle Erlang C modelidir [2, 27]. Literatürde çağrı merkezlerinde gerekli işgücü sayısının hesaplanması konusunda geliştirilen modeller genellikle Erlang C hesaplarının bazı kısıtlarına alternatif oluşturma yönündedir. Bu kapsamda yapılmış olan çalışmalara [2, 49, 54–56] iyi birer örnektir.

Brown ve ark. [54], bir çağrı merkezinin operasyonlarının istatistiksel analizini kuyruk bilimi perspektifi ile yapmışlardır. Çağrı merkezi operasyonları için teorik yapının istatistiksel analizinden nasıl karakteristikler çıkarılacağını ve pratik olarak faydalı matematiksel modellerin nasıl geliştirileceğini araştırmışlardır. Bu matematiksel modelin temel girdileri; operatör sayısı, çağrı hacmi, servis zamanı, bekleme zamanı gibi sistem temelleriyle ilgili istatistiklerdir. Çalışmalarındaki deneysel analizler Erlang modellerinin temel varsayımlarının karşılanıp karşılanmadığına bakılmaksızın, müşteri gecikmelerini tahmin etmek için Erlang modellerinin tercihini desteklemektedir.

Gong ve ark. [27], çalışmalarında kuyruk servis sistemlerinde çalışan personelin optimum dağıtılması ile ilgili çalışmışlardır. Kuyruk modeli $M/M/s+M$ olan bir çağrı merkezi modellemiştir. Kuyruk modellerini müşterilerin bekleme süresi ile memnuniyetlerini de dikkate alacak şekilde geliştirilmiştir. Geliştirdikleri modelde

müşterilerin tekrarlayan davranışları açıkça dikkate alınmıştır. Tatmin olan bir müşterinin geri dönebileceğini ve bu durumun varış oranını etkileyebileceğinden bahsetmişlerdir. Modeli, müşteri tekrar davranışını dikkate almayan geleneksel modellerle deneysel yöntemleri kullanarak karşılaştırmışlardır. Ayrıca, yöneticilerin personel dağılımı yaparken müşterilerin farklı davranışlarını da dikkate alan bir atamayı nasıl yapabileceklerini göstermişlerdir.

Araştırmacıların bir bölümü matematiksel modelleme yöntemleriyle de bu sistemi optimize eden yaklaşımlar geliştirirken, bir diğer alternatif de benzetim uygulamaları olarak öne çıkmaktadır. Benzetim modelleri de çağrı merkezlerinde her zaman aralığı için gerekli olan uygun personel sayısını elde etmek için son zamanlarda sıklıkla başvurulan bir yöntemdir [55]. Bu kapsamda Atlason ve ark. [17] ile Cezik ve L'Ecuyer [22]'in çalışmaları literatürde dikkate değerdir.

2.1.3. Vardiya çizelgeleme

Vardiya çizelgeleme problemi planlama dönemi boyunca her bir çalışma gününde açılması gereken vardiyaların başlama-bitiş zamanlarının belirlenmesini ve bu vardiyalara atanacak işgücü sayısının ortaya konulmasını içermektedir. Genel olarak ana amaç en az maliyetle periyodik düzeydeki istenen işgücü talebinin karşılanmasıdır [33].

Bu problem ile ilgili literatürde farklı alanlarda yapılmış olan oldukça fazla çalışma yer almaktadır. Öncelikle yapılan çalışmalarda problemin TSP ve KTPS ile modellendiği, ancak problemlerin değişken ve kısıt sayısı arttıkça (daha karmaşık hale geldikçe) simülasyon temelli ve sezgisel/meta-sezgisel tekniklerin uygulanmaya başladığı gözlenmiştir [1].

Thompson [57] çalışmasında vardiya çizelgeleme problemi için iki ayrı model önermiştir. Bu modellerden birinde periyot başına kabul edilebilir servis seviyesini baz alan bir yaklaşım sergilerken diğerinde planlama dönemi boyunca ortalama

servis düzeyini dikkate alan bir yaklaşım ortaya koymuştur. Geliştirilen modeller KTSP temellidir.

Atlason ve ark. [17] ise Thompson [57]'nin modelinin simülasyon temelli bir yaklaşım ile modellenebileceği ortaya koymuşlardır. Geliştirdikleri modeli sayısal bir örnek üzerinde test etmişler ve Thompson [57]'nin sonuçlarına kıyasla daha uygulanabilir model geliştirdiklerini belirtmişlerdir.

Koole ve Van der Sluis [33] çalışmalarında geleneksel yaklaşımda servis düzeyinin periyodik boyutta sağlanması gerektiğini belirtildiğini, bunun ise zor bir kısıt olduğunu ortaya koymuşlardır. Bu yaklaşımın yerine tüm planlama ufku (genellikle bir gün veya bir hafta) üzerinden genel bir hizmet düzeyi hedefini karşılayacak bir model geliştirmişlerdir. Servis düzeyinin periyodik boyutta sağlanması kısıtının zor bir kısıt olduğu doğrudur. Fakat bu kısıtın dahil edilmediği bir modellemenin çağrı merkezleri açısından gerçek sistemi tam olarak yansıtmadığı da bir gerçektir.

Bhulai ve ark. [5], çağrı merkezlerinde vardiya çizelgelemesi için TSP tabanlı bir model geliştirilmiştir. Yöntem iki adımdan oluşmaktadır. Öncelikle, birinci adımda personel seviyeleri belirlenmektedir. Sonraki adımda, ilk adımda elde edilen sonuçlar çalışmadaki ikinci adım olan vardiya çizelgeleme problemi için girdi olarak kullanılmaktadır. Model, uygulanabilirliğinin kolay olması ve oldukça kısa hesaplama zamanına sahip olması açısından dikkate değerdir.

Defraeye ve Nieuwenhuyse [23], yaptıkları çalışmada durağan olmayan stokastik talepe ve hizmet seviyesi kısıtlamalarına sahip sistemlerde optimal vardiya çizelgelemelerini tahmin etmek için dal-sınır algoritması yaklaşımını sunmuşlardır. Algoritma, küçük çağrı merkezleri, bankalar ve perakende satış mağazaları gibi çalışma saatleri sınırlı olan hizmet sistemleri için tasarlanmıştır. Yapılan deneyler, algoritmanın çözüm uzayında optimum çözümü kapsamayan bölgelerden kaçınmada etkili olduğunu göstermiştir. Dahası tahmin edilen optimum çözümü bulmak için sınırlı sayıda hesaplama yapmaktadır. Çalışmanın sonunda algoritma iki aşamalı

algoritmalarla kıyaslanmış ve kendi sundukları modelin daha rekabetçi olduğunu, daha kaliteli sonuçları daha hızlı zamanda bulduğunu göstermişlerdir.

Mattia ve ark. [36], vardiya sayısının birden fazla olduğu ve esnek çalışma saatlerinin olduğu bir çağrı merkezinde tam sayılı programlama ile vardiya çizelgeleme problemini ele almışlardır. Geliştirdikleri modelde taleplerin ardışık zaman dilimlerindeki korelasyonunu ele almışlar, bu şekildeki modellemenin diğer vardiya çizelgeleme yaklaşımlarından daha etkin olduğunu göstermişlerdir. Geliştirilen dal-kesme algoritmasının gerçek yaşam problemlerinin çözümünde de etkili olduğunu faaliyette olan bir çağrı merkezinden alınan gerçek veriler ile ortaya koymuşlardır.

Bu çalışmalara ek olarak Henderson ve Berry [58] ile Aykin [59, 60]'in çalışmaları bu kapsamdaki temel çalışmalar olarak değerlendirilmektedir.

Literatürdeki çalışmalarda genel olarak bu alt problem kapsamındaki modellemeler az sayıda vardiya içeren konfigürasyonlar olarak ele alınmış olup önerilen yöntemlerin performanslarının çoklu ve/veya örtüşen/çakışan (overlapping) vardiya düzeninde nasıl olacağı bilinmemektedir.

2.1.4. Personel çizelgeleme

İyi bir şekilde oluşturulan personel çizelgelerinin düşük maliyet, kaynakların etkin kullanımı, çalışan memnuniyetinin artışı, adil iş yükü ve vardiya dağıtımı gibi faydaları vardır [34]. Literatürde bu problemin modellenmesinde de genellikle TSP/KTSP modelleri tercih edilmiştir [39].

Aykin [59], çalışmasında personel çizelgeleme problemi için bir TSP modeli geliştirmiştir. Çalışmada her bir vardiya için ayrı ayrı mola değişkenleri (yemek ve dinlenme) tanımlanmıştır. Bu çalışmada konu edilen problemde, iki vardiya söz konusudur. Vardiya süreleri dokuzar saattir. Bu sürelere molalar da dahildir. Personellerin otuz dakika yemek molası ile iki adet on beş dakika dinlenme mola

hakları vardır. Bu çalışmada yazarın ortaya koyduğu mola yaklaşımı literatürde bu konuda yapılan çalışmalara temel teşkil etmesi açısından önemlidir. Bu tezdeki mola yaklaşımında da Aykin'in modelinden esinlenilmiştir.

Thompson [57], çalışmasında simülasyon analizi ile işgücü maliyetleri, servis düzeyi ve işgücü kullanım oranı açısından farklı vardiya ve personel KTSP modellerini karşılaştırmıştır. Bu çalışmada sadece maliyetin en küçüklenmesinden ziyade çok kriterli amaç fonksiyonuna sahip bir bakış açısı ortaya koyulmuştur.

Ertogral ve Bamuqabel [25], çalışmalarında operatörleri esnek olan ve olmayan şeklinde iki gruba ayırarak vardiya çizelgeleme problemi için 2 farklı KTSP temelli model ortaya koymuşlardır. Esnek olmasını operatörün farklı dillere hâkim olması ile açıklamaktadırlar. Fakat bu çalışmada geliştirilen modeller personellerin mola zamanlarını belirlememektedir.

Dietz [3], çalışmasında personel çizelgeleme problemi için kuadratik tabanlı bir model geliştirmiştir. Geliştirdiği modeli klasik bir TSP modeli ile kıyaslayarak kuadratik modelin daha iyi sonuçlar verdiğini ortaya koymuştur. Bu çalışmada çakışan ve çoklu bir vardiya düzeni ele alınmış ancak yine çalışanların mola zamanları belirlenmemiştir.

Nah ve Kim [38], bir hastane rezervasyon çağrı merkezinin personel çizelgeleme problemi için TSP temelli bir model geliştirmişlerdir. Geliştirilen model gerçek bir vaka çalışması ile test edilmiş ve bu yaklaşım ile firmanın beklenen maliyeti ile çağrı terk etme oranının (abandonment rate) azalttığını tespit etmişlerdir. Çalışmadaki model işçilik, bekleme ve terk etme maliyetleri gibi çeşitli maliyetleri bir arada ele alması açısından dikkate değerdir. Çalışmada çalışanların mola zamanlarının belirlenme işlemi yapılmakta ancak planlama dönemi boyunca sadece 2 vardiya düzeni kullanılmaktadır.

Ásgeirsson ve Sigurðardóttir [15], çalışmalarında bu problem için bir TSP modeli önermişlerdir. Bu modeli bir yerel arama tabanlı algoritma (local search based

algorithm) ile kıyaslayarak TSP'nin makul zamanda çok daha iyi sonuçlar verdiğini belirtmişlerdir. Modellerini dört farklı şirket verileri üzerinde test etmişler ve yüksek kalitede uygulanabilir personel çizelgeleri sağlamışlardır.

Örmeci ve ark. [39] çalışmalarında çağrı merkezlerindeki personel çizelgeleme problemini; operasyonel maliyetleri, müşteri temsilcisi memnuniyetini ve müşteri servis hedeflerinin dengelenmesini amaçlayarak ele almışlardır. Günümüzde çağrı merkezlerinin çalışanlarına ulaşım hizmeti de verdiğini ve bunun operasyonel maliyetlerin önemli bir kısmını oluşturduğu vurgulanmış; bu nedenle geliştirilen KTSP modeline bu ulaşım gereksinimleri de dâhil edilmiştir.

Thompson ve Pullman [61], işgücü çizelgeleme problemlerinde gerçek zamanlı personel dinlenme molaları üzerine çalışmışlardır. Literatürde yer alan personel çizelgeleme çalışmalardan 64 tanesini incelemişler ve bu çalışmalar sadece %18'inin molaları dikkate aldığını belirtmişlerdir. Çalışmadaki bulgular ticari personel çizelgeleme yazılımları ile desteklenmiştir. Yazarlar çalışmalarının bazı istenmeyen sonuçlarından da bahsetmişlerdir. Bu olumsuz sonuçlardan biri, tüm molaların gerçek zamanlı olarak ele alınmasının daha az kârlı personel dağılımına sebep olduğudur. Bir diğeri ise, eğer bazı işçiler bazı molalara hiç atanmazsa ya da mola zamanı kısıtlamaları gevşetirse daha fazla mola gerçek zamanlı olarak atanabilir. Böyle bir durumda ise doğal olarak işgücü verimi azalabilecek ya da sözleşme yükümlülüklerinin ihlali durumu ortaya çıkabilecektir.

Ingolfsson ve ark. [32], entegre bir personel seviyesi belirleme ve personel çizelgeleme modeli geliştirmişlerdir. Düşük maliyetli personel çizelgelerini bulmak için, gün içi her bir periyotta belirli bir minimum değerin üzerinde olan değişen zamanlı servis seviyeli bir yöntem ortaya koymuşlardır. Daha önce kullanılan yaklaşımların iki aşamalı olduğunu (personel gereksinimlerini belirlemek ve her periyotta gerekli personelin sağlandığı bir minimum maliyet çizelgesi bulmak) ve bu tip yaklaşımlarda ilk aşamanın ikinci aşama için bazen olursuz ve/veya yetersiz çözümler sunduğunu belirtmişlerdir. Ortaya koydukları yöntem, bir çizelge değerlendiricisi ve çizelge oluşturucusu arasında tekrar eden bir yöntemdir. Çizelge

değerlendiricisi, rastgeleleştirme yöntemini (randomization method) kullanarak geçici servis seviyelerini hesaplar ve servis seviyesinin arzu edilenin altında olduğu olursuz aralıkları tanımlar. Çizelge oluşturucusu ise iyileştirilmiş çizelgeler oluşturmak amacıyla her biri olursuz aralık için kısıt ekleyerek, optimal çözümü ortadan kaldırmadan olursuz çözümleri ortadan kaldırmaya çalışan bir dizi tamsayı programı çözer.

Taşkıran ve Zhang [44], vardiya çizelgelemesi, izin günleri ve mola atamalarını içeren yetenek tabanlı bir TSP modeli ortaya koymuşlar ve modeli çözmek için iki aşamalı sıralı bir yaklaşım geliştirmişlerdir. Geliştirilen modeli uygun test örnekleri ile sınayarak, yaklaşımlarının, diğer modellerin çözümüne göre daha iyi sonuçlar verdiğini göstermişlerdir. Sonuç olarak işçilerin yeteneklerine göre eğitilmesinin ve çizelgelenmesinin önemli ölçüde maliyet tasarrufu sağladığını göstermişlerdir.

Yine literatürde bu çalışmalar gibi TSP/KTSP modelleri ile yapılan çalışmalar oldukça fazladır [5, 25, 62, 63]. Öte yandan literatürde sadece sezgisellerin kullanıldığı ya da hibrit çalışmalar da mevcuttur [20, 30, 32, 34].

Özellikle Canon'un çalışması [20], çağrı merkezlerinin işgücü planlamasında KP'nin kullanımı perspektifinden bu çalışmadakine benzemektedir. Canon çalışmasında ilk üç alt problemi, KTSP, KP ve Tabu Arama Algoritması (Tabu Search) kullanarak modellemiştir. Yani bu tezden farklı olarak, Canon ikinci alt problemi de (personel seviyesi belirleme) ele almıştır. Fakat çalışmasında mola zamanlarını belirleme için sadece yemek molaları dikkate alınmış olup ayrıca modeller bilinen referans setler üzerinde test edilmemiştir.

2.2. Kısıt Programlama Literatürü

Bilindiği üzere değişken ve çözüm uzaylarının sayısı matematiksel modellerin temel özellikleridir. Değişken boyutlarının artması veya kötü tanımlanmış çözüm uzayları TSP modellerinin problemi tam olarak kapsayamamasına ve dolayısıyla optimum

çözüm elde edilememesine neden olmaktadır. Yani problem boyutu büyüdüğünde TSP temelli modeller problemi çözmek için çok fazla zaman harcamaktadırlar [64].

Son 30 yılda, bu sorunun üstesinden gelinebilmesi için bilgisayar bilimciler farklı değişken veri yapıları, grafik teorisi, yapay çıkarım, yeni bilgisayar programlama teknikleri ve dal-sınır ağaçları (branch and bound trees) tekniklerini birleştirici bir çerçeve geliştirmişlerdir. Bu çerçeve KP olarak adlandırılmaktadır ve KP, günümüzde karmaşık Kısıt Sağlama Problemleri (KSP) ve Kısıt Optimizasyon Problemleri (KOP) çözmek için kullanılan modern bir yöntemdir. Çeşitli uygulama alanlarındaki gerçek hayat problemlerini çözmekte kullanılabilmektedir [64]. KP, İÇP'ler gibi zorlu optimizasyon problemleri için matematiksel programlamaya göre daha esnektir [65].

Ernst ve ark. [4] işgücü çizelgeleme problemleri için kapsamlı bir literatür çalışması ortaya koymuşlardır. Çalışmalarında literatürde bu alandaki çalışmaları işgücü çizelgeleme metodları açısından da sınıflandırmışlardır. Metodlar beş farklı gruptan oluşmakta ve KP bu sınıflardan biridir.

Yine Van den Bergh ve ark. [1]'nin çalışması da işgücü çizelgeleme problemlerinin kapsamlı bir literatürünü içermektedir. KP, bu çalışmada da bu tarz problemlerin çözüm tekniklerine dahil edilmiştir.

İÇP literatürünün farklı uygulama alanlarında KP kullanımı göze çarpmaktadır. Bu uygulama alanlarına özellikle hemşire çizelgeleme [66–70] ve sağlık/bakım [71, 72], ulaşım [73], genel [74–77], perakende [78], askeri [79] ve spor [80] örnek olarak verilebilir. “Genel” olarak ifade edilen çalışmalar KP'nin teorik olarak işlendiği çalışmaları içermektedir.

Rahimian ve ark. [70] çalışmalarında oldukça kısıtlı hemşire çizelgeleme problemini etkin bir şekilde çözmek için TSP ile KP yaklaşımlarını birleştiren bir melez algoritma geliştirmişlerdir. Algoritmalarını çeşitli problem örneklerinden oluşan iki farklı veri kümesi üzerinde test etmişler ve algoritmalarının literatürde bu alanda

geliştirilen algoritmalarla yeterli düzeyde rekabetçi sonuçlar elde ettiğini göstermişlerdir.

Bertels ve Fahle [72] çalışmalarında evde sağlık hizmetlerinin planlanması ile ilgili bir algoritma sunmuşlardır. Evde sağlık hizmetinden kasıt evde hasta ziyareti ile hemşirelik hizmetleridir. Bu sektörün sağlık hizmeti sektöründe büyüyen bir sektör olduğunu vurgulamışlardır. Ele aldıkları problem temel olarak hemşire çizelgeleme problemine benzese de buna ek olarak ulaşım optimizasyonu problemini de içermesi açısından dikkate değerdir. Geliştirilen model, lineer programlama ve kısıt programlama tabanlıdır. Model, hem ulaşım maliyetlerini en aza indirmek hem de hasta ve hemşirelerin memnuniyetini en üst düzeye çıkarmak gibi kombine bir amaç fonksiyonu içermektedir. Geliştirdikleri modeli 120 farklı senaryo (20-50 hemşire, 80-200 hasta, 200-600 iş/gün) ile test etmişler ve oldukça etkili sonuçlar elde etmişlerdir.

Yunes ve ark. [73] çalışmalarında Brezilya'nın büyük bir kentinde hizmet veren şehir içi otobüs şirketinin genel ekip çizelgeleme problemi ele alınmıştır. Problem iki ayrı alt problem içermektedir: ekip çizelgeleme (crew scheduling) ve personel çizelgeleme (crew rostering). Bu problemlerin çözümü için temel olarak KP tabanlı bir model geliştirmişlerdir. Ayrıca, matematiksel bir model ile KP'yi birleştiren hibrit bir algoritma oluşturmuşlardır. Önerilen algoritmalar, söz konusu şirketten elde edilen gerçek dünya verileri üzerinde uygulanmış ve test edilmiştir. Deneylerinin analizi, geliştirilen modeller ile şirketin ihtiyaçlarına uygun, yüksek kaliteli ve çoğu zaman en uygun çözümleri bulmanın mümkün olduğunu göstermektedir. Bu çözümler bir masaüstü bilgisayarda makul hesaplama süreleri içinde elde edilmiştir.

Chapados ve ark. [78] çalışmalarında bir perakende satış mağazası personel çizelgeleme problemini ele almışlardır. Problem için işletmenin beklenen kârlarının maksimizasyonunu sağlayacak KP tabanlı bir model geliştirmişlerdir. Modellerini orta ölçekli bir perakende mağazasında yapılan bir vaka çalışması ile test etmişlerdir. Bu yaklaşım ile, ilgili mağaza açısından %7'lik kâr artışı elde etmişlerdir.

Li ve Womer [79] çalışmalarında Amerikan Deniz Kuvvetleri'ne ait bir filo için mürettebat çizelgeleme problemini ele almışlardır. Çalışmada bu problem birden çok yetenekli denizcilerin ekipler arası bütünleşik görevleri yerine getirmesi için ekip oluşturduğu bir “çok amaçlı kaynaklarla proje planlama problemi” (a project scheduling problem with multi-purpose resources) olarak KP ile modellenmiştir. Geliştirilen KP modelin performansı çeşitli örnekler üzerinde test edilmiş ve ayrıca aynı problem için geliştirilen bir KTSP modeli karşılaştırılmıştır. Deneysel sonuçlar, KP modelinin tüm test örneklerine uygun çözümler bulabildiğini ve tatmin edici bir çözüm kalitesi sağladığını göstermektedir. Problemin boyutu arttıkça, KTSP'ye nazaran KP modelin avantajı hem çözüm kalitesinde hem de çözüm hızındaki sağladığı artıştır. Sonuçlar, geliştirdikleri KP modelin gerçek dünyadaki büyük boyutlu problemlerle başa çıkmada umut verici olduğunu göstermektedir.

Larson ve ark. [80] çalışmalarında İsveç hentbol liginin çizelgelenmesi problemini ele almışlardır. Literatürdeki önceki çalışmalarda lig veya turnuva fikstürünün hazırlanması gibi sportif faaliyetlerin çizelgelenmesi işleminin daha kolay alt problemlere ayrılarak gerçekleştirildiğini belirtmişler, bu durumda optimal olmayan sonuçlarla karşılaşılabilceğini ifade etmişlerdir. Geliştirdikleri KP model ile bu problemin alt problemlere ayrılmaksızın bir bütün olarak ele alınabileceğini göstermişlerdir. Model, ele alınan çok sayıda senaryo ile test edilmiş ve uygun hesaplama zamanında optimal çözümler üretebildiği ortaya konulmuştur.

Literatürde İÇP'ler için KP kullanımının olduğu daha birçok çalışma yer almaktadır. Sonuç olarak KP tabanlı modeller, çok ve karmaşık kısıtlar içeren İÇP'ler için ideal bir çerçeve sunmaktadır [1].

BÖLÜM 3. METODOLOJİ

Araştırmacılar işgücü çizelgeleme problemleri için literatürde birçok farklı yöntem kullanmışlardır. Tablo 2.2.'de de görüleceği üzere bu yöntemler bazı çalışmalarda tekil olarak kullanılmış, bazı çalışmalarda da çoklu (birden çok ve birbirinden bağımsız) ya da hibrit (birden çok ve iç içe/bağımlı) olarak kullanılmıştır.

Tezde ele alınan problemler birbirini tamamlayıcı özellikte olup bu bölümde, ele alınan problemlerin modellenmesi hakkında sırasıyla bilgi verilecektir. Modellere ait kümeler, parametreler ve karar değişkenleri gibi girdiler tanımlanacak olup genel model gösterimleri detaylı olarak açıklanacaktır. Ayrıca tüm modellere ait CPLEX kodlarına ekte ayrı ayrı yer verilmiştir. Eğer bir girdi daha önce tanımlanmış ve tekrar farklı bir modelde kullanılmışsa tekrar bir tanımlama yapılmamıştır. Fakat kıyaslama örnekleri için kullanılan matematiksel model farklı bir kaynaktan olduğu gibi alındığı için o model (Model 3a) ve kıyaslama test örnekleri için geliştirilen KP temelli model (Model 3b) için diğerlerinden bir anlamda daha bağımsız bir matematiksel gösterim yapılmıştır.

3.1. Vardiya Çizelgeleme Problemi İçin Geliştirilen Model (Model 1)

Çağrı merkezlerinde periyodik düzeyde gerekli operatör sayısı belirlendikten sonraki problem bu operatör ihtiyaçlarını en iyi şekilde karşılayacak olan vardiyaların seçilmesidir. Bu problemin çözümü ile hangi operatörün hangi vardiyaya atandığından ziyade hangi vardiyalardan kaçar adet seçileceği/açılacağı ve seçilenlere/açılanlara toplam kaçar adet operatör atanacağı belirlenmektedir.

Bir önceki alt problemde periyot başına gerekli olan operatör sayısı hesaplanırken servis düzeyi, kullanılan yöntemler (Erlang formülasyonları, simülasyon veya

diğerleri) içerisinde gömülü olarak bulunmaktadır. Yani personel seviye belirleme probleminin çözümü neticesi, belirlenen servis düzeyini karşılayacak düzeyde operatör kullanımını ortaya koymaktadır. Dolayısıyla vardiya çizelgeleme probleminin çözümü sonucunda seçilecek olan vardiyalar da talebi en uygun şekilde karşılamının yanında dolaylı olarak bu amaca hizmet edecektir.

Talebi en uygun şekilde karşılamadan kasıt periyot başına fazlalık olan kısmın en küçüklenmesidir. Bunun yanında bir de herhangi bir vardiyanın açılıp açılmamasına karar verilmesi gerekmektedir. Zira bir vardiyanın açılmasının da bazı maliyetleri söz konusu olacaktır. Ayrıca periyot başına istenen operatör talebinin karşılanamaması gibi bir durumda olabilir. Bunun da bir maliyeti vardır.

Bu çalışmada, özetle vardiya çizelgeleme problemi için açılacak vardiya sayısının, periyot başına istenen talebin aşılmasının ve periyot başına istenen talebin karşılanamamasının maliyetini minimize eden bir model önerilmiştir. Model detayları aşağıda verilmiştir:

Kümeler

D	Planlama dönemi boyunca günler kümesi ($\forall d \in D = \{1, \dots, D \}$)
S	Bir gündeki vardiyalar kümesi ($\forall s \in S = \{1, \dots, S \}$)
P	Bir gündeki periyotların kümesi ($\forall p \in P = \{1, \dots, P \}$)

Parametreler

C^P	Periyot başına çalışma maliyeti
C^S	Vardiya açma maliyeti
C^U	Periyot başına talebin karşılanmama maliyeti
E_{sp}	s . vardiyanın p . periyodu içerip içermediği (içeriyorsa 1, değilse 0)
γ	Bir vardiyaya atanabilecek en çok operatör sayısı
N_{dp}	d . gün p . periyot için ihtiyaç duyulan operatör sayısı

Karar değişkenleri

a_{ds}	d . gün s . vardiyaya atanan operatör sayısı ($d \in D, s \in S$)
p_{ds}	d . gün s . vardiya seçilirse 1, diğer durumda 0. ($d \in D, s \in S$)
u_{dp}	d . gün p . periyotta karşılanmayan talep sayısı ($d \in D, p \in P$)

Karar ifadeleri

d . gün p . periyottaki fazlalık operatör sayısı aşağıdaki Denklem 3.1 ile hesaplanır:

$$r_{dp} = \sum_{s \in S} (E_{sp} \cdot a_{ds}) - N_{dp} \quad (\forall d \in D, \forall p \in P) \quad (3.1)$$

Matematiksel model

Amaç fonksiyonu

$$\text{Min.} \left(\sum_{\substack{d \in D, \\ p \in P}} C^P \cdot r_{dp} + \sum_{\substack{d \in D, \\ s \in S}} C^S \cdot p_{ds} + \sum_{\substack{d \in D, \\ p \in P}} C^U \cdot u_{dp} \right) \quad (3.2)$$

Önerilen modelin amaç fonksiyonu (Denklem 3.2) sırasıyla periyot başına fazla operatör maliyetinin, vardiya açma maliyetinin ve periyot başına karşılanmayan talebin minimize edilmesi esasına göre oluşturulmuştur. Maliyet parametreleri her operatör açısından aynıdır. Zira operatörler arasında verimlilik ve kapasite farkı olmadığı varsayılmaktadır. İleriki zamanlarda yapılacak çalışmalarda kapasite/yetenek seviyeleri de probleme ayrıca dahil edilebilir.

Kısıtlar

Modelde yer alan kısıtların matematiksel gösterimi Denklem 3.3 ve Denklem 3.4'te gösterildiği şekildedir.

$$\sum_{s \in S} (E_{sp} * a_{ds}) + u_{dp} \geq N_{dp} \quad (\forall d \in D, \forall p \in P) \quad (3.3)$$

$$p_{ds} \leq a_{ds} \leq \gamma \cdot p_{ds} \quad (\forall d \in D, \forall s \in S) \quad (3.4)$$

Önerilen modelde görüldüğü üzere 2 adet ana kısıt yer almaktadır. Bu kısıtların açıklaması şu şekildedir:

1. Denklem 3.3 ile her günün her bir periyodunda ihtiyaç duyulan operatör sayısı garantilenmiş olur.
2. Denklem 3.4 ile eğer bir vardiya açıksa o vardiyaya atanabilecek en çok operatör sayısını aşmayacak ve 0 atama olmayacak (hiç atama yapılmaması) şekilde operatör atanır. Açılmamış vardiyaya operatör atanması engellenir.

Ayrıca modelin Denklem 3.5 ve Denklem 3.6'da gösterildiği gibi alan kısıtları (domain constraints) vardır.

$$p_{ds}, E_{sp} \in \{0,1\} \quad (\forall d \in D, \forall s \in S, \forall p \in P) \quad (3.5)$$

$$a_{ds}, u_{dp}, N_{dp}, \gamma \geq 0 \text{ ve tamsayı} \quad (\forall d \in D, \forall s \in S, \forall p \in P) \quad (3.6)$$

Modele ait CPLEX kodlarına Ek-A'da yer verilmiştir.

3.2. Personel Çizelgeleme Problemi İçin Geliştirilen Modeller

Bu bölümde mevcut operatörlerin bir önceki problem (vardiya çizelgeleme) çözümünü neticesinde oluşturulan vardiyalara atanması, operatörlerin mola ve izin günlerinin belirlenmesi problemine yönelik geliştirilen TSP ve KP tabanlı modellerin detaylarına sırasıyla yer verilecektir.

3.2.1. Tam sayılı programlama modeli (Model 2a)

Personel çizelgelemesi için geliştirilen TSP modeline ait olup daha önce tanımlanmayan girdiler ve model yapısı aşağıdaki gibidir:

Kümeler

A	Operatörler kümesi ($\forall a \in A = \{1, \dots, A \}$)
Y	Mola tipleri kümesi ($\forall y \in Y = \{1, \dots, Y \}$)
B_{ys}	s . vardiyadaki y tipi molanın zaman aralıkları kümesi
S_p^O	p . periyot ile zamansal olarak örtüşen vardiya kümesi

Parametreler

C^H	Saatlik çalışma maliyeti
G_{ds}	d . gün s . vardiyaya atanacak toplam operatör sayısı
H	Planlama dönemindeki iş günü sayısı
L_y	y tipindeki molanın periyot olarak uzunluğu
W_s	s . vardiyanın toplam saat uzunluğu

Karar değişkenleri

x_{ads}	Eğer a . operatör d . gün s . vardiyaya atanırsa 1, diğer durumda 0. ($\forall a \in A, \forall d \in D, \forall s \in S$)
z_{adsyp}	Eğer a . operatör d . gün s . vardiyaya atanırsa ve y tipindeki molasına p . periyotta başlarsa 1, diğer durumda 0. ($\forall a \in A, \forall d \in D, \forall s \in S, \forall y \in Y, \forall p \in P$)

Karar ifadeleri

d . gün p . periyotta y tipi molada olan toplam operatör sayısı Denklem 3.7 ile bulunur:

$$W_{dyp} = \sum_{\bar{p}=p-L_y+1}^p \left(\sum_{\substack{a \in A, \\ s \in S_p^O}} z_{adsy\bar{p}} \right) \quad (\forall d \in D, \forall y \in Y, \forall p \in P) \quad (3.7)$$

Matematiksel model

Amaç fonksiyonu

$$\text{Min.} \sum_{\substack{a \in A, \\ d \in D, \\ s \in S}} (C^H \cdot W_s \cdot x_{ads}) \quad (3.8)$$

Önerilen modelin amaç fonksiyonunu gösteren Denklem 3.8, toplam işgücü maliyetini en küçükleyecek şekilde düzenlenmiştir. Yine operatörler açısından bir farklılık olmadığı varsayılmıştır.

Kısıtlar

Önerilen modelde 5 tane ana kısıt yer almaktadır. Kısıtların matematiksel gösterimleri şu şekildedir:

$$\sum_{s \in S} x_{ads} \leq 1 \quad (\forall a \in A, \forall d \in D) \quad (3.9)$$

$$\sum_{\substack{d \in D, \\ s \in S}} x_{ads} \leq (H - 1) \quad (\forall a \in A) \quad (3.10)$$

$$x_{ads} - \sum_{p \in B_{ys}} z_{adsyp} = 0 \quad (\forall a \in A, \forall d \in D, \forall s \in S, \forall y \in Y) \quad (3.11)$$

$$\sum_a x_{ads} = G_{ds} \quad (\forall d \in D, \forall s \in S) \quad (3.12)$$

$$\sum_{s \in S_p^O} G_{ds} - \sum_y W_{dyp} \geq N_{dp} \quad (\forall d \in D, \forall p \in P) \quad (3.13)$$

Model 2a'daki ana kısıtlar ve açıklamaları şu şekildedir:

1. Denklem 3.9'da gösterilen kısıt ile operatörlerin planlama döneminde herhangi bir günde en fazla bir vardiyaya atanmalarına izin verilmektedir.

2. Denklem 3.10, operatörlerin planlama döneminde en az 1 gün tatil yapmalarına olanak sağlayan kısıtı ifade etmektedir.
3. Denklem 3.11 ile operatörlerin hem tüm dinlenme molalarını vermeleri garantilenir hem de bu molaları hangi periyotlarda verecekleri tespit edilir.
4. Denklem 3.12, ilgili gün ilgili vardiyaya atanacak operatör sayısını, vardiya çizelgeleme probleminin çözümü neticesinde elde edilen “*vardiyalardaki gün başına operatör sayısına*” eşitler. Yani vardiya çizelgeleme probleminin çözüm değerlerinden biri bu alt problemin bir parametresi olarak kullanılmaktadır.
5. Denklem 3.13’de gösterilen kısıt ile planlama dönemindeki günlerin her gün içi periyodunda olması gereken kadar operatörün çalıştırılması garantilenir (cover constraint).

Modele ait alan kısıtları ise Denklem 3.14-Denklem 3.16’da gösterildiği gibidir.

$$x_{ads}, z_{adsyp} \in \{0,1\} \quad \left(\begin{array}{l} \forall a \in A, d \in D, s \in S, y \in Y, \\ p \in P \end{array} \right) \quad (3.14)$$

$$G_{ds}, W_{dyp}, N_{dp}, L_y, W_s \geq 0 \text{ ve tamsayı} \quad (\forall d \in D, s \in S, y \in Y, p \in P) \quad (3.15)$$

$$H > 1 \text{ ve tamsayı} \quad (3.16)$$

Modele ait CPLEX kodlarına Ek-B’de yer verilmiştir.

3.2.2. Kısıt programlama, IBM ILOG CP ve geliştirilen KP model (Model 2b)

Bu bölümde, geliştirilen KP temelli modelin detaylarını vermeden önce KP’nin genel yapısı ve KP çözücü olarak kullanılan IBM ILOG CPLEX Optimization Studio paket programı hakkında genel bilgiler verilecektir. Daha sonra geliştirilen KP modeline ait detaylara yer verilecektir.

3.2.2.1. Kısıt programlama

Kısıt programlama; yapay zekâ, yöneylem araştırması ve çizge teorisi (graph theory) gibi tekniklere dayanarak kombinatoriyal arama problemlerini çözmek için kullanılan

güçlü bir paradigmadır. KP'deki temel fikir, kullanıcının kısıtlamaları belirtmesi ve genel amaçlı bir çözücü aracılığıyla bu kısıtların çözülmesidir [81]. Tarihçesine bakıldığında kökleri 1960'lara uzanmasına karşın tam anlamıyla ortaya çıkışı 1980'lerdedir [82].

KP'nin başlıca avantajlarından biri, matematiksel kısıtlamalar, ayrık kısıtlamalar, ilişkisel kısıtlamalar ve mantıksal kısıtlamalar gibi çeşitli kısıtlama türlerinin ifade edilmesinde büyük esneklik sağlamasıdır [83]. Bu sayede kompleks problemlerin formüle edilmesinde ve çözülmesinde kolaylıkla kullanılabilir.

KP, çizelgeleme ve planlama gibi çeşitli uygulama alanlarında gerçek yaşam problemlerini çözmek için yaygın olarak kullanılmaktadır [84]. Geniş temsiliyet kapsamı ve genel olarak uygulanabilir çözümlerle optimizasyon problemlerini çözmede etkili olduğu kanıtlanmıştır [64].

KP, KSP ve KOP olarak adlandırılan problemleri çözmek için kullanılır. KSP, verilen kısıtları sağlayan değerlere atanan bir değişkenler dizisi olarak tanımlanabilir. Daha açık ifadeyle bir KSP'nin çözümü, tüm kısıtları sağlayacak şekilde her bir değişkene bir değer atanmasıdır.

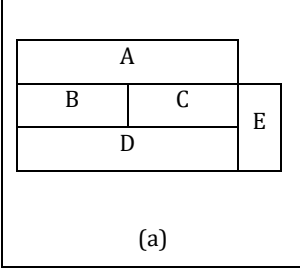
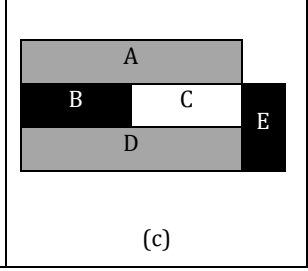
KOP ise basit olarak KSP'ye bir amaç fonksiyonunun (örneğin maliyet minimizasyonu) eklenmesi ve bu amaç fonksiyonunun optimizasyonu ile ilgili problemlerdir [82]. Yani temeli KSP'lerdir.

Bir KSP üç bileşenden oluşmaktadır [81]:

1. Bir değişkenler dizisi: $X = \{x_1, \dots, x_n\}$
2. Bir değer kümesi (domain): $v_i \in D(x_i), i = 1, 2, \dots, n$
3. Bir kısıtlar seti: $C = \{C_1, \dots, C_m\}$

Aşağıdaki Şekil 3.1.'de KSP'nin temel yapısı bir Harita Boyama Problemi (Problem of Map Colouring) üzerinde anlatılmıştır. Bu tür bir problemde, bir eyaleti veya

ülkeyi gösteren bir haritada her bir bitişik bölge sayıları sınırlı olan farklı bir renk gerektirmektedir. Örnekte de 5 farklı bölge bulunmakta (Şekil 3.1a) ve 3 farklı renk kullanılarak bitişik bölgelerin farklı renklerde olacak şekilde boyanması istenmektedir.

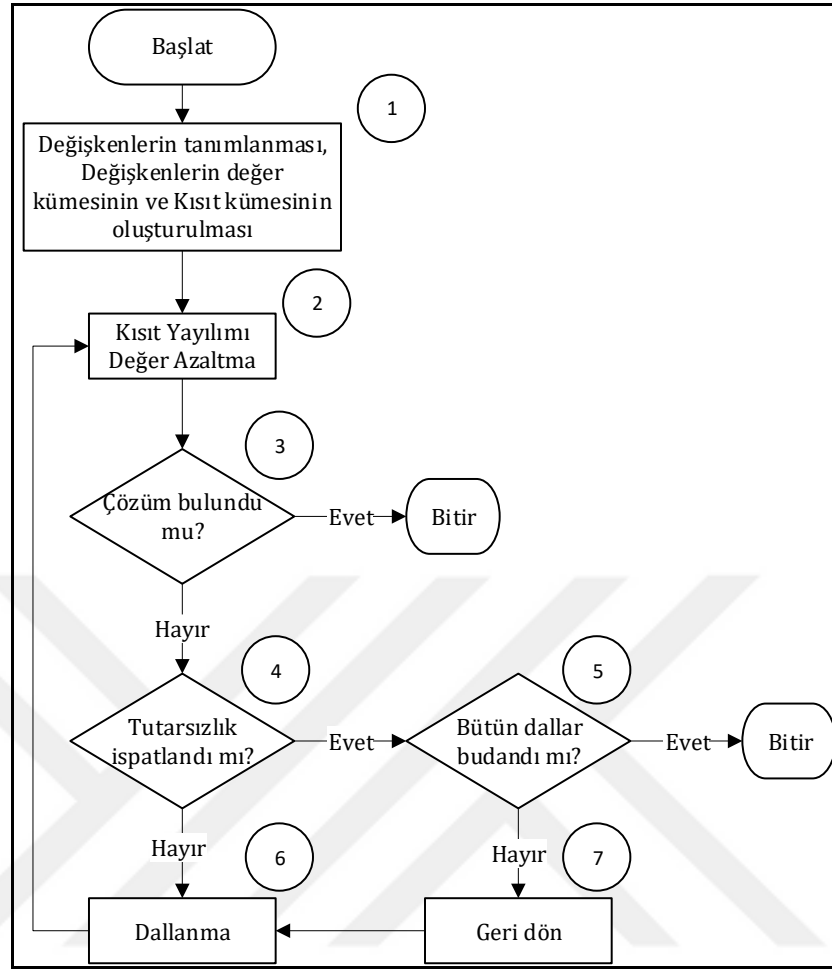
 (a)	Değişkenler : A, B, C, D, E Domain : Beyaz, Gri, Siyah Kısıtlar : $A \neq B, A \neq C, B \neq C, B \neq D,$ $C \neq D, C \neq E, D \neq E$ (b)	 (c)
--	---	--

Şekil 3.1. Bir harita boyama probleminin KSP olarak gösterimi.

Problemin KSP formundaki tanımı Şekil 3.1b'deki gibidir. Değişkenler (A, B, C, D, E) bölge adlarını temsil etmektedir. Değer kümesi (domain) ise her bir değişkenin alabileceği olası değerleri ifade etmektedir (*Beyaz, Gri, Siyah*). Kısıtlar vasıtasıyla da değişkenlerin alabileceği değerler sınırlandırılmış olmaktadır.

Örneğin; $A \neq B$ kısıtı, A bölgesi ile B bölgesinin renklerinin aynı olmamasını ifade etmektedir. Kısıtların sadece doğrusal olmadığı, mantıksal ve global kısıtların da kullanılabildiğine dikkat çekmek gerekmektedir. Kullanılacak bir çözücü ile tüm kısıtların sağlandığı ve değişkenlere değer kümesi içinden değer atandığı bir çözüm elde edilmektedir. Örnek bir çözüm de Şekil 3.1c'de gösterilmiştir.

Bir KSP'nin süreç işleyiş adımları Şekil 3.2.'de gösterilmiştir [85]. Algoritma; değişkenlerin, değer kümesinin ve kısıtların tanımlanması ile başlar (1). İkinci aşama KP'nin en önemli süreci olarak da adlandırılan Kısıt Yayılımı (Constraint Propagation) ve Değer Azaltma (Domain Reduction) işlemlerinin yapıldığı aşamadır (2). Bu kavramlar daha sonra detaylı olarak incelenecektir. (3) ile ifade edilen aşamada çözümün bulunup bulunmadığının kontrolü yapılır. Eğer çözüm bulunmuşsa algoritma sonlandırılır. Eğer çözüm bulunamamışsa tutarsızlık kontrolünün yapıldığı aşamaya geçilir.



Şekil 3.2. KSP'nin genel bir algoritması [85].

(4) ile gösterilen aşamada tutarsızlık kontrolü yapılmaktadır. Tutarsızlık en az bir değişkenin çözüm kümesinden hiçbir elemana sahip olmaması (boş olması), değer almamış olması durumudur. Eğer tutarsızlık ispatlanmadıysa dallanma (branching) yapılarak bazı arama stratejileri ile (daha sonra değinilecektir) çözüm aranmaya devam edilir (6). Dallanma, bir ana problemi geçici kısıtlar kullanarak alt problemlere bölme işlemidir. Bu durumda tekrar (2) ile gösterilen aşamaya geri dönülür.

Eğer (4)'te tutarsızlık ispatlandı ise (5)'te gösterildiği üzere tüm dalların budanıp (fathoming) budanmadığının kontrolü yapılır (7). Dalların hepsi budandı ise algoritma sonlanır. Bu durumda bir çözüm üretilmiştir fakat çözüm tutarlı değildir. Eğer bütün dallar budanmadı ise algoritma bir önceki aşamaya geri döner (backtracking) ve dallandırma yapılır (6).

Kısıt Yayılımı ve Değer Azaltma

Matematiksel programlamada, kısıtlar sadece bir çözümün uygunluğunu test etmek için kullanılır. Öte yandan, KP’de değer kümesinden (domain) bazı olası değerler elenebilir, yeni kısıtlar eklenebilir ve kısıtlara göre tutarsızlık test edilebilir [83]. İşte KP’nin ana fikri, kısıtları hesaplama çabasını azaltmak için Kısıt Yayılımı ve Değer Azaltma süreçlerinde aktif olarak kullanmasıdır [82].

Kısıt Yayılımı ve Değer Azaltma’nın temel amacı, arama alanını küçülterek, kesin bir şekilde olursuz çözümlere yol açacak olan değerleri değişken alanlarından kaldırmaktır [82]. Bu işlemler literatürde bazı algoritmalar vasıtasıyla yapılmaktadır. Bunlara ileriye doğru kontrol (forward checking), ark tutarlılığı (arc consistency), k-tutarlılık (k-consistency) gibi algoritmalar örnek olarak verilebilir [81].

Arama (Search)

KSP’ler için diğer önemli bir süreç aramadır [81]. Bu konuyu daha iyi ifade etmek için öncelikle arama ağaçları hakkında ön bir bilgi vermek gerekmektedir. Bilindiği üzere bir problemin modellenmesinde üzerinde düğümlerin ve dalların olduğu arama ağaçları kullanılmaktadır. Bu düğüm ve dallar, değişkenleri ve olası çözümleri ifade etmektedir. Arama denilen süreç bu düğümlenen/dallanan yapının araştırılması ve tüm değişkenlerin değer almasının sağlanmasıdır.

Literatürde arama için kullanılan birçok algoritma bulunmaktadır. Bunlara geri izleme (dönüş) algoritması (backtracking algorithm), evrimsel algoritmalar (evolutionary algorithms), büyük komşuluk arama (large neighborhood search) algoritması ve makine öğrenmesi (machine learning) örnek olarak verilebilir [83].

Burada özellikle geri izleme (dönüş) algoritması KSP için önem arz etmektedir. Bu algoritma KSP’ler için temel algoritmadır [81]. KSP’de herhangi bir dalda çözüme ulaşılmadıysa geri dönüş mekanizması denilen bir süreç başlatılır. Bu sayede arama ağacının en son noktasında başarısız olması muhtemel dal üzerinde gereksiz zaman

harcaması yapmamak için bir geri atlama ile tutarsızlığa neden olan en baştaki düğüme gidilmektedir. Böylece tüm dalların araştırılması daha kolaylıkla tamamlanmaktadır [82].

Bu algoritmalar dallanma yapmak üzere çeşitli stratejiler kullanmaktadırlar. Bu stratejilerden bazılarına aşağıda yer verilmiştir [86]:

1. Derinlik öncelikli arama stratejisi (depth-first search): Bu tip arama stratejisinde arama en alt düzeydeki düğümden başlatılır. İlgili düğüm ile ilgili çözümler ortaya konduktan sonra diğer düğümlere geçilir. Çözücü, üzerinde çalıştığı dalı tamamen budamadan başka bir dala geçmemektedir.
2. Yeniden başlatma stratejisi (restart search): Bu stratejide önceden belirlenen bir hata sayısına ulaşıncaya kadar arama sürdürülür. İlgili değere ulaşıncaya arama tekrar başlatılır.
3. Çok noktalı arama stratejisi (multi point search): Çözüm uzayının aynı anda birçok noktasında arama yapma stratejisidir.
4. Otomatik arama stratejisi: Çözücünün arama esnasında yukarıdaki arama stratejilerden birine otomatik olarak karar verdiği arama stratejisidir.

Kısıt Optimizasyon Problemleri

KSP, tüm kısıtları karşılayan uygun bir çözüm sunar. Fakat gerçek hayatta, maliyetin veya zamanın en aza indirilmesi ya da kârın en çoklanması gibi kesin bir hedefle optimal veya nispeten daha iyi bir çözüm bulmaya çalışılır. İşte bu tarz problemler KSP'ye bir amaç fonksiyonu eklenmiş hali olan KOP'lardır. Yani sadece kısıtları sağlayan bir çözüm kümesi bulunmasından ziyade hem kısıtları sağlayan hem de amaç fonksiyonunu en iyileyen çözüm kümesi ortaya konmaktadır. KOP'ların çözüm algoritması KSP'lere dayanmaktadır [84].

Bu tarz problemlerde çözüm süreci içerisinde alt ve üst sınır (lower and upper bounds) hesaplamaları devreye girer. Arama sürecinde amaç fonksiyonu, değeri yeni bir değerden daha fazla ya da daha az olmaya (problemin türüne göre) zorlayan bir

kısıt olarak yeniden yazılır. Arama, alt sınırın üst sınıra eşit olduğu veya tüm düğümlerin budandığı zaman sona erer [83].

KP'nin genel olarak ana özellikleri şunlardır [87]:

1. Heterojen kısıt kullanımına (doğrusal, doğrusal olmayan, mantıksal vb. kısıtların bir arada kullanılması) ve konveks olmayan çözüm uzaylarını ele almaya izin vermesi,
2. Problemin boyutundan bir anlamda bağımsız olabilmesi,
3. Optimizasyon Programlama Dili (OPD) kullanımına olanak vermesidir.

Dahası, KP modeller daha az hesaplama çabası gerektirerek gerçek vakalar için daha yararlı analizler sağlamaktadır [88].

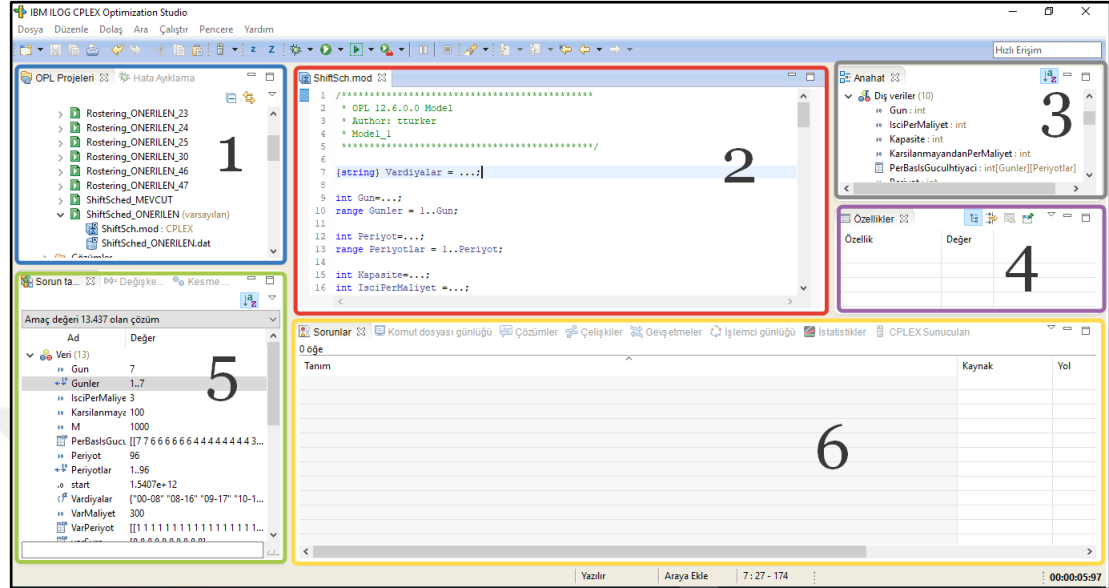
3.2.2.2. IBM ILOG CPLEX Optimization Studio ve KP

ILOG CPLEX Optimization Studio, IBM firması tarafından üretilen ve desteklenen, matematik ve kısıt programlamayı kullanarak karar optimizasyonu modellerinin hızlı bir şekilde geliştirilmesini ve kullanılmasını sağlayan bütünsel bir geliştirme ortamıdır. İçerisinde OPD barındıran yüksek performanslı CPLEX ve CP Optimizer (KP Eniyileme) çözücülerini destekleyen özelliğe sahiptir [89].

OPD, genel amaçlı programlama dillerinden çok daha az çaba gerektiren, optimizasyon modellerinin doğal bir temsilini sağlayan bir programlama dilidir [90]. Bu programdaki CPLEX modülü matematiksel modelleme ile ilgili bir çözücüdür. CP Optimizer ise KP modelleri ile ilgili bir çözücüdür.

CP Optimizer, kısıt tabanlı kütüphaneler ve araçlar içeren piyasadaki en başarılı çözücülerden biridir. KP için gerekli tekniklerin çoğunu içermekte olup, kısıtlamaya dayalı yapılandırıcı ve programlayıcı kütüphanelerini kullanan C++ ve Java tabanlı kısıt kütüphanelerine de sahiptir [81].

Programın ekran görüntüsü Şekil 3.3.'de gösterilmiştir. Program, şekilde de görüldüğü üzere temel olarak 6 farklı bölüm içermektedir.



Şekil 3.3. ILOG CPLEX Optimization Studio programı genel ekran görüntüsü.

Şekil 3.3.'de gösterilen 1 numaralı bölüm geliştirilen modele ait dosyaların (model, veri, ayar dosyaları vb.) ve bu dosyalar ile kurgulanan çalıştırma düzenlerinin bulunduğu bölümdür. 2 numara ile gösterilen bölüm ise modellere ait temel model dosyalarının, model dosyalarına bağlı çalıştırılacak veri dosyalarının ve program parametrelerini düzenlemeye yönelik ayar dosyalarının düzenlendiği bölümdür. Kısaca OPD kullanılarak modellerin kodlandığı, gerekli veri tanımlamalarının yapılabildiği kısımdır.

Programın 3 numaralı bölümü kullanıcılara geliştirdikleri modelin ana hatlarını göstermektedir. Bir anlamda yazılan kodların simgesel ve tanımsal özetini sunmaktadır. 4 numaralı bölüm ise çözüm neticesinde ortaya çıkan değerlerin tekil veriler kapsamında kısa bir özetini gösterirken, 5 numaralı bölüm tüm veri yapılarına ait çözüm sonuçlarını içeren daha detaylı bir özet sunmaktadır.

Programda bir modelin çözülmesi neticesinde ortaya çıkan asıl sonuçlar (çözüm değerleri, çalıştırma performans bilgileri vb.) Şekil 3.3.'deki 6 numaralı bölümde

gösterilmektedir. Şekilde de görüldüğü üzere bu bölüm 8 ayrı iç bölüm (tab) içermektedir. Bu bölümler ve tanımları şu şekildedir:

1. Sorunlar: Bu bölümde varsa modelin çalıştırılması esnasında ortaya çıkan yanlış veri tanımlamaları gibi OPD kodlama hatalarından kaynaklanan sorunlar ve modelde yazılmasına karşın kullanılmayan veri yapıları gösterilmektedir. Kodlama açısından bir sorun veya kullanılmayan bir veri yapısı yoksa bu bölüm boş olmaktadır.
2. Komut dosyası günlüğü: Bu bölümde kullanıcıların eğer kodlama sırasında ayrıca kod yazarak temel çözümlerden farklı olarak sonuçta gösterilmesini istedikleri bilgiler varsa bu bilgilere yer verilmektedir.
3. Çözümler: Bu bölümde temel model çözümleri (amaç fonksiyonu değeri, karar değişkenlerine ait çözüm değerleri vb.) yer almaktadır.
4. Çelişkiler: Bu bölümde model çözümü neticesi özellikle optimum olmayan sonuç olması durumunda eğer varsa çelişkiler (conflicts) yer almaktadır.
5. Gevşetmeler: Bu bölümde model çözümü neticesinde eğer program çözücüsü tarafından yapıldı ise gevşetmelere (relaxations) yer verilmektedir.
6. İşlemci günlüğü: Bu bölümde modelin çözümü esnasında programa ait çözücünün teknik işlemleri gösterilmektedir.
7. İstatistikler: Bu bölümde modelin çözümü neticesinde ortaya çıkan bazı istatistiki sonuçlar (amaç fonksiyonu değeri, toplam kısıt/değişken sayısı, toplam çözüm havuzu vb.) yer almaktadır. Ayrıca çözücünün çalışma sürecince bulduğu sonuçlar sonuç/zaman temelli bir grafikte gösterilmektedir.
8. CPLEX sunucuları: Bu bölümde eğer bir CPLEX sunucusu kullanımı varsa o sunucuya ait bilgiler yer almaktadır.

ILOG CPLEX Optimization Studio'da geliştirilen bir model genel olarak aşağıdaki formattadır [89]:

1. Parametreler
2. Karar değişkenleri
3. Karar ifadeleri

4. Amaç fonksiyonu
5. Kısıtlar

Programın içinde barındırdığı OPD, problemleri daha kolay modellemek için tasarlanmış özel değişkenler, kısıtlar ve anahtar kelimeler kullanmaya imkân verir. Programın detayları için kaynakçada da yer verilen kullanım rehberleri incelenebilir [86, 90].

Bu aşamada programda geliştirilebilecek modellerin formatı hakkında bazı genel bilgiler verilecektir.

Parametreler

CPLEX'te modele ait parametreler, parametre tipleri ve parametre yapıları olarak iki başlıkta incelenebilir [90]. Parametre tipi olarak kesikli (integer), sürekli (float) ve sözel (string) parametre tipleri kullanılmaktadır. Bazı örnek gösterimler aşağıdaki gibidir:

```
int i = 7; (Kesikli parametre tipi gösterimi, modelde  $i$ 'nin değeri 7)
float f = 5.6; (Sürekli parametre tipi gösterimi, modelde  $f$ 'nin değeri 5,6)
{string} isciler = { "Ali", "Veli", "Ayşe" }; (Sözel parametre tipi gösterimi)
```

Parametre yapısı olarak ise aralık (range), dizi (array), tanımlama grubu (tuple) ve küme (set) kullanılabilmektedir. Bazı örnek gösterimler aşağıdaki gibidir:

```
range gunler = 1..i;
```

Yukarıdaki ifade aralık parametre yapısı gösterimi olup modelde 1'den i 'e kadar numaralandırılan i (7) gün olduğunu ifade eder.

```
int izinA1 [isciler] = [4, 2, 5];
```

Yukarıdaki ifade ise tek boyutlu bir dizinin parametre yapısı gösterimidir. Modelde işçilerin önceden belirli izin günleri bu şekilde tanımlanmıştır. Örnekte 1. İşçinin (Ali) 4. Gün izinli olduğunu gösterir.

```
int izinA2 [isciler] [gunler] = [[0,0,0,1,0,0,0]
                                [0,1,0,0,0,0,0]
                                [0,0,0,0,1,0,0]];
```

Yukarıdaki matris çok boyutlu bir dizinin parametre yapısı gösterimi olup modelde 3 işçinin planlama dönemindeki izin günlerinin 0-1 olarak tanımlandığını ifade eder. İşçinin izin günü ise 1 ile, izin günü değilse 0 ile gösterilir.

```
tuple izinT {string isciler; int gunler;};
{izin} izinler = { <"Ali", 4>, <"Veli", 2>, <"Ayşe", 5>}
```

Yukarıdaki gösterim tanımlama grubunun parametre yapısıdır. Modelde 3 işçinin planlama dönemindeki izin günleri tanımlama grubu şeklinde yazılmıştır.

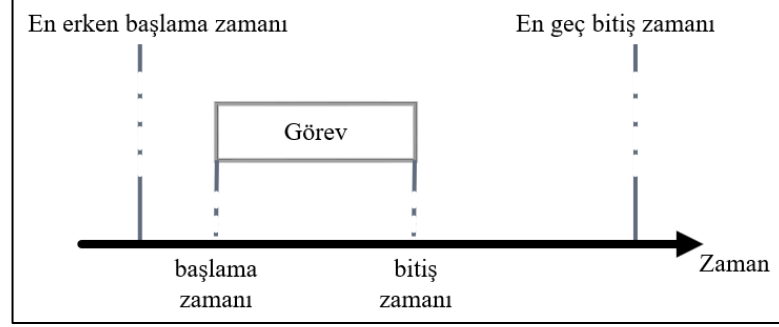
```
{int} calissin [isciler] = [{1, 2}, {5, 6, 7}, {4}];
```

Yukarıda küme parametre yapısı gösterimine yer verilmiştir. Modelde 3 işçinin planlama dönemindeki kesin olarak çalışılmasının istendiği günler bu şekilde tanımlanmıştır. Örnekte 2. İşçi (Veli)'nin 5., 6. ve 7. günlerde kesin olarak çalışması istenmektedir.

Karar değişkenleri

Karar değişkenleri bir problemde bilinmeyen bilgileri temsil eder. IBM ILOG CPLEX Optimization Studio farklı tip karar değişkenlerini kullanmaya olanak vermektedir. Bunlar kesikli (integer), sürekli (float), 0-1 (boolean), aralık (interval) ve sıralama (sequence) karar değişkenleridir. Burada diğerlerinden farklı yapıları nedeniyle aralık ve sıralama karar değişkenlerine değinmek istiyoruz.

TSP modellerinde kullanılan kesikli, sürekli ve 0-1 karar değişkenlerinden ziyade KP modelde karar değişkenleri interval ve sequence olarak tanımlanabilir. Bir interval karar değişkeni Şekil 3.4.'deki gibi tanımlanmaktadır.



Şekil 3.4. Bir aralık (interval) değişkeni.

Bir aralık değişkeninin kendine has 4 farklı özelliği şu şekildedir [89]:

1. Başlangıç
2. Bitiş
3. Boyut
4. Opsiyonellik

Aralık karar değişkeni için örnek bir kodlama gösterelim. Örnekte bir işçinin gün içerisinde dinlenme molası vereceği zamanı (periyotu) belirleyen bir karar değişkenine yer verilsin. Bu durumda değişken aşağıdaki şekilde yazılacaktır:

dvar interval iscidinlenmemola [i in isciler] [g in gunler] in 0..96;

Bu şekildeki yazım ile ilgili karar değişkeninin 0 ile 96 arasında bir değer alması sağlanacaktır. Yani karar değişkeni alabileceği en erken başlama ile en geç bitme arasında bir değer almış olacaktır. Fakat mola vereceği zaman dilimi birden çok periyot uzunluğunda ise tanımlamaya boyut (size) anahtar sözcüğü eklenecektir. Örneğin yemek molasının 2 periyot sürdüğünü varsayalım. Bu durumdaki yazım şu şekilde olacaktır:

dvar interval isciyemekmola [i in isciler] [g in gunler] in 0..96 size 2;

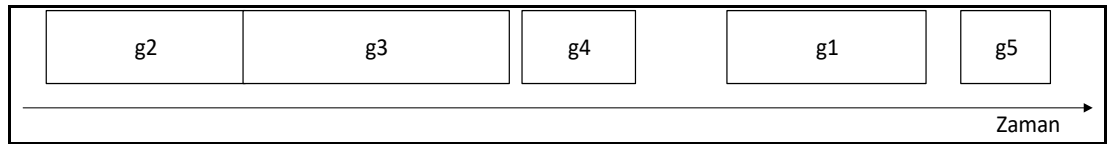
Yukarıdaki yazım düzenlerinde işçinin her gün çalıştığı ve çalıştığı gün mola verdiği varsayılmıştır. Fakat eğer işçiler her gün çalışmıyorsa opsiyonellik özelliği devreye girecektir. Yani ilgili karar değişkeni çözümde olabilir de olmayabilir de. Daha açık ifadeyle karar değişkeninin herhangi bir değer alması zorunlu değilse bu özellik kullanılmaktadır. Opsiyonellik “optional” söz diziminin kullanılması ile sağlanmaktadır. Aşağıda örnek bir kullanım şekli gösterilmiştir:

dvar interval isciyemekmola [i in isciler] [g in gunler] optional in 0..96 size 2;

Bir sıralama (sequence) karar değişkeni ise bir dizi aralık değişkenleri içerisindeki sıralamayı temsil eder. Bu karar değişkeni özellikle örtüşmemesi gereken karar değişkenlerini ifade etmede bazı fonksiyonların kullanılması ile ifade edilmektedir. Örneğin bir işçinin aynı anda sadece bir göreve atanabildiği modellerde karar değişkeni bu şekilde tanımlanabilir. Standart ve örnek bir gösterim aşağıdaki gibidir:

dvar sequence <dizi ismi> in <aralık ismi>;
dvar sequence sıra in all(t in tasks) gorev [t];

Yukarıdaki örnekte “gorev [t]” t. görevi temsil eden aralık karar değişkeni iken, “sıra” sıralama karar değişkenidir. Örnek olarak sıralama karar değişkeninin 5 tane {g1, g2, g3, g4, g5} aralık değişkeni üzerinde tanımlandığını ele alalım. Bu durumda sıralama karar değişkeni olan “sıra” çözümde örneğin $g2 \rightarrow g3 \rightarrow g4 \rightarrow g1 \rightarrow g5$ şeklinde olacaktır. Ortaya çıkan çözüm Şekil 3.5.’teki gibidir:



Şekil 3.5. Örnek bir sıralama karar değişkeni çözümü gösterimi.

Karar ifadeleri

Karar ifadeleri (decision expressions), modele ait var olan parametrelerin ve/veya karar değişkenlerinin kullanılması ile elde edilen değerleri gösterir. Daha açık

ifadeyle var olan parametrelerden/karar değişkenlerinden matematiksel ya da mantıksal yollarla üretilen değerlerdir denilebilir.

Karar ifadeleri CPLEX'te “dexpr” anahtar sözcüğü ile ifade edilir ve ihtiyaç duyulan CPLEX OPD fonksiyonlarının/anahtar sözcüklerinin kullanımı ile elde edilirler [90]. Bazı örnek gösterimler aşağıdaki gibidir:

dexpr int atamasayisi [g in gorevler]=sum (i in isciler) atama[g][i];

Yukarıdaki örnekte “atama[g][i]”, g. göreve i. işçinin atanıp atanmamasını belirleyen bir 0-1 karar değişkeni olduğu varsayılmıştır. Her bir göreve atanan toplam işçi sayısının hesaplanması istendiğinde yukarıda gösterildiği şekilde bir karar ifadesi oluşturulması gerekmektedir.

Karar ifadeleri özellikle CPLEX'te var olan fonksiyonların/anahtar sözcüklerin kullanılması ile elde edilir denilmiştir. Örneğin yukarıdaki örnekte toplama yapmayı sağlayan “sum” anahtar sözcüğü kullanılmıştır. CPLEX'te bunun gibi birçok fonksiyon/anahtar sözcük bulunmaktadır [90].

Tablo 3.1.'de modelde kullanılan bazı fonksiyonların/anahtar sözcüklerin tanımlarına alfabetik olarak yer verilmiştir. Tablo 3.1.'de verilen OPD sözdizimlerini kısaca açıklayacak olursak:

count

Bu sözdizimi iki argümana sahiptir. Bu argümanlardan ilki işlem yapılmak istenen karar değişkenini, diğeri ise kaç defa eşit olduğu istenen değeri ifade etmektedir. Standart bir gösterim şu şekildedir:

count (karar değişkeni, karar değişkeninin kaç defa eşit olduğunun sayılmak istendiği değer);

Tablo 3.1. Modelde kullanılan OPD sözdizimleri ve tanımları [89].

Sözdizimi	Tanım
<i>count</i>	Bir karar değişkeninin belirli bir değere kaç defa eşit olduğunu saymak için kullanılan sözdizimi.
<i>cumulFunction</i>	Sınırlı olan bir kaynağın zaman içerisindeki kullanım durumuna erişmek için kullanılan sözdizimi.
<i>lengthOf</i>	Bir aralık karar değişkeninin uzunluğuna erişmek için kullanılan sözdizimi.
<i>noOverlap</i>	Bir dizideki aralıkların çakışmasını önlemek için kısıt sözdizimi.
<i>presenceOf</i>	Aralık karar değişkeninin varlığını zorlamak için kullanılan sözdizimi.
<i>pulse</i>	Bir aralık değişkeninin kümülatif fonksiyona katkısını göstermek için kullanılan sözdizimi.
<i>synchronize</i>	Aralık karar değişkeninin başlangıç ve bitişini senkronize etmek için kullanılan kısıt sözdizimi.
<i>stepwise</i>	Basamaklı lineer fonksiyonlar için kullanılan sözdizimi.

Örnek bir gösterim de şu şekildedir:

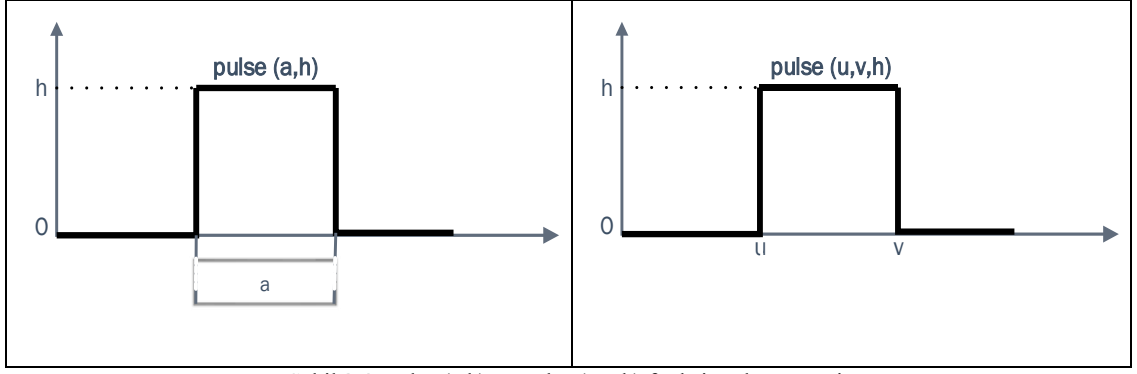
```
dexpr int toplamatanmasayisi [i in isciler] = count (all (g in görevler) atama[g][i], 1);
```

Burada “atama[g][i]” g. göreve i. işçinin atanıp atanmamasını belirleyen bir 0-1 karar değişkenidir. Eğer işçi bir göreve atanıyorsa bu durumda karar değişkeni 1 değerini, atanmıyorsa 0 değerini almaktadır. İlgili karar değişkeninin kaç defa 1 değerini aldığı bu şekilde hesaplanabilir.

cumulFunction, *pulse* ve *stepwise*

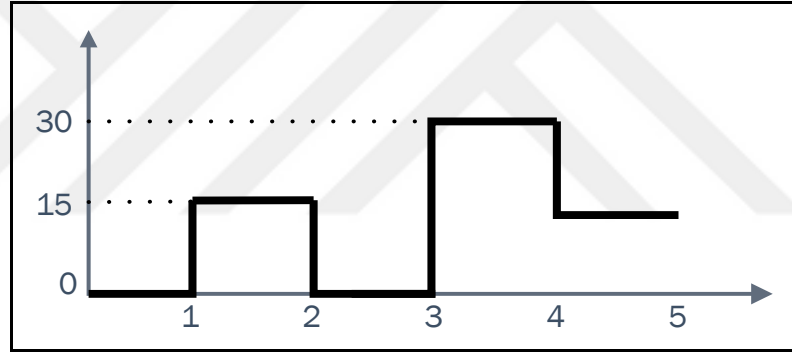
Bir kümülatif fonksiyon OPD’de *cumulFunction* anahtar kelimesi ile ifade edilmektedir. Şekil 3.6.’da gösterildiği üzere *a* aralık karar değişkeni ile ifade edildiğinde $f = pulse(a, h)$ şeklinde ve herhangi bir *h* parametresi ile ifade edildiğinde $f = pulse(u, v, h)$ şeklinde gösterilebilmektedir.

$f = pulse(a, h)$ fonksiyonunun değeri *a* aralık karar değişkeninin başlangıcı ve sonu arasında *h* değerindedir. $f = pulse(u, v, h)$ fonksiyonunun değeri ise *u* ile *v* arasında *h* değeri kadardır.



Şekil 3.6. pulse (a,h) ve pulse (u,v,h) fonksiyonları örneği.

Daha açık bir ifadeyle bir kümülatif fonksiyonun değeri Şekil 3.7.’de gösterildiği üzere sabit bir zaman veya bir aralık değişkeni ile ilişkili olarak azalan ya da artan eğilim gösteren bir basamak fonksiyonunu (step function) ifade etmektedir (stepwise{ 0→1, 15→2, 0→3, 30→4, 15→5}).



Şekil 3.7. Kümülatif fonksiyon örneği.

lengthOf

CPLEX’te bir aralık karar değişkeninin uzunluğuna lengthOf anahtar sözcüğü ile ulaşılır. Örnek bir gösterim şu şekildedir:

```
dexpr int toplamcalissuresi [i in isciler] = sum (g in gorevler) lengthOf (atama[i][g]);
```

Burada “atama[g][i]” karar değişkeninin g. göreve i. işçinin atanmasını belirleyen bir aralık karar değişkeni olduğu ele alınmıştır. Uzunluğunun da görevin süresi olduğu düşünülmüştür. Her bir işçinin planlama dönemindeki toplam çalışma süresinin hesaplanması istendiğinde bu şekilde bir karar ifadesi oluşturulması gerekmektedir.

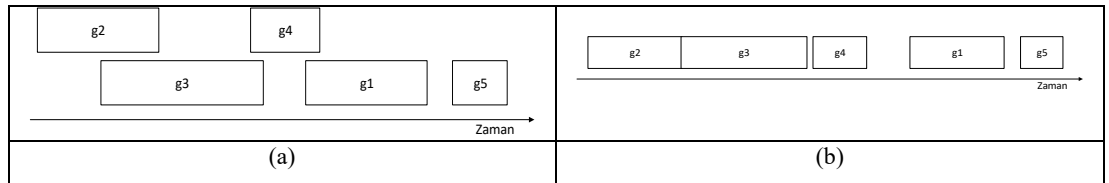
lengthOf anahtar sözcüğü ile planlama dönemi boyunca çalıştığı sürelerle (uzunluklara) ulaşılmaktadır.

noOverlap

Örtüşmesizlik kısıt sözdizimi olarak tanımlanabilir. Örtüşmesinin istenmediği görevleri (bir makinanın aynı anda sadece bir işlem yapması gibi) ifade etmek için sıralama karar değişkeni ile birlikte kullanılır. Burada sıralama karar değişkeninden bahsederken kullanılan örnek tekrar ele alınacak ve bu sözdiziminin nasıl kullanıldığı bir örnek üzerinde gösterilecek olursa:

```
dvar sequence sıra in all(t in tasks) gorev [t];
noOverlap (sıra);
```

Yukarıdaki örnekte “gorev [t]” t. görevi temsil eden aralık karar değişkeni iken, “sıra” sıralama karar değişkenidir. Burada noOverlap sözdizimi ile görevlerin örtüşmemesi/çakışmaması sağlanmış olmaktadır. Şekil 3.8.’de sırasıyla noOverlap kullanılmaması ve kullanılması durumlarındaki çözümler resmedilmiştir.



Şekil 3.8. noOverlap'sız ve noOverlap ile sıralama karar değişkeni çözüm örneği

Görüldüğü üzere noOverlap kullanılmaması durumunda (Şekil 3.8a) görevler zaman skalasında örtüşebilirken, noOverlap kullanılması durumunda (Şekil 3.8b) görevlerde zaman skalasında hiçbir şekilde örtüşme olmayacaktır.

presenceOf

Mantıksal bir sözdizimi olup, aslında 0-1 tam sayılı programlamaya benzer bir fonksiyon icra etmektedir. Eğer bir aralık karar değişkeninin çözümde olması

gerekiyorsa (1 durumu) ve olması gerekmiyorsa (0 durumu) durumlarını izah etmektedir. Örnek bir gösterim şu şekildedir:

```
dexpr int atamasayisi [g in gorevler] = sum (i in isciler) presenceOf (atama[g][i]);
```

Yukarıdaki örnekte “atama[g][i]” g. göreve i. işçinin atanıp atanmamasını belirleyen bir aralık karar değişkenidir. Her bir göreve atanan toplam işçi sayısının hesaplanması istendiğinde bu şekilde bir karar ifadesi oluşturulması gerekmektedir.

```
synchronize
```

Aynı anda başlamasının ve bitmesinin istendiği aralık karar değişkenleri için kullanılan kısıt sözdizimidir. Yani karar değişkenleri eğer çözümde yer alıyorsa bu sözdizimi sayesinde senkronize hareket etmiş olurlar. Örnek bir gösterim şu şekildedir:

```
forall (g in gorevler)
```

```
synchronize (gorev [g], all (p in paylasimlar: g.no == p.gorevno) paylasim [p]);
```

Yukarıdaki örnekte “gorev [g]”, g. görevi gösteren aralık karar değişkenidir. “paylasim [p]” ise hangi işin hangi işçi tarafından yapılacağını gösteren bir aralık karar değişkenidir. Modelde “gorev [g]” ile ilgili gerekli kısıtlar tanımlandıktan sonra bu şekilde bir sözdizimi kullanımı sayesinde “paylasim [p]” nin de aynı görev ile aynı anda başlayıp bitmesini kolaylıkla sağlayacaktır.

3.2.2.3. Kısıt programlama modeli (Model 2b)

KP tarafından modellenen herhangi bir çizelgeleme probleminde problemin temel yapı taşı görevlerdir [88]. Önerilen KP modelde, gün içi verilecek molaların her birinin bir görev olarak düşünülmesi ve bu görevlere işgücü kaynağının atanması fikrinden yola çıkılmıştır.

Bir önceki aşama olan vardiya çizelgeleme problemi çözümü ile hangi gün hangi vardiyaya kaç operatörün atanacağı belirlenmiş durumda olduğundan herhangi bir vardiyaya atanan operatörün kaç defa mola vermesi gerektiği bilirse planlama dönemi boyunca atanan toplam operatör sayısının o kadar katı kadar görev ortaya çıkmaktadır.

Örneğin; bir operatör, atandığı vardiyada 4 defa mola veriyorsa planlama dönemi boyunca atanan toplam operatör sayısının 4 katı kadar görev oluşacaktır. Önerilen KP modeli ile bu görevlerin hangi operatörler tarafından yapılacağı belirlenmeye çalışılmaktadır. Aslında bu şekildeki bir yaklaşım ile çizelgeleme probleminin boyutu önemli ölçüde azaltılmış olmaktadır.

Kümeler açısından; KP modelde bir önceki bölümde detayları verilen TSP modelinde izah edilen kümelerden “*s. vardiyadaki y tipi molanın zaman aralıkları*” kümesi hariç tüm kümeler kullanılmıştır. Parametreler açısından ise belirtilmesi gereken nokta TSP modelde saatlik çalışma maliyeti (C^H) dikkate alınırken KP modelde periyot başına çalışma maliyeti (C^P) dikkate alınmıştır. Sebebi, toplam maliyet hesabında aralık (interval) uzunluğunun kullanılmasıdır. Örnek verecek olursak 15 dakikalık periyot düzeni ele alınıyorsa bir saatlik çalışma süresi bu düzende 4 periyot içerdiğinden periyodik çalışma maliyeti $C^P = C^S/4$ olarak alınır.

Önerilen modele ait olup daha önce tanımlanmayan girdiler ve model yapısı aşağıdaki gibidir:

Kümeler

J	Görevler kümesi ($\forall j \in J = \{1, \dots, \tau Y \}$)
M	Görev boyutları kümesi ($\forall m \in M = \{1, \dots, M \}$)
R	Açılan vardiyalar kümesi ($\forall r \in R = \{1, \dots, R \}$)
O	Görev-kaynak paylaşımı tanımlama grubu (tuple) $O = \{< j, a > \forall j \in J, \forall a \in A\}$
S^A	Atama yapılan vardiyalar tanımlama grubu

	$(S^A = \{< r, d, s > \forall r \in R, \forall d \in D, \forall s \in S\})$
S^P	Vardiyaların başlama ve bitiş periyotları tanımlama grubu $(S^P = \{< s, i, e > \forall s \in S, \forall i \in P, \forall e \in P\})$
T	Görevler tanımlama grubu $(T = \{< j, d, y, s, m, i, e > \forall j \in J, \forall d \in D, \forall y \in Y, \forall s \in S, \forall m \in M, \forall i \in P, \forall e \in P\})$

Parametreler

C^P	Periyot başına işgücü maliyeti
τ	Planlama döneminde yapılan toplam atama sayısı
δ_{ys}	s. vardiyadaki y tipindeki molanın başlama periyodu
ε_{ys}	s. vardiyadaki y tipindeki molanın bitiş periyodu

Modelde öncelikle olası tüm görevlerin oluşturulması gerekmektedir. Herhangi bir görev modelde Şekil 3.9.'da da gösterildiği gibi 7'li tanımlama grubu düzeninde (T) gösterilmektedir.

Sıra (İndeks) No	Gün	Mola Tipi	Vardiya	Boyut	Başlama	Bitiş
---------------------	-----	-----------	---------	-------	---------	-------

Şekil 3.9. Görevlerin veri yapısı.

j , görev sıra (indeks) numarası olup d . gün y tipindeki görevin s . vardiyada i . periyotla e . periyot aralığında m büyüklüğünde gerçekleştirileceğini ifade etmektedir. Buradaki i . ve e . periyotlar ilgili tipteki görevin başlama ve bitiş periyodunu göstermekte olup yukarıda verilen δ_{ys} ve ε_{ys} parametrelerinden elde edilir.

Örneğin 2. vardiyadaki 1. tipteki bir görev bu vardiyadaki 1. mola zamanlarını ifade etmekte olup, başlangıç ve bitiş zamanı δ_{12} ve ε_{12} ile gösterilmektedir. Vardiyaların başlangıç ve bitiş periyotları (S^P) ile atama yapılan vardiyalar (S^A) 3'lü tanımlama grubu düzeninde tanımlanmıştır.

S^P tanımlama grubu elemanları s . vardiyanın i . periyotta başladığını e . periyotta sona erdiğini gösterirken, S^A tanımlama grubu elemanları i' sıra numarası olmak üzere d .

gün s . vardiyaya atama yapılmış olduğunu ifade eder. Görev kaynak paylaşımı için oluşturulan O tanımlama grubu da j indeks numaralı görevde a kaynağının kullanıldığını gösterir.

Karar değişkenleri

KP’de bir görevin gerçekleştirildiği zaman aralığı aralık (interval) karar değişkeni ile temsil edilebildiği daha önce izah edilmişti. Bunun yanı sıra başlangıçta bir görevin en erken başlama ve en geç bitme aralığındaki pozisyonu bilinmemektedir. Bu pozisyon sıralama değişkenleri (sequence variables) sayesinde belirlenmektedir. Ayrıca aralık değişkenleri farklı kısıtların kullanımı ile opsiyonel olarak da düşünülebilir.

Modelde kullanılan karar değişkenleri ve tanımları aşağıdaki gibidir:

x_{ka}	k . atama yapılan vardiyada a . operatörün görev almasını gösterir aralık karar değişkeni, i ile e periyotları arasında opsiyonel, $(e-i)$ boyutunda.	$\forall k \in S^A, \forall a \in A, \forall i \& e \in S^P$
w_t	Numarası t olan görevi gösterir aralık karar değişkeni, i ile e periyotları arasında opsiyonel, m boyutunda.	$\forall t \in T, (i, e \text{ ve } m \text{ } T\text{'nin bir elemanı})$
z_o	o . görev kaynak paylaşımını gösterir opsiyonel aralık karar değişkeni.	$\forall o \in O$
q_{ka}	z_o karar değişkenine ait sıralama karar değişkeni.	$\forall k \in S^A, \forall a \in A$

Kümülatif fonksiyonlar

Vardiyalara atanan operatörlerin sayısı ile molaya çıkan operatörlerin sayıları periyot periyot izlenerek aralarındaki farkın periyot başına ihtiyaç duyulan operatör sayısına eşit veya bu sayıdan daha fazla olmasının sağlanması gerektiği, bunun da zor bir kısıt olduğu daha önce vurgulanmıştır. Aynı periyotları kapsayan birden çok vardiya olması bu problemi daha da zorlaştırmaktadır.

Önerilen modelde bu işlev kümülatif fonksiyonlar sayesinde oldukça kolay bir şekilde gerçekleştirilmektedir. Diğer bir ifadeyle görevler ve bu görevlerin kaynak (operatör) kullanımı zamanın bir fonksiyonu olarak temsil edilmektedir.

Bu fonksiyon kullanılarak oluşturulan ve aşağıda Denklem 3.17, Denklem 3.18 ve Denklem 3.19 ile gösterilen denklemler ile sırasıyla vardiyalara atanan operatörlerin, molaya çıkan operatörlerin ve periyot başına ihtiyaç duyulan operatörlerin miktarları hesaplanmaktadır.

$$f_a^W = pulse(x_{ka}, 1) \quad \forall d \in D, \forall k \in S^A, \forall a \in A, k.d = d \quad (3.17)$$

$$f_a^B = pulse(w_t, 1) \quad \forall d \in D, \forall t \in T, t.d = d \quad (3.18)$$

$$f_a^R = pulse(p - 1, p, N_{dp}) \quad \forall d \in D, \forall p \in P \quad (3.19)$$

Kısıt programlama modeli

KP temelli modeller matematiksel modellerin aksine modelleme dillerinin fonksiyonel yapılarındaki farklılıklar nedeniyle kullanılan KP paketlerine büyük ölçüde bağımlıdır. Bu çalışmada modelleme dili olarak IBM ILOG CPLEX Optimization Studio 12.6 paket programındaki KP sözdizimleri kullanılmıştır. Modelde kullanılan sözdizimlerinin tanımlarına önceki bölümde yer verilmiştir.

Yukarıda verilen parametreler, karar değişkenleri ve sözdizimlerinin tanımları ışığında KP temelli model aşağıdaki gibi formüle edilmiştir:

Amaç fonksiyonu

$$\text{minimize} \sum_{\substack{k \in S^A, \\ a \in A}} (C^P \cdot lengthOf(x_{ka})) \quad (3.20)$$

Denklem 3.20 ile ifade edilen amaç fonksiyonu toplam işgücü maliyetini en küçükleyecek şekilde düzenlenmiştir. *lengthOf* ifadesi ile daha önce de belirtildiği

üzere a . operatörün k . atama yapılan vardiyadaki görevinin periyot olarak uzunluğu elde edilmektedir. Yine operatörler arasında herhangi bir farklılık bulunmamaktadır.

Kısıtlar

Durum (Mantıksal) kısıtlar

Modelde bazı kısıtlar `presenceOf` sözdizimi kullanılarak oluşturulmuştur. Bu kısıtlar şu şekildedir:

$$\sum_{a \in A} \text{presenceOf}(x_{ka}) = G_{ds} \quad \forall k \in S^A, \forall d \in D, \forall s \in S \quad (3.21)$$

$$\sum_{k \in S^A} \text{presenceOf}(x_{ka}) \leq 1 \quad \forall a \in A, \forall d \in D, k.d = d \quad (3.22)$$

$$\sum_{k \in S^A} \text{presenceOf}(x_{ka}) \leq (H - 1) \quad \forall a \in A \quad (3.23)$$

$$\sum_{\substack{t \in T, \\ o \in O}} \text{presenceOf}(z_o) = (\text{presenceOf}(x_{ka})) \quad \forall a \in A, \forall k \in S^A, \forall y \in Y, \\ t.j = o.j, o.a = a, t.s = k.s, \\ t.d = i.d, t.y = y \quad (3.24)$$

$$\sum_{o \in O} \text{presenceOf}(z_o) = \text{presenceOf}(w_t) \quad \forall t \in T, t.j = o.j \quad (3.25)$$

Denklem 3.21, ilgili gün ilgili vardiyaya atanacak operatör sayısını, vardiya çizelgeleme probleminin çözümü neticesinde elde edilen gün başına vardiyalardaki operatör sayısına eşitler. Denklem 3.22 ile operatörlerin planlama ufunda herhangi bir günde en fazla bir vardiyaya atanmalarına izin verilmektedir. Denklem 3.23, operatörlerin planlama ufunda en az 1 gün tatil yapmalarına olanak sağlamaktadır.

Denklem 3.24, operatörlerin atandıkları vardiyadaki her bir görevden (moladan) sadece bir tane yapmasını garantiler (Operatörlerin atandıkları vardiyada istenen tüm

molalarını vermeleri garantilenmiş olur). Denklem 3.25 sayesinde her bir görevin sadece bir operatör tarafından yapılmasını sağlanır.

Senkronizasyon formatı

Bir önceki alt bölümde ayrıntıları verilen senkronizasyon kısıt yapısı modelde Denklem 3.26'da gösterildiği gibi kullanılmıştır. Bu kısıt ile görev-kaynak paylaşımlarının (allocations) başlangıç ve bitiş zamanlaması yapılmış olur.

$$\text{synchronize}(w_t, \{z_1, \dots, z_o : o \in O\}) \quad \forall t \in T, t.j = o.j \quad (3.26)$$

Örtüşme önleme

Modelde Denklem 3.27'deki gibi `noOverlap` kısıtı kullanılarak bir operatörün görev aldığı (molada olduğu) zaman aralığında aynı zaman dilimine gelebilecek başka bir görevde (molada) kullanılması engellenmiş olur.

$$\text{noOverlap}(q_{ka}) \quad \forall a \in A, \forall k \in S^A \quad (3.27)$$

Kümülatif fonksiyonlar

Planlama dönemindeki günlerin her gün içi periyodunda olması gereken kadar operatörün çalıştırılmasının garantilenmesi gerekmektedir. Bu gereklilik modelde Denklem 3.28'deki kısıt ile sağlanmaktadır. Bu kısıt sayesinde vardiyaya atanan operatörlerin toplam sayısı ile molaya çıkan operatörlerin toplam sayısı arasındaki farkın o anlık istenen talebe eşit veya büyük olması sağlanmış olur.

$$f_d^W - f_d^B \geq f_d^R \quad \forall d \in D \quad (3.28)$$

Alan kısıtları

Modele ait alan kısıtları Denklem 3.29'da gösterildiği gibidir.

$$G_{ds}, \delta_{ys}, \varepsilon_{ys}, \tau \geq 0 \text{ ve tamsayı} \quad \forall d \in D, \forall s \in S, \forall y \in Y \quad (3.29)$$

Modele ait CPLEX kodlarına Ek-C’de yer verilmiştir.

3.3. Kıyaslama Test Örnekleri Modelleri

Önceki bölümlerde geliştirilen modelin doğrulanması adına kıyaslama test örneklerinin kullandığı belirtilmişti. Bu bölümde öncelikle bahsi geçen bu test örneklerine, ilgili matematiksel modele ve daha sonra bu matematiksel modele yönelik geliştirilen KP temelli modele değinilecektir.

3.3.1. Kıyaslama test örneklerinde kullanılan matematiksel model (Model 3a)

Curtois ve Qu [9] çalışmalarında personel çizelgeleme problemi için geliştirilen bazı modellerin kıyaslama test örnekleri ([8]) üzerindeki performansını ortaya koymuşlardır.

Çalışmalarında yer alan ve test örneklerinde kullanılan matematiksel modele aşağıda yer verilmiştir:

Parametreler

I	Çalışanlar kümesi
h	Planlama dönemindeki toplam gün sayısı
D	Planlama dönemindeki günler kümesi ($= \{1, \dots, h\}$)
W	Planlama dönemindeki haftalar kümesi ($= \{1, \dots, h/7\}$)
T	Vardiya tipleri kümesi
R_t	t tipindeki vardiyadan sonra atanılamayacak vardiya kümesi
N_i	i . çalışanın atanılmayacağı günler kümesi
l_t	t tipindeki vardiyanın uzunluğu (dakika)
m_{it}^{max}	i . çalışanın atanabileceği t tipindeki maksimum vardiya sayısı

b_i^{min}	<i>i.</i> çalışanın çalışması gereken toplam minimum süre (dakika)
b_i^{max}	<i>i.</i> çalışanın çalışması gereken toplam maksimum süre (dakika)
c_i^{min}	<i>i.</i> çalışanın art arda çalışabileceği minimum vardiya sayısı
c_i^{max}	<i>i.</i> çalışanın art arda çalışabileceği maksimum vardiya sayısı
o_i^{min}	<i>i.</i> çalışanın art arda atanabileceği minimum izin gün sayısı
a_i^{max}	<i>i.</i> çalışanın art arda atanabileceği maksimum hafta sonu sayısı
q_{idt}	<i>i.</i> çalışanın <i>d.</i> gün <i>t</i> tipindeki vardiyaya atanmak isteyipte atanmadığında oluşan penaltı değeri
p_{idt}	<i>i.</i> çalışanın <i>d.</i> gün <i>t</i> tipindeki vardiyaya atanmamak isteyipte atandığında oluşan penaltı değeri
u_{dt}	<i>d.</i> gün <i>t</i> tipindeki vardiyaya atanması istenen toplam çalışan sayısı
v_{dt}^{min}	<i>d.</i> gün <i>t</i> tipindeki vardiyaya atanması istenenden az sayıda çalışan atanması durumunda kullanılacak ağırlık değeri
v_{dt}^{max}	<i>d.</i> gün <i>t</i> tipindeki vardiyaya atanması istenenden çok sayıda çalışan atanması durumunda kullanılacak ağırlık değeri

Karar değişkenleri

x_{idt}	<i>i.</i> çalışan <i>d.</i> gün <i>t</i> tipindeki vardiyaya atandığında 1, diğer durumda 0.
k_{iw}	<i>i.</i> çalışan <i>w.</i> hafta sonuna atandığında 1, diğer durumda 0.
y_{dt}	<i>d.</i> gün <i>t</i> tipindeki vardiyaya atanması istenen toplam çalışan sayısından az olan değer
z_{dt}	<i>d.</i> gün <i>t</i> tipindeki vardiyaya atanması istenen toplam çalışan sayısından fazla olan değer

Amaç Fonksiyonu

$$\min \sum_{i \in I} \sum_{d \in D} \sum_{t \in T} q_{idt}(1 - x_{idt}) + \sum_{i \in I} \sum_{d \in D} \sum_{t \in T} p_{idt} x_{idt} + \sum_{d \in D} \sum_{t \in T} y_{dt} v_{dt}^{min} + \sum_{d \in D} \sum_{t \in T} z_{dt} v_{dt}^{max} \quad (3.30)$$

Denklem 3.30'da gösterilen amaç fonksiyonu çalışanların talep ettikleri vardiyaya atanma seviyesini en üst düzeyde tutmayı ve her bir günün her bir periyodunda

gerekli personel ihtiyacının üstünde veya altında personel çalışmasını en asgari düzeyde tutmayı amaçlamaktadır.

Kısıtlar

$$\sum_{d \in D} x_{idt} \leq 1 \quad \forall i \in I, d \in D \quad (3.31)$$

$$x_{idt} + x_{i(d+1)u} \leq 1 \quad \forall i \in I, d \in \{1 \dots h-1\}, \quad (3.32)$$

$$t \in T, u \in R_t$$

$$\sum_{t \in T} x_{idt} \leq m_{it}^{max} \quad \forall i \in I, t \in T \quad (3.33)$$

$$b_i^{min} \leq \sum_{d \in D} \sum_{t \in T} l_t x_{idt} \leq b_i^{max} \quad \forall i \in I \quad (3.34)$$

$$\sum_{j=d}^{d+c_i^{max}} \sum_{t \in T} x_{ijt} \leq c_i^{max} \quad \forall i \in I, d \in \{1 \dots h - c_i^{max}\} \quad (3.35)$$

$$\sum_{t \in T} x_{idt} + \left(s - \sum_{j=d+1}^{d+s} \sum_{t \in T} x_{ijt} \right) + \sum_{t \in T} x_{i(d+s+1)t} > 0 \quad \forall i \in I, s \in \{1 \dots c_i^{min} - 1\}, \quad (3.36)$$

$$d \in \{1 \dots h - (s+1)\}$$

$$\left(1 - \sum_{t \in T} x_{idt} \right) + \sum_{j=d+1}^{d+s} \sum_{t \in T} x_{ijt} + \left(1 - \sum_{t \in T} x_{i(d+s+1)t} \right) > 0 \quad \forall i \in I, s \in \{1 \dots o_i^{min} - 1\}, \quad (3.37)$$

$$d \in \{1 \dots h - (s+1)\}$$

$$k_{iw} \leq \sum_{t \in T} x_{i(7w-1)t} + \sum_{t \in T} x_{i(7w)t} \leq 2k_{iw} \quad \forall i \in I, w \in W \quad (3.38)$$

$$\sum_{w \in W} k_{iw} \leq a_i^{max} \quad \forall i \in I \quad (3.39)$$

$$x_{idt} = 0 \quad \forall i \in I, d \in N_i, t \in T \quad (3.40)$$

$$\sum_{i \in I} x_{idt} - z_{dt} + y_{dt} = u_{dt} \quad \forall d \in D, t \in T \quad (3.41)$$

Model, 10 adet kısıttan oluşmaktadır ve kısıt açıklamaları şu şekildedir:

Denklem 3.31 ile gösterilen kısıt ile bir çalışan bir günde en fazla bir vardiyaya atanabilir. Denklem 3.32’de gösterilen kısıt vardiya rotasyonunu ifade etmektedir.

Yani belirli bir gün belirli bir vardiyada çalışan işçi ertesi gün önceden belirlenmiş olan bazı vardiyalarda çalışamaz.

Denklem 3.33, çalışanların planlama dönemi boyunca atanabilecekleri toplam vardiya sayısını sınırlayan kısıtı ifade etmektedir. Çalışanlar planlama dönemi boyunca tanımlı olan minimum ve maksimum çalışma süreleri kapsamında çalışmaktadırlar. Bu kısıt da Denklem 3.34'deki şekliyle modellenmiştir.

Denklem 3.35'deki kısıt maksimum sıralı çalışma gün sayısını ifade etmektedir. Yani çalışanlar ara vermeksizin en fazla belirli gün peş peşe çalışabilir. Örneğin c_i^{max} değerinin $i=12$ için 3 olduğunu düşünölsün ($c_{12}^{max} = 3$). Bu değeri 12 numaralı işçinin art arda en fazla 3 gün çalışacağını ifade etmektedir.

Denklem 3.36'de gösterilen kısıt ise minimum sıralı çalışma gün sayısını göstermektedir. Bu kısıt ile çalışanlar ara vermeksizin en az belirli gün peş peşe çalışabilir. Örneğin c_i^{min} değerinin $i=15$ için 2 olduğunu düşünölsün ($c_{15}^{min} = 2$). Bu durumda 15 numaralı çalışan ara vermeksizin en az 2 gün çalışacaktır.

Çalışanların en az belirli gün art arda izin yapmaları istenmektedir. Bu kısıt da Denklem 3.37'de gösterildiğı gibi minimum sıralı izin gün sayısını sınırlandırmaktadır. Örneğin o_i^{min} değerinin $i=18$ için 2 olduğunu düşünelim ($o_{18}^{min} = 2$). Bu değeri 18 numaralı çalışanın ara vermeksizin en az 2 gün peş peşe izinde olacağını (çalışmayacağını) ifade etmektedir.

Çalışanlar en fazla belirli sayıda hafta sonu günü çalışabilir. Hafta sonu olarak Cumartesi ve Pazar günleri düşünölmüştür. Bu kısıtlar Denklem 3.38 ve Denklem 3.39'da gösterildiğı şekilde yazılmaktadır.

Denklem 3.40, izin günleri kısıtını ifade etmektedir. Bu kısıt çalışanların önceden belirlenen izin günlerinde çalışmamalarını sağlamaktadır. Her bir günün her bir vardiyasındaki çalışan ihtiyacını karşılama kısıtı ise Denklem 3.41'de gösterildiğı gibidir.

3.3.2. Kıyaslama test örnekleri için geliştirilen KP temelli model (Model 3b)

Kıyaslama test örneklerini kullanabilmek için modelin ilgili çalışmada ([9]) belirtilen formatta düzenlenmesi gerekmiştir. Daha açık ifadeyle KP yaklaşım tarzı değiştirilmeksizin modelin yeniden yazılması gerekmiştir. KP yaklaşımı ancak bu şekilde literatürdeki bilinen sonuçlarla karşılaştırabilirdi. Yeni duruma göre düzenlenmiş model ve detayları aşağıdaki gibidir:

Kümeler

J	Görevler kümesi ($\forall j \in J = \{1, \dots, \tau T \}$)
M	Görev uzunlukları kümesi ($\forall m \in M = \{1, \dots, M \}$)
P	Bir çalışma günündeki periyotlar kümesi ($\forall p \in P = \{1, \dots, P \}$)
O	Görev paylaşımları tanımlama grubu $O = \{< j, i > \forall j \in J, \forall i \in I\}$
T^P	Vardiyaların başlama ve bitiş zamanları $(T^P = \{< t, c, e > \forall t \in T, \forall c \& e \in P\})$
G	Görevler tanımlama grubu $(G = \{< j, d, t, m, c, e > \forall j \in J, \forall d \in D, \forall t \in T, \forall m \in M, \forall c \& e \in P\})$

Parametreler

u_g^{KP}	g. göreve atanması istenen toplam çalışan sayısı
v_g^{KPmin}	g. göreve atanması istenenden az sayıda çalışan atanması durumunda kullanılacak ağırlık değeri
v_g^{KPmax}	g. göreve atanması istenenden çok sayıda çalışan atanması durumunda kullanılacak ağırlık değeri
τ	Planlama dönemindeki toplam atama sayısı

Karar değişkenleri

x_{gi}^{KP}	i . çalışan g. göreve atandığında 1, diğer durumda 0.
---------------	---

k_{iw}^{KP}	i . çalışan w . hafta sonuna atandığında 1, aksi halde 0.	
y_g^{KP}	g . göreve atanması istenen toplam çalışan sayısından az olan değer.	
z_g^{KP}	g . göreve atanması istenen toplam çalışan sayısından fazla olan değer.	
w_g	Numarası g olan görevi gösterir aralık karar değişkeni, $\forall g \in G$, (c , e ve m ile e periyotları arasında opsiyonel, m boyutunda. G 'nin bir elemanı)	
z_o	o . görev kaynak paylaşımını gösterir opsiyonel aralık karar değişkeni.	$\forall o \in O$
q_{ig}^{KP}	z_o karar değişkenine ait sıralama karar değişkeni.	$\forall i \in I, \forall g \in G$

Görüldüğü üzere karar değişkenleri Model 3a'dan farklı olarak görev (g)-kaynak (i) temelli tanımlanmıştır. Model 2b'de görev olarak molalar, kaynak olarak çalışanlar (mola-işçi) ele alınmıştı. Model 3b'de ise görev olarak vardiyalar, kaynak olarak yine çalışanlar (vardiya-işçi) düşünülmüştür. Yani her bir vardiya görev olarak düşünülüp bu görevlerde hangi kaynakların kullanılacağına belirlenmesi geliştirilen modelin ana düşüncesidir.

Modelde kullanılan karar ifadeleri ve kısıtlara geçmeden önce bir hususu açıklamakta fayda görülmektedir. KP temelli modelin geliştirilmesi esnasında her bir kısıt farklı OPD ifadeleri/formülasyonları (count, cumulfFunction, presenceOf, sum) ile tek tek yazılmıştır. Bu şekilde yazılan her bir kısıtın sadece bir yazımı her bir çalıştırmada aktif halde, diğerleri pasif durumda bırakılmıştır. Böylelikle hangi OPD ifadesinin kullanımının daha etkin sonuçlar ortaya konulduğu test edilmiş ve sadece etkin sonuçlar veren kısıt yapısı tüm örnekler kapsamında kullanılmıştır. İleriki sayfalarda ilgili kısıtın sadece en etkin sonucu veren kısıt yapısına yer verilmiştir.

Kümülatif fonksiyonlar

Modeldeki bazı kısıtlar kümülatif fonksiyonlar kullanılarak yazılmıştır. Kısıtlarda kullanılacak olan kümülatif fonksiyonların gösterimlerine aşağıda yer verilmiştir.

$$f_{id}^A = pulse(z_o, 1) \quad \forall d \in D, \forall i \in I, \forall o \in O, \forall g \in G, \quad (3.42)$$

$$g.d = d, o.i = i, o.j = g.j$$

$$f_{idt}^R = pulse(z_o, 1) \quad \forall d \in D, \forall i \in I, \forall t \in T, \forall o \in O, \quad (3.43)$$

$$\forall g \in G, g.d = d, o.i = i, o.j = g.j,$$

$$g.t = t$$

$$f_{it}^M = pulse(z_o, 1) \quad \forall i \in I, \forall t \in T, \forall o \in O, \forall g \in G, \quad (3.44)$$

$$g.t = t, o.i = i, o.j = g.j$$

f_{id}^A , i . çalışanın d . gün atandığı vardiya sayısını hesaplamaktadır. Hesaplamayı görev-kaynak paylaşımı karar değişkeni üzerinden yapmaktadır. Bu değer çalışanların gün içinde atanacağı vardiya sayısını kısıtlamada kullanılacaktır. f_{idt}^R ise i . çalışanın d . gün t . vardiyaya atanma sayısını kümülatif olarak belirlemektedir. Bu değer de vardiya rotasyonu kısıtında kullanılacaktır. Problemden çalışanların planlama dönemi boyunca atanabilecekleri toplam vardiya sayısı sınırlı idi. Modelde bu hesaplama f_{it}^M ile yapılmaktadır.

Sayma formülasyonları

Kıyaslama test örneklerini modellemedeki KP yaklaşımında ilgili kısıtları daha etkin modellemek için genel KP modelinden farklı olarak sayma (count) formülasyonu da kullanılmıştır. Modelde kullanılan sayma formülasyonları aşağıda gösterilmiştir.

$$SAY_i^{Cmak} \quad d \in \{1 \dots h - c_i^{max}\}, \quad (3.45)$$

$$= count \left(all \left(\begin{matrix} n \\ g \text{ in } G \end{matrix} in(d..(d + c_i^{max})), x_{gi}^{KP}, 1 \right) \right) \quad \forall i \in I, \forall g \in G, \\ g.d = n$$

$$SAY_i^{Cmin1} = count(all(g \text{ in } G) x_{gi}^{KP}, 1) \quad \forall i \in I, \forall g \in G, g.d = d \quad (3.46)$$

$$s \in \{1 \dots c_i^{min} - 1\},$$

$$d \in \{1 \dots h - (s + 1)\}$$

$$SAY_i^{Cmin2} = count(all(g \text{ in } G) x_{gi}^{KP}, 1) \quad \forall i \in I, \forall g \in G, \quad (3.47)$$

$$g.d = (d + s + 1)$$

$$s \in \{1 \dots c_i^{min} - 1\},$$

$$\begin{aligned}
& d \in \{1 \dots h - (s + 1)\} \\
& \forall i \in I, \forall g \in G, g.d = n \quad (3.48) \\
& SAY_i^{Cmin3} = count \left(all \left(n \text{ in } ((d + 1) \dots (d + s)), \right)_{g \text{ in } G} x_{gi}^{KP}, 1 \right) \quad s \in \{1 \dots c_i^{min} - 1\}, \\
& \quad \quad \quad d \in \{1 \dots h - (s + 1)\}
\end{aligned}$$

$$\begin{aligned}
& SAY_i^{Omin1} = count(all(g \text{ in } G)x_{gi}^{KP}, 1) \quad \forall i \in I, \forall g \in G, g.d = d \quad (3.49) \\
& \quad \quad \quad s \in \{1 \dots o_i^{min} - 1\}, \\
& \quad \quad \quad d \in \{1 \dots h - (s + 1)\}
\end{aligned}$$

$$\begin{aligned}
& SAY_i^{Omin2} = count(all(g \text{ in } G)x_{gi}^{KP}, 1) \quad \forall i \in I, \forall g \in G, \quad (3.50) \\
& \quad \quad \quad g.d = (d + s + 1) \\
& \quad \quad \quad s \in \{1 \dots o_i^{min} - 1\}, \\
& \quad \quad \quad d \in \{1 \dots h - (s + 1)\}
\end{aligned}$$

$$\begin{aligned}
& SAY_i^{Omin3} = count \left(all \left(n \text{ in } ((d + 1) \dots (d + s)), \right)_{g \text{ in } G} x_{gi}^{KP}, 1 \right) \quad \forall i \in I, \forall g \in G, g.d = n \quad (3.51) \\
& \quad \quad \quad s \in \{1 \dots o_i^{min} - 1\}, \\
& \quad \quad \quad d \in \{1 \dots h - (s + 1)\}
\end{aligned}$$

Denklem 3.45 ile hesaplanan değer (SAY_i^{Cmak}) çalışanların ara vermeksizin en fazla belirli gün peş peşe çalışabilmesini kısıtlamada kullanılacaktır. Bu formülasyon her bir çalışanın art arda olan çalışma gün sayılarını toplamaktadır. Denklem 3.46 (SAY_i^{Cmin1}), Denklem 3.47 (SAY_i^{Cmin2}) ve Denklem 3.48 (SAY_i^{Cmin3}) ile hesaplanan değerler çalışanların ara vermeksizin en az belirli gün peş peşe çalışabilmesini kısıtlamada kullanılacaktır. Çalışanların ara vermeksizin en az belirli gün art arda izin yapmalarını sağlayacak kısıtta Denklem 3.49 (SAY_i^{Omin1}), Denklem 3.50 (SAY_i^{Omin2}) ve Denklem 3.51 (SAY_i^{Omin3}) ile bulunan değerler kullanılacaktır.

Amaç Fonksiyonu

$$\begin{aligned}
& \min \sum_{\substack{t \in I \\ g \in G \\ d \in D \\ t \in T}} q_{idt}(1 - x_{gi}^{KP}) + \sum_{\substack{t \in I \\ g \in G \\ d \in D \\ t \in T}} p_{idt}x_{gi}^{KP} + \sum_{g \in G} y_g^{KP} v_g^{KPmin} + \sum_{g \in G} z_g^{KP} v_g^{KPmax} \quad g.t = t, \quad (3.52) \\
& \quad \quad \quad g.d = d
\end{aligned}$$

Denklem 3.52’de gösterilen amaç fonksiyonu çalışanların talep ettikleri vardiyaya atanma seviyesini en üst düzeyde tutmayı ve her bir günün her bir periyodunda gerekli personel ihtiyacının üstünde veya altında personel çalışmasını en asgari düzeyde tutmayı amaçlamaktadır.

Kısıtlar

Model 3a’daki 1 numaralı kısıt (Bir çalışanın bir günde birden fazla vardiya atanamaması) KP temelli modelde aşağıdaki gibi yazılmıştır (Denklem 3.53):

$$f_{id}^A \leq 1 \quad \forall d \in D, \forall i \in I \quad (3.53)$$

Belirli bir gün belirli bir vardiyada çalışanın ertesi gün belirtilen bazı vardiyalarda çalışamamasının sağlandığı 2 numaralı kısıt Denklem 3.54 ile gösterildiği gibidir:

$$f_{idt}^R + f_{i(d+1)u}^R \leq 1 \quad \forall i \in I, d \in \{1 \dots h-1\}, \quad (3.54)$$

$$t \in T, u \in R_t$$

Çalışanların planlama dönemi boyunca atanabilecekleri toplam vardiya sayısı sınırlı idi. Bunu sağlayan 3. kısıt modelde Denklem 3.55’deki şekliyle yer almıştır.

$$f_{it}^M \leq m_{it}^{max} \quad \forall i \in I, t \in T \quad (3.55)$$

Model 3a’daki 4 numaralı kısıt (Çalışanların planlama dönemi boyunca tanımlı olan minimum ve maksimum çalışma süreleri kapsamında çalışabilmesi) KP modelinde aşağıdaki gibi yazılmıştır (Denklem 3.56):

$$b_i^{min} \leq \sum_{g \in G} x_{gi}^{KP} * 60 * lengthOf(w_g) \leq b_i^{max} \quad \forall i \in I \quad (3.56)$$

Çalışanların ara vermeksizin en fazla belirli gün peş peşe çalışabilmesini sağlayan 5 numaralı kısıt KP modelinde Denklem 3.57 ile gösterildiği gibidir.

$$SAY_i^{Cmak} \leq c_i^{max} \quad \forall i \in I \quad (3.57)$$

Çalışanlar ara vermeksizin en az belirli gün peş peşe çalışabiliyordu. Bunu sağlayan 6 numaralı kısıt ise KP modelde Denklem 3.58'deki şekliyle yer almıştır.

$$SAY_i^{Cmin1} + SAY_i^{Cmin2} + (s - SAY_i^{Cmin3}) > 0 \quad \forall i \in I, \quad (3.58)$$

$$s \in \{1 \dots c_i^{min} - 1\}$$

Çalışanların art arda en az belirli gün izin yapmaları ile ilgili 7 numaralı kısıt KP modelinde aşağıdaki gibi yazılmıştır (Denklem 3.59):

$$(1 - SAY_i^{Omin1}) + (1 - SAY_i^{Omin2}) + SAY_i^{Omin3} > 0 \quad \forall i \in I, \quad (3.59)$$

$$s \in \{1 \dots o_i^{min} - 1\}$$

Çalışanların en fazla belirli sayıda hafta sonu günleri çalışabilmesini sağlayan 8 numaralı kısıt KP modelinde Denklem 3.60 ve Denklem 3.61 ile gösterildiği gibidir.

$$k_{iw}^{KP} \leq \sum_{t \in T} x_{g_1 i}^{KP} + \sum_{t \in T} x_{g_2 i}^{KP} \leq 2k_{iw}^{KP} \quad \forall i \in I, w \in W, \quad (3.60)$$

$$\forall g \in G, g_1.d = 7w,$$

$$g_2.d = 7w - 1$$

$$\sum_{w \in W} k_{iw}^{KP} \leq a_i^{max} \quad \forall i \in I \quad (3.61)$$

Çalışanların verilen izin günlerinde çalışmamaları gerekmektedir. Bunu sağlayan 9 numaralı kısıt ise KP modelinde Denklem 3.62'deki şekliyle yer almıştır.

$$\sum_{\substack{g \in G, \\ o \in O}} presenceOf(z_o) = 0 \quad \forall i \in I, d \in N_i, \quad (3.62)$$

$$g.d = d,$$

$$o.i = i, o.j = g.j$$

Model 3a'daki 10 numaralı kısıt (Her bir vardiyadaki ihtiyacı karşılama kısıtı) KP temelli modelde aşağıdaki gibi yazılmıştır (Denklem 3.63):

$$\sum_{\substack{i \in I, \\ o \in O}} presenceOf(z_o) - z_g^{KP} + y_g^{KP} = u_g^{KP} \quad \forall g \in G, o.i = i, \quad (3.63)$$

$$g.j = o.j$$

Bu kısıtlara ek olarak Model 3b’de Model 2b’deki gibi senkronizasyon formülasyonu (*synchronize*) ve örtüşme önleme (*noOverlap*) kısıtları da kullanılmıştır. Denklem 3.64’de gösterilen kısıt ile görev-kaynak paylaşımlarının (*allocations*) başlangıç ve bitiş zamanlaması yapılmış olur. Denklem 3.65’deki gibi *noOverlap* kısıtı kullanılarak da bir operatörün görev aldığı (vardiyada olduğu) zaman aralığında aynı zaman dilimine gelebilecek başka bir görevde kullanılması engellenmiş olur.

$$synchronize(w_g, \{z_1, \dots, z_o : o \in O\}) \quad \forall g \in G, g.j = o.j \quad (3.64)$$

$$noOverlap(q_{ig}^{KP}) \quad \forall i \in I, g \in G \quad (3.65)$$

Modele ait CPLEX kodlarına Ek-D’de yer verilmiştir. Model 3b’deki kısıtların genel yapısı Tablo 3.2.’de özet olarak gösterilmiştir. Bu modele ait bulgulara sonraki bölümlerde yer verilecektir.

Tablo 3.2. Kıyaslama için geliştirilen KP modelinin özet kısıt yapısı

[9]’deki Kısıtlar	Kısıt Tanımları	Model 3b’deki Kısıt Yapısı
1	Bir çalışanın bir günde birden fazla vardiya atanamaması	<i>Kümülatif fonksiyon</i>
2	Her vardiyadan sonra en az belirli bir miktar çalışılmaması	<i>Kümülatif fonksiyon</i>
3	Çalışanların atanabileceği tür olarak en çok vardiya sayısı	<i>Kümülatif fonksiyon</i>
4	Planlama döneminde minimum ve maksimum çalışma süresi	<i>Durum (Mantıksal) kısıtları</i>
5	İzin günü olmaksızın art arda maksimum atanma sayısı	<i>Sayma (Count) fonksiyonu</i>
6	İzin günü olmaksızın art arda minimum atanma sayısı	<i>Sayma (Count) fonksiyonu</i>
7	Art arda minimum ardışık izin günü sayısı	<i>Sayma (Count) fonksiyonu</i>
8	Maksimum hafta sonu çalışma gün sayısı	<i>Durum (Mantıksal) kısıtları</i>
9	Önceden belirlenen izin günlerinde atanmaması	<i>Durum (Mantıksal) kısıtları</i>
10	Vardiyalarda ihtiyaç duyulan çalışan sayısının dengelenmesi	<i>Durum (Mantıksal) kısıtları</i>

BÖLÜM 4. UYGULAMA ÇALIŞMASI

Bu bölümde, geliştirilen modellerin öncelikle halihazırda faaliyet gösteren bir firmadan elde edilen gerçek veriler üzerindeki davranışları incelenmiştir. Daha sonra problem büyüklüğüne göre oluşturulan test örnekleri ve standart kıyaslama test örnekleri kullanılarak model davranışları ortaya konulmuştur.

4.1. Gerçek Yaşam Problemi

Geliştirilen modellerin davranışlarını incelemek için birden çok firmaya dış kaynak sağlayan bir çağrı merkezi firmasından elde edilen gerçek veriler kullanılmıştır. Bu aşamada öncelikle problem detayları verilecek olup daha sonra uygulama ve hesaplama sonuçlarına değinilecektir.

4.1.1. Problem tanımı

Geliştirilen modellerin vardiya ve personel çizelgeleme problemlerine yönelik çözümler üretebildiğini göstermek için öncelikle gerçek veriler kullanılmıştır. Verilerin alındığı firmadan vardiya bilgileri ve yakın bir geçmişe ait periyot başına gerekli minimum operatör sayısı bilgileri alınmıştır.

İlgili firmada personel ihtiyacının tahmini her gün için 15 dakikalık periyotlarla yapılmaktadır. Bu durumda her bir gün için 96 periyot ($|P|=96$) olmaktadır (24 saat*1 saatte 4 tane 15 dakikalık periyot=96 periyot). Planlama da haftalık (7 gün) olarak yapılmaktadır ($|D|=7$).

Firmadaki vardiyalara ait temel bilgiler Tablo 4.1.'de gösterilmiştir.

Tablo 4.1. Firmaya ait vardiya şablonları

Vardiya Numarası (s)	Saat Aralığı	Durumu	Vardiya		1 Tipindeki	2 Tipindeki	3 Tipindeki	4 Tipindeki
			Periyotları		Molanın	Molanın	Molanın	Molanın
			Baş.	Bit.	Periyot	Periyot	Periyot	Periyot
			(i)	(e)	Aralığı	Aralığı	Aralığı	Aralığı
1	00-08	✓	1	32	3-8	11-17	21-26	27-32
2	08-16	✓	33	64	35-40	43-49	53-58	59-64
3	09-17	☑	37	68	39-44	47-53	57-62	63-68
4	10-18	✓	41	72	43-48	51-57	61-66	67-72
5	11-19	☑	45	76	47-52	55-61	65-70	71-76
6	12-20	✓	49	80	51-56	59-65	69-74	75-80
7	13-21	☑	53	84	55-60	63-69	73-78	79-84
8	14-22	☑	57	88	59-64	67-73	77-82	83-88
9	15-23	☑	61	92	63-68	71-77	81-86	87-92
10	16-00	✓	65	96	67-72	75-81	85-90	91-96

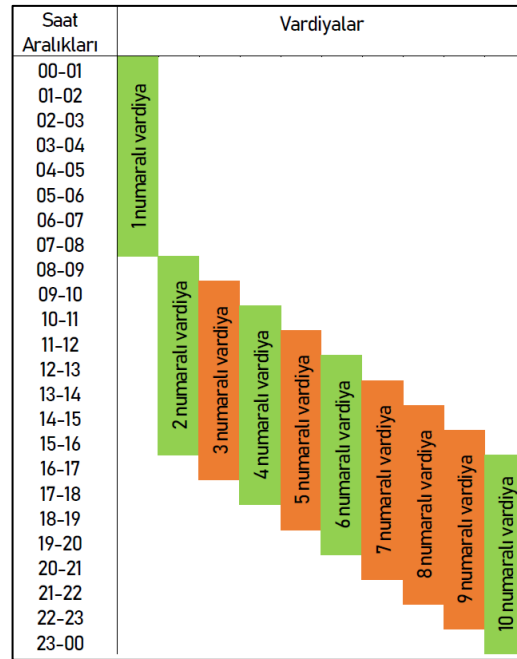
✓-Varolan vardiyalar, ☑-Analize dâhil edilen vardiyalar, Baş.=Başlangıç, Bit.=Bitiş,

$S^P = \{ \langle s, i, e \rangle \mid s \in S, i \& e \in P \}$.

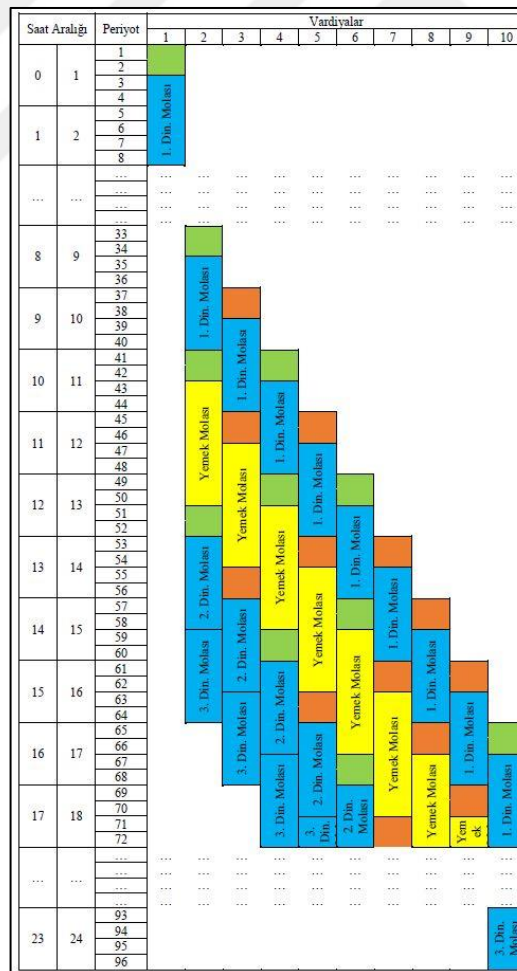
Firmada hafta içi yani Pazartesi ($d = 1$) ile Cuma ($d = 5$) günleri arasında 5 vardiya ($s = 1, 2, 4, 6, 10$), hafta sonu ($d = 6, 7$) ise 3 vardiya ($s = 1, 2, 10$) düzeninde çalışılmaktadır. Modele bunlara ek olarak 5 vardiya daha dahil edilmiş ve hafta içi/sonu farketmeksizin tüm vardiyalar değerlendirmeye tabi tutulmuştur. Analize dâhil edilenlerle birlikte toplam 10 elemanlı vardiya kümesi oluşturulmuştur ($S = \{ \text{"00-08", "08-16", "09-17", "10-18", "11-19", "12-20", "13-21", "14-22", "15-23", "16-00"} \}$).

Ele alınan vardiya düzenleri Şekil 4.1.'de görselleştirilmiştir. İlgili şekilde yeşil renkteki vardiyalar firmada hali hazırda kullanılan vardiyalar olup turuncu renktekiler ise ekstradan analize dahil edilenlerdir.

Firmada her bir operatör atandığı vardiyada 4 tip mola vermektedir ($|Y|=4$). 1., 3. ve 4. mola tipleri dinlenme molalarını, 2.tip mola ise yemek molasını ifade etmektedir. Her bir vardiya için mola tiplerine ait periyot aralıkları da Tablo 4.1.'de gösterildiği gibidir. Her bir vardiyada dinlenme molası için 1,5 saatlik mola zaman aralıkları bulunurken, yemek molası için 2,5 saatlik mola zaman aralığı vardır. Mola zaman aralıklarına örnekler Şekil 4.2.'de gösterilmiştir.



Şekil 4.1. Ele alınan vardiya düzenleri.



Şekil 4.2. Mola gösterimli vardiya düzeninden bir kesit.

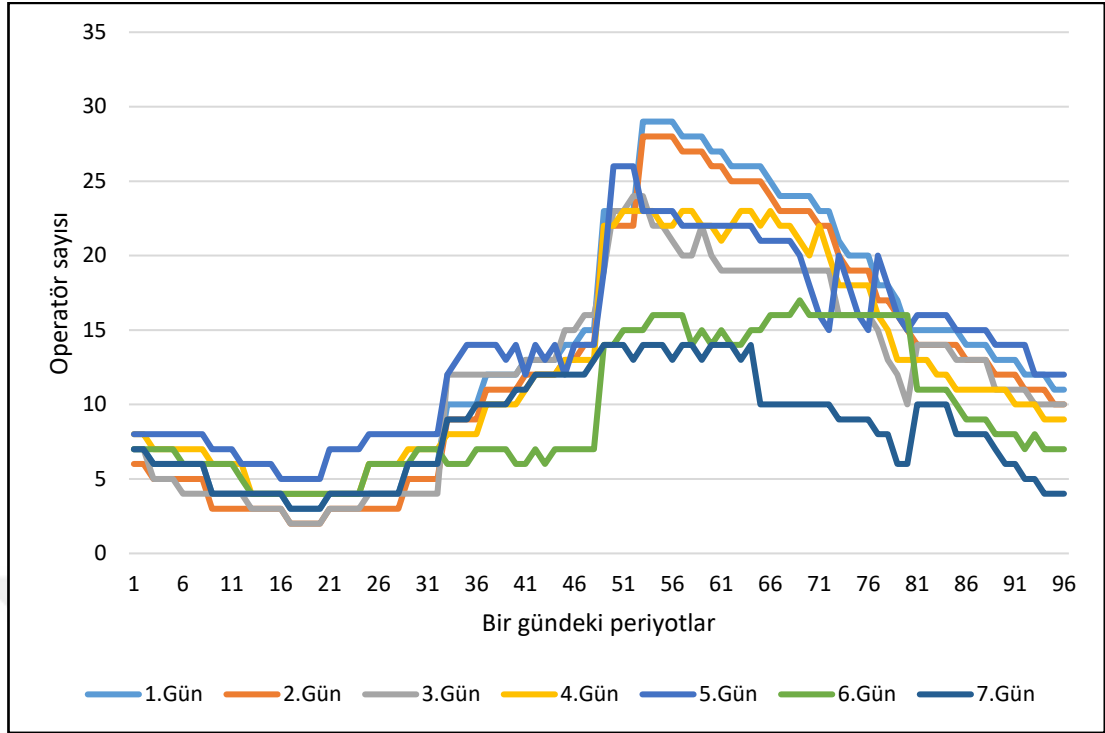
Şekil 4.2.'de görüldüğü üzere (Tablo 4.1.'e de bakılabilir) örnek olarak 4. vardiya, 41 ile 72. periyotlar arasını içermekte ve bu vardiyanın 1. dinlenme molasının verilebileceği periyotların kümesi $B_{14}=\{43, 44, \dots, 48\}$, yemek molasına çıkılabilecek periyotların kümesi $B_{24}=\{51, 52, \dots, 57\}$ (son periyotta molaya çıkılamayacağı için mola penceresine dahil edilmemiştir), 2. dinlenme molasının verilebileceği periyotların kümesi $B_{34}=\{61, 62, \dots, 66\}$ ve 3. dinlenme molasının verilebileceği periyotların kümesi $B_{44}=\{67, 68, \dots, 72\}$ şeklinde olmaktadır. Hiçbir operatör mola için belirlenen periyot aralıkları dışındaki bir periyota mola vermek üzere atanamaz.

Bu kümelere bağlı olarak da y tipindeki bir molanın s . vardiyadaki başlangıç (δ_{ys}) ve bitiş periyotları (ε_{ys}) da belirlidir. Örneğin yine 4. vardiyadaki 1. dinlenme mola aralığı $\delta_{14}=43$. periyotta başlamakta ve $\varepsilon_{14}=48$. periyotta sona ermektedir. Vardiya uzunlukları da tüm vardiyalar için 8'er saattir ($W_{vs \in S}=8$).

Firmada 76 operatör çalışmaktadır ($|A|=76$). Bu operatörler günde sadece bir vardiyada çalışabilmekte ve 7 günlük planlama ufkunda en az bir gün izinde olmaktadır. Bir vardiyaya kapasite kısıtından dolayı da en fazla 25 operatör atanabilmektedir ($\gamma = 25$).

Çalışanlar çalıştıkları gün her bir molaya ilgili molalar için ayrılan zaman dilimlerinde çıkmak zorundalar ve molada durma uzunlukları dinlenme molaları için 1 periyot (L_1, L_3 and $L_4=1$), yemek molası için ise 2 periyot ($L_2=2$) olmaktadır. Diğer bir ifadeyle bir operatör bir çalışma gününde 3 adet 15 dakikalık dinlenme molası ve 1 adet 30 dakikalık yemek molası vermektedir. Firmanın planlama dönemi olan yedi gün için on beş dakikalık periyot başına ihtiyaç duyduğu işgücü sayısı (N_{dp}) da bilinmektedir ve Şekil 4.3.'de gösterilmiştir.

Problemde anlatılan organizasyon için, yedi günlük planlama ufkunda öncelikle vardiya çizelgeleme problemi çözülecektir. Bu problem çözümü neticesinde d . gün s . vardiyaya atanacak operatör sayıları (G_{ds}) ortaya çıkmaktadır. Ancak bu sayılar personel çizelgeleme probleminde, operatörlerin mola vermeleri esnasında periyot başına istenen talebi sağlayabilmek adına belirli oranda artırılarak kullanılmaktadır.



Şekil 4.3. Planlama döneminde periyot başına ihtiyaç duyulan operatör miktarı.

Bu işlemin yapılmaması personel çizelgeleme probleminde olursuz çözüme neden olacaktır. Zira vardiya çizelgeleme probleminde periyot başına istenen talebe göre fazla operatör miktarı minimize edilmektedir ve bu durumda mola pencerelerindeki fazlalık miktarın “0” olması ya da çok düşük olması her günün her bir periyodunda yeterli operatörün olmasını garantileyen kısıtın sağlanamamasına neden olacaktır.

Güncellenen G_{ds} , bu parametre yardımıyla oluşturulan görevler tanımlama grubu (T) ve atama yapılan vardiyalar tanımlama grubu (S^A) personel çizelgeleme problemi için girdileri oluşturacaktır. Girdi verilerinin geri kalanı Tablo 4.2.’de gösterilmiştir.

Tablo 4.2. Model 1 için diğer girdi verileri

Parametre	Değer	Birim
C^H	12	TL/Operatör (Saat başına)
C^P	3	TL/Operatör (Periyot başına)
C^S	300	TL/Vardiya
C^U	100	TL/Operatör (Periyot başına)
H	7	Gün

TL=Türk Lirası

4.1.2. Uygulama ve hesaplamalı sonuçlar

Bu bölümde öncelikle vardiya çizelgeleme probleminin bir önceki bölümde detayları anlatılan örnek kapsamındaki çözümlerine yer verilecek olup daha sonra personel çizelgeleme problemine ait her iki modelden (TSP/KP) elde edilen çözüm sonuçlarına değinilecektir.

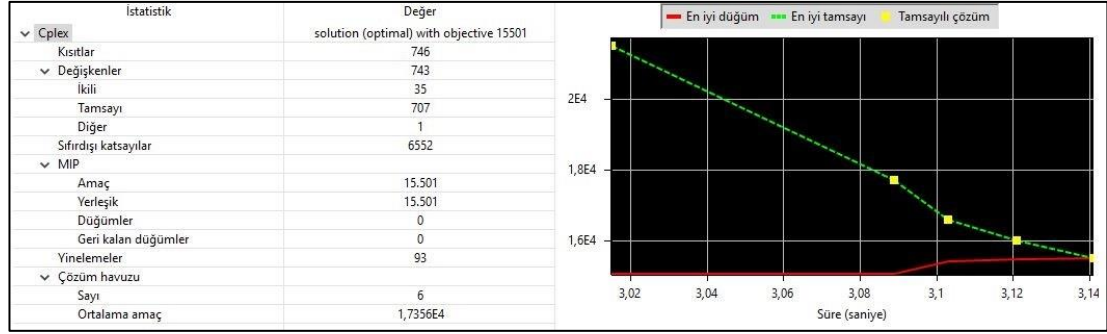
Tüm modeller daha öncede belirtildiği üzere IBM Akademik Girişim (IBM Akademik Initiative) web sitesinde akademisyenlere ücretsiz olarak sunulan IBM ILOG CPLEX Optimization Studio 12.6.0 adlı programda geliştirilmiştir. Modeller, Intel i5-3210M 2.5 GHz işlemci ve 8 GB belleğe sahip bir dizüstü bilgisayarda çalıştırılmıştır. Hesaplama zamanı 1 saat (3600 saniye) olarak ayarlanmıştır. Tablolarda verilen tüm zamanlar saniye biriminde yazılmıştır.

Vardiya çizelgeleme modeli (Model 1) hem mevcut vardiya düzeninde hem de analize dâhil edilen vardiyalar düzeninde çözülmüştür. Her iki durum için optimal çözüm elde edilmiş olup, önerilen durumda mevcut durumun amaç fonksiyonuna göre yaklaşık %15 iyileştirme sağlanmıştır.

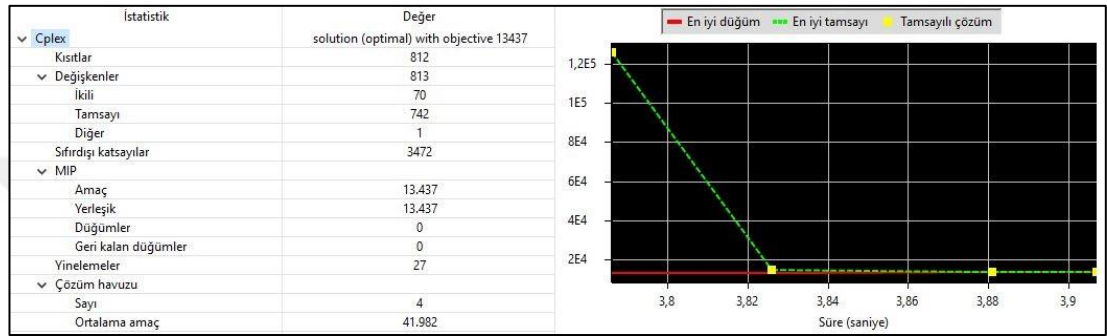
Model 1'e ait çözüm detayları Tablo 4.3.'deki gibidir:

Tablo 4.3. Model 1'in çözüm sonuçlarına ait detaylar				
Problem	Amaç Fonksiyonu Değ.	Zaman (s)	Kısıt Sayısı	Karar Değişkeni Sayısı
Mevcut Düzen	15501	4,87	746	743
Önerilen Düzen	13437	4,95	812	813

Mevcut durumun Model 1 ile çözümünü gösterir IBM ILOG CPLEX programı İstatistikler alt bölümüne ait ekran görüntüsü Şekil 4.4.'deki gibidir. Çözümler ve Komut dosyası günlüğünü içerir bilgilere de Ek-E'de yer verilmiştir. Önerilen durumun Model 1 ile çözümünü gösterir IBM ILOG CPLEX programı İstatistikler alt bölümüne ait ekran görüntüsü de Şekil 4.5.'de gösterilmiştir. Çözümler ve Komut dosyası günlüğünü içerir bilgilere de Ek-F'de yer verilmiştir.



Şekil 4.4. Mevcut durumun Model 1 ile çözümünü gösteren ekran görüntüsü.



Şekil 4.5. Önerilen durumun Model 1 ile çözümünü gösteren ekran görüntüsü.

Her iki durum için Tablo 4.4.’deki gibi gün-vardiya sonuçları elde edilmiştir. Tablo 4.4.’deki hücreler ilgili gün ilgili vardiyaya kaç adet operatör atanması gerektiğini göstermektedir. “0” içeren hücreler, ilgili gün ilgili vardiyanın açılmayacağını dolayısıyla bir atamanın söz konusu olmayacağını ifade etmektedir.

Tablo 4.4. Günlere göre vardiyalara atanan operatör sayıları (G_{ds})

Gün	Vardiya																			
	1		2		3		4		5		6		7		8		9		10	
	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö	M	Ö
1	7	7	13	15	-	0	9	0	-	0	7	14	-	0	-	0	-	0	15	15
2	6	6	14	17	-	0	8	0	-	0	6	11	-	0	-	0	-	0	14	14
3	7	7	15	13	-	0	7	0	-	11	5	0	-	0	-	0	-	0	14	15
4	8	8	13	13	-	0	5	0	-	0	5	10	-	0	-	0	-	0	13	13
5	8	8	14	14	-	0	6	0	-	0	7	12	-	0	-	0	-	0	16	16
6	7	7	16	10	-	0	0	0	-	0	0	6	-	0	-	0	-	0	17	11
7	7	7	14	14	-	0	0	0	-	0	0	0	-	0	-	0	-	0	10	10

M- Mevcut Durum, Ö-Önerilen Durum

Planlama döneminde mevcut durum için toplam 31 adet vardiya açılırken ($|R| = 31$), önerilen durum için 27 adet açılmıştır ($|R| = 27$). İlgili planlama döneminde

karşılanmayan talep gözlenmemiştir. Mevcut durum için toplam 672 (96 periyot*7 gün) periyotun 560'ında gerekenden en az 1 adet fazla operatör bulundurulmuştur. Önerilen durum için bu rakam 544'tür. Fazlalık olan bu rakamlar bir sonraki aşamada (personel çizelgeleme probleminde) molaların verilmesi esnasında ilgili periyotta ihtiyaç duyulan kadar operatör bulundurulması kısıtını sağlamaya yardımcı olacaktır.

Şu anda talepleri karşılayacak en uygun vardiyalar (S^A) belirlenmiş durumdadır. Artık mevcut operatörlerin oluşturulan vardiyalara atanmasına, molaya çıktıkları periyotların ve izin günlerinin belirlenmesine yönelik model çözümlerine yer verilecektir.

Bir önceki bölümde vardiyalara atanan operatör sayılarında belirli oranlarda artışlar yapılacağı belirtilmişti. Model çözümleri vardiya çizelgeleme problemindeki önerilen durum kapsamında bu artış oranları ile değerlendirilecektir. Artış oranları sonucu oluşan değerlere yukarı yuvarlama yapılarak tamsayı elde edilmiştir.

Model 2a için CPLEX parametrelerinde zaman sınırı koymanın dışında bir değişiklik yapılmamıştır. Model 2b için ise programdan daha efektif sonuçlar alabilmek adına yayılma (propagation) ve aramayı kontrol edebilmek ayrıca arama aşamalarını belirlemek üzere parametre ayarlaması yapılmıştır.

Programda birçok parametre ayarı varsayılan (default) olarak belirlenmiş durumdadır. Çalışmada çözüm için en iyi parametreler seçilmeye çalışılmıştır. Kullanılan parametreler hakkında bilgi Tablo 4.5.'te gösterilmiştir. Tabloda yer verilmeyen parametreler programın varsayılan parametreleridir.

Tablo 4.5. KP çözücüsünde kullanılan parametre ayarları

Parametre	Ayar Değeri
Hata Limiti (Fail limit)	100000
İşlemede Çalışan İşçi Sayısı (Number of workers)	1
Arama Fazı (Search phase)	x_{ka}
Arama Tipi (Search type)	Yeniden Başlatma (Restart)

KP çözücünde optimal çözüm almak uzun hesaplama zamanı alabilmektedir. Bu nedenle bir durdurma kuralı (sınırı) belirlenmesi gerekmektedir. Bu, bir zaman sınırı olabileceği gibi hata (başarısızlık) limiti, dal sınırı ve/veya çözüm sınırı da olabilmektedir. Bu çalışmada durdurma kuralı olarak sadece hata limiti tercih edilmiş ve bu değer 100000 olarak ayarlanmıştır. Bu, en fazla 100000 adet hatanın ortaya çıkmasıyla birlikte aramanın sonlandırılmasını ifade etmektedir. Diğer sınırlar varsayılan durumda bırakılmıştır.

Bilgisayar işlemcisinde çalışan işçi sayısı n 'ye ayarlanırsa, KP çözücü motoru her biri kendi iş parçacığında sorunu çözmek için birlikte çalışacak olan n işçi oluşturur. Birden çok işçi kullanmak, tek bir işçi kullanmaktan daha fazla bellek gerektirmektedir. Bu anlamda bazı örnekler bazında n değeri 1'den büyük olarak seçilmiş ve çözüm esnasında aşırı bellek kullanımı göze çarpmıştır. Sonuç olarak tüm örneklerin çözümü kapsamında n değerinin 1 olarak belirlenmesine karar verilmiş ve bu şekilde çözümler alınmıştır.

Bir arama fazı ise, arama hareketlerinin sırasının ve değerlerin test edilmesi gereken sıranın belirlenmesine olanak sağlamaktadır. Operatörlerin vardiyalara atanmasını önceliklendirmek adına arama fazı Tablo 4.5.'de görülen karar değişkeni şeklinde belirlenmiştir.

Arama stratejisi olarak bazı örnekler kapsamında KP çözücündeki tüm arama stratejileri (Çok noktalı, Derinlik öncelikli, Otomatik ve Yeniden başlatma) sırasıyla denenmiş ve “Yeniden Başlatma” arama stratejisi ile daha etkin sonuçlar alındığı gözlenmiştir. Régin de [91] çalışmasında farklı arama tiplerini irdelemiş ve “Yeniden Başlatma” arama tipi ile daha efektif sonuçlar alınabileceğini göstermiştir. Bu nedenle tüm örneklerin çözümü kapsamında arama stratejisi olarak “Yeniden Başlatma” seçilmiş ve bu şekilde çözümler alınmıştır.

Modellerin çözüm detayları Tablo 4.6.'daki gibidir. Çözüm sonuçlarına göre vardiya çizelgeleme probleminin çözümü neticesinde ortaya çıkan vardiya başına atanan operatör miktarında (G_{ds}) yapılacak artışlar, oranın %22'den az olması ve %46'dan

büyük olması durumunda personel çizelgeleme problemi için olurlu bir çözüm vermemektedir. Bu iki değer arasındaki tüm sonuçlar her iki model için de optimal sonuç vermektedir. Oranın %22 veya %23 yapılması ile toplam $\tau=386$ atama yapılmış ve en düşük optimal çözüm ($386 \times 12 \text{ TL/saat} \times 8 \text{ saat}=37056$) elde edilmiştir.

Tablo 4.6. Model 2a ve Model 2b'ye ait çözüm detayları

Artır.	τ	Model 2a (TSP)				Model 2b (KP)			
		Amaç	Zaman	Kısıt	Karar	Amaç	Zaman (s)	Kısıt	Karar
Oranı		Fonk.	(s)	Sayısı	Değ.	Fonk.	(Hata Lim.)	Sayısı	Değ.
(%)		Değeri			Sayısı	Değeri			Sayısı
15	365	Olursuz ^a				Çözüm Yok ^b			
20	376	Olursuz ^a				Çözüm Yok ^b			
21	382	Olursuz ^a				Çözüm Yok ^b			
22*	386	37056	797,23			37056	7,07 (263,06)	264586	122992
23*	386	37056	886,41			37056	7,09 (263,65)	264586	122992
24	390	37440	898,57			37440	6,16 (231,89)	267066	124224
25	390	37440	906,35	677214	876709	37440	6,29 (229,30)	267066	124224
30	409	39264	915,33			39264	6,75 (260,41)	278846	130076
46	456	43776	928,12			43776	8,09 (209,65)	307986	144552
47	462	Olursuz ^a				Çözüm Yok ^b			

Kalın punto ile yazılan değerler optimal değerlerdir.

*En düşük optimal değere karşılık gelen artırma oranı

^aTam sayılı bir çözüm bulunamadı.

^bArama tamamlandı, çözüm bulunamadı.

Örnek olması açısından Model 2a ve Model 2b'de %22'lik artışa karşılık gelen duruma ait IBM ILOG CPLEX çıktılarına da sırasıyla Ek-G ve Ek-H'da yer verilmiştir. Özellikle Ek-G'de Komut dosyası günlüğü alt başlığı altında görülebileceği üzere modelin çalıştırılması/çözülmesi sonucu Tablo 4.7.'deki gibi birçok farklı çözüm çıktısı alınabilmektedir. Aynı sonuçlar Model 2b'den de alınabilir. Tekrar olmaması açısından Ek-H'da yeniden yer verilmemiştir.

Tablo 4.7. IBM ILOG CPLEX'ten alınabilecek bazı çözüm çıktıları

Alınabilecek çözüm değeri	Çıktı örneği ¹
i. Gün m Vardiyasında Çalışanlar (Satırlar GÜN, Sütunlar VARDİYA)	[[{8 10 19 20 28 36 53 57 71} {3 6 15 29 30 39 41 44 45 49 50 55 56 59 60 61 64 67 74} {} {} {} {1 5 12 14 18 26 27 32 33 34 35 37 40 46 47 48 52 65} {} {} {} {2 4 9 11 13 16 21 31 42 43 51 54 58 62 68 69 70 72 76}] ... 1. Gün 00-08 Vardiyasında Çalışanlar= {8 10 19 20 28 36 53 57 71} 1. Gün 08-16 Vardiyasında Çalışanlar= {3 6 15 29 30 39 41 44 45 49 50 55 56 59 60 61 64 67 74} 1. Gün 09-17 Vardiyasında Çalışanlar= {} ...
i. Gün m Vardiyasında Toplam Çalışan Sayısı (Satırlar GÜN, Sütunlar VARDİYA)	[[9 19 0 0 0 18 0 0 0 19] [8 21 0 0 0 14 0 0 0 18] [9 16 0 0 14 0 0 0 0 19] [10 16 0 0 0 13 0 0 0 16] [10 18 0 0 0 15 0 0 0 20] [9 13 0 0 0 8 0 0 0 14] [9 18 0 0 0 0 0 0 0 13]]
i. Gün Çalışanlar (Satırlar GÜN)	1. Gün 00-08 Vardiyasında Toplam 9 işçi çalışmaktadır. 1. Gün 08-16 Vardiyasında Toplam 19 işçi çalışmaktadır. 1. Gün 09-17 Vardiyasında Toplam 0 işçi çalışmaktadır... 1. Gün Çalışanlar= {1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 18 19 20 21 26 27 28 29 30 31 32 33 34 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 64 65 67 68 69 70 71 72 74 76} 2. Gün Çalışanlar= {1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 28 30 31 33 34 35 37 38 39 41 42 44 45 46 47 49 50 51 52 53 54 55 58 59 60 62 63 65 67 68 69 70 71 72 73 75 76} ...
i. Gün Çalışmayanlar	1. Gün Çalışmayanlar= {7 17 22 23 24 25 38 63 66 73 75} 2. Gün Çalışmayanlar= {3 22 27 29 32 36 40 43 48 56 57 61 64 66 74} ...
j. İşçinin Toplam Çalıştığı Süre (Sütunlar İŞÇİ)	[40 48 32 32 32 32 40 48 32 40 40 48 40 40 48 40 40 48 40 32 40 24 32 ...
j. İşçinin i. Gün Çalışıp Çalışmadığı (Satırlar İŞÇİ, Sütunlar GÜN)	[[1 1 1 1 1 0 0] [1 1 1 1 1 0 1] [1 0 0 1 1 0 1] ... 3. İşçi {1 4 5 7} günlerinde çalışmakta {2 3 6} günde/günlerinde çalışmamaktadır...
j. İşçinin i. Gün Çalıştığı Vardiya (Satırlar İŞÇİ, Sütunlar GÜN)	[[{"12-20"} {"08-16"} {"08-16"} {"00-08"} {"16-00"} {} {}] [{"16-00"} {"16-00"} {"16-00"} {"08-16"} {"08-16"} {} {"08-16"}] ...
j. İşçinin i. Gün Çalıştığı Vardiyada molalarını p. periyotlarda vermesi	1. İşçi 1. Gün 12-20 Vardiyasında çalışmakta ve ilk molasını {54}. periyotta, yemek molasını {65}. periyotta, ikinci molasını {73}. periyotta ve üçüncü molasını {75}. periyotta vermektedir...
i. Gün p. Periyot Birinci Molasına Çıkanlar ² (Satırlar GÜN, Sütunlar PERİYOT)	[[{} {} {10} {53 71} {36} {8 20 57} {19} {28} {} {} {} {}] {} {} {} {} {} ... 1. Gün 3. Periyotta {10}. işçi/işçiler birinci molaya çıkacaktır. 1. Gün 4. Periyotta {53 71}. işçi/işçiler birinci molaya çıkacaktır...
j. İşçinin i. Gün Birinci Molaya Çıktığı Periyot ² (Satırlar İŞÇİ, Sütunlar GÜN)	[[{54} {40} {40} {6} {71} {} {}] [{70} {67} {69} {40} {35} {} {40}] ...

¹“...” ile belirtilen kısımlar ilgili metnin devamı olduğunu, fakat alan kısıtından dolayı yer verilemediğini göstermektedir.² Benzer çıktılar diğer mola tipleri için de alınabilir.

İki model birbiri ile kıyaslandığında çözüm zamanı açısından KP, TSP'ye göre şaşırtıcı derecede daha iyi bir performans göstermiştir. Tablo 4.6'da KP'deki zaman sütununda parantez dışındaki değer optimum sonuca ulaşılan süreyi göstermektedir. Parantez içindeki değer ise önceden belirlenen hata limitine ulaşılan süreyi göstermektedir.

TSP en küçük optimal sonuca 797,23 saniyede ulaşırken, KP bu değere 7,07 saniyede ulaşmıştır. TSP'nin optimal sonuca ulaşma süresi KP'nin optimal sonuca ulaşma süresinin ortalama olarak yaklaşık 110 katıdır.

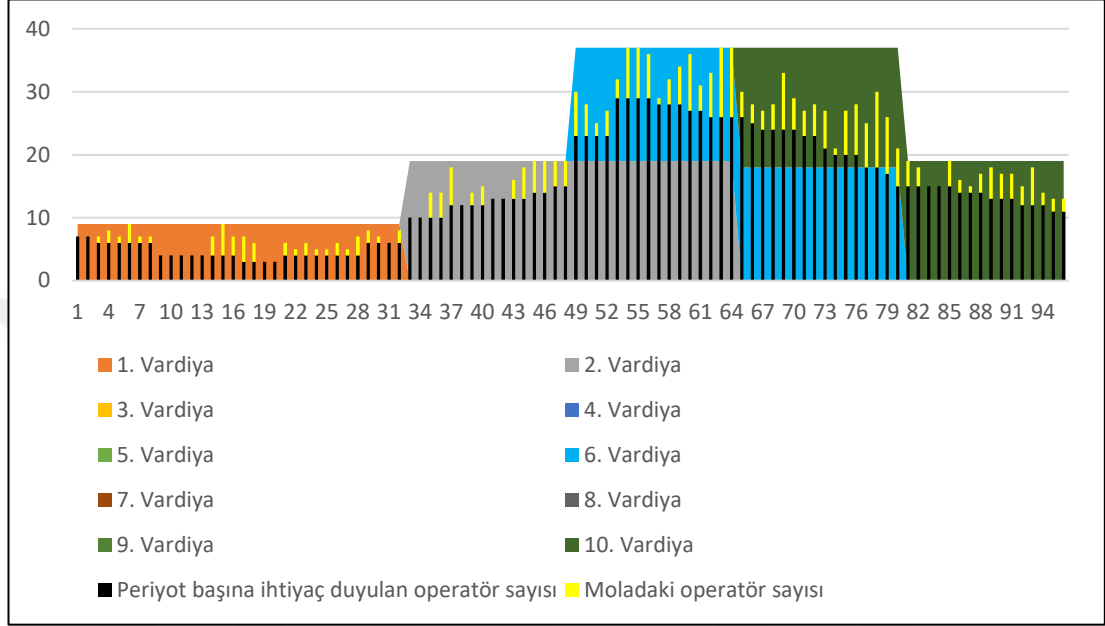
TSP modelinde toplam kısıt ve değişken sayısı artış oranına bağlı olarak değişmemiştir. Fakat, KP'de planlama döneminde yapılan toplam atama sayısı (τ) ile doğru orantılı bir artış söz konusudur. Zira toplam atama sayısının artmasına paralel olarak toplam görev kümesinin eleman sayısı ($|J| = \tau|Y|$) artmaktadır. Her bir operatör 4 tip göreve kaynak teşkil edeceğinden dolayı, en küçük optimal çözümdeki 386 atamaya karşılık toplam $386 \times 4 = 1544$ görev oluşturulmuştur. Aynı atama sayıları için kısıt ve değişken sayıları toplamı aynıdır.

TSP'deki toplam kısıt sayısı KP'dekinin ortalama yaklaşık 2,5 katı iken, bu oran toplam değişken sayısı için ortalama yaklaşık 7 kattır. TSP'de toplam değişken sayısı toplam kısıt sayısından fazla iken, KP için tersi durum söz konusudur.

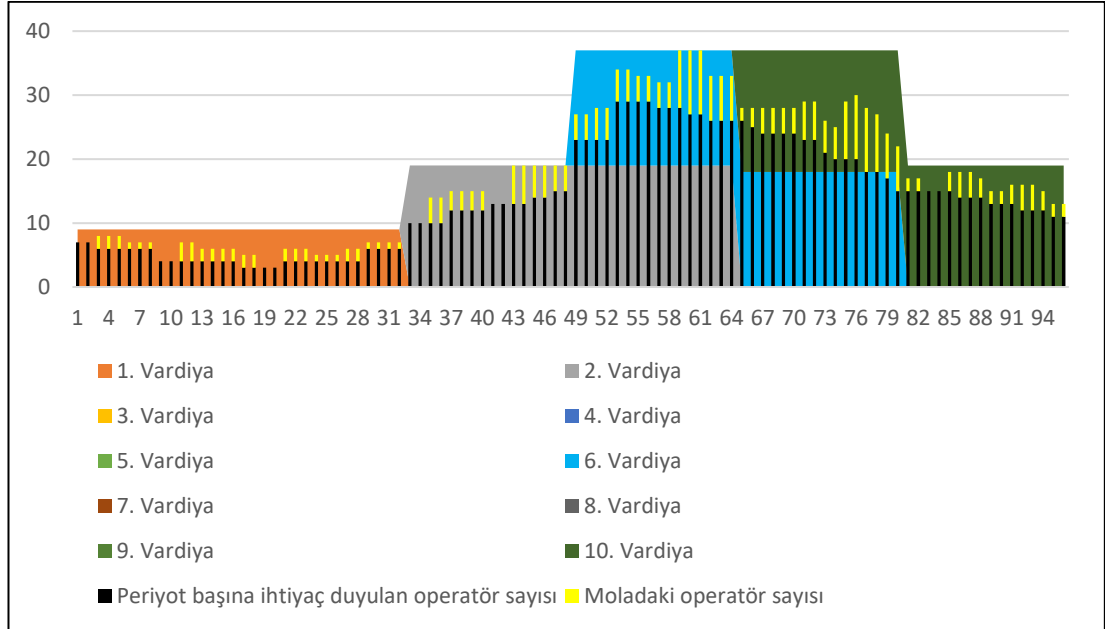
%22'lik artış oranına göre güncellenmiş G_{ds} değerleri Tablo 4.8.'deki gibidir.

Tablo 4.8. %22'lik artış oranına göre güncellenmiş G_{ds} değerleri										
Günler	Vardiyalar									
	1	2	3	4	5	6	7	8	9	10
1	9	19	0	0	0	18	0	0	0	19
2	8	21	0	0	0	14	0	0	0	18
3	9	16	0	0	14	0	0	0	0	19
4	10	16	0	0	0	13	0	0	0	16
5	10	18	0	0	0	15	0	0	0	20
6	9	13	0	0	0	8	0	0	0	14
7	9	18	0	0	0	0	0	0	0	13

Şekil 4.6. ve Şekil 4.7.'de 1.gün için vardiyalara atanan işçi miktarları (kutular) ile ihtiyaç duyulan ve molada olan işçi sayıları (çubuklar ile) sırasıyla TSP ve KP için gösterilmektedir. Her iki model için 1.gün fazlalık olan operatör sayısı ortalama 4,45 operatör/periyo'tur (427 operatör/96 periyot).

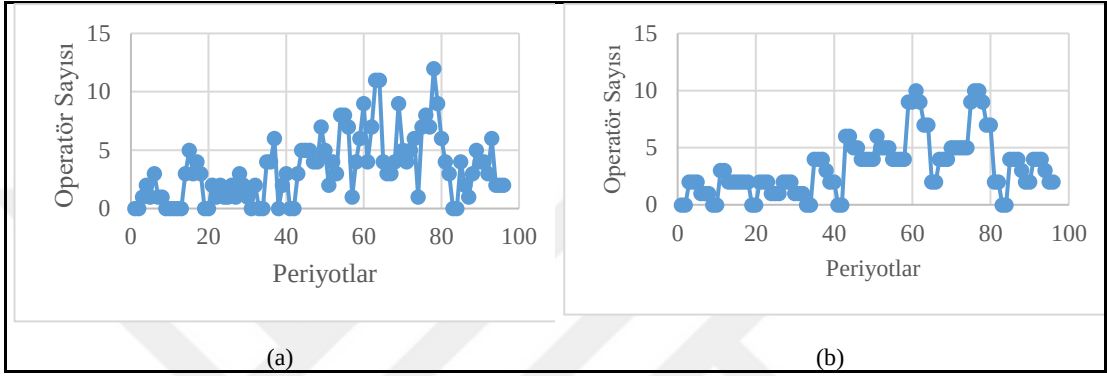


Şekil 4.6. Model 2a'ya ait özet değerler (1. Gün).



Şekil 4.7. Model 2b'ya ait özet değerler (1. Gün).

Şekil 4.8.'de sırasıyla TSP (Şekil 4.8a) ve KP (Şekil 4.8b) modelleri için 1. günde periyot başına molaya çıkan operatör sayıları gösterilmiştir. Şekillerde görüldüğü üzere mola periyotlarına operatörlerin yayılımı KP modelde TSP'ye göre daha homojendir. Bu şunu ifade etmektedir: Örneğin bir vardiyaya 18 operatör atanmışsa, 6 periyotluk bir mola penceresinin her bir periyodunda 3 kişinin molaya çıkması homojen dağılımı gösterir.



Şekil 4.8. Periyot başına molada olan operatör sayıları (1. Gün).

4.2. Deneysel Çalışmalar

Bu bölümde, personel çizelgeleme problemi için önerilen algoritmalar öncelikle bir önceki bölümde kullanılan veri kümesinden türetilen bazı örnekler üzerinde test edilmiştir. Daha sonra aynı işlem literatürde bu çalışmada incelenen problem için birebir bir kıyaslama seti bulunmadığı için www.schedulingbenchmarks.org adlı web sitesinden alınan benzer test örnekleri [8] üzerinde yapılmıştır.

Modelin bu örneklerdeki kıyaslamayı yapabilmesi adına yeniden dizayn edildiği daha önce belirtilmişti. Bu karşılaştırmalı test örnekleri, endüstriyel çalışmalar ve bilimsel yayınlar dahil olmak üzere çeşitli kaynaklardan derlenmiş oldukça kullanışlı verilerdir. Örnekler, gerçek dünya uygulamalarına dayanmaktadır.

4.2.1. Simüle test örnekleri ve elde edilen sonuçlar

Bu bölümde personel çizelgeleme modellerinin (Model 2a ve Model 2b) performanslarını karşılaştırmak için, gerçek verilerini kullandığımız firmadan alınan

verilerdeki periyot başına istenen operatör talebini küçükten büyüğe değiştirerek 11 adet test örneği oluşturulmuştur. Bu şekilde, periyot başına istenen talep miktarlarının artışına bağlı olarak personel çizelgeleme modellerinin çalışma durumlarının karşılaştırılması düşünülmüştür.

Test örneklerini oluşturmadaki temel nokta periyot başına istenen operatör talebini belli oranlarda artırmak ve azaltmaktır. Yeni örnek verileri MS Excel kullanarak oluşturulmuştur. Şöyle ki, firmadan elde edilen değerler Şekil 4.9.’daki gibi “Mevcut” adlı MS Excel sayfasındaki değerler gibi olsun.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	ORAN	PERİYOTLAR											
2	GÜNLER	1	2	3	4	5	6	7	8	9	...	...	96
3	1	7	7	5	5	5	4	4	4	4	...	...	4
4	2	7	7	5	5	5	4	4	4	4	...	...	4
5	3	7	7	5	5	5	4	4	4	4	...	...	4
6	4	7	7	5	5	5	4	4	4	4	...	...	4
7	5	8	8	8	8	8	8	8	8	7	...	...	6
8	6	7	7	7	7	7	6	6	6	6	...	...	5
9	7	7	7	6	6	6	6	6	6	4	...	...	4
10													

Şekil 4.9. MS Excel’de örnek bir mevcut durum değerleri.

Her bir örnek için yeni değerler mevcut değerlerden YUVARLA formülü kullanılarak oluşturulmuştur. Şekil 4.10.’da da gösterildiği üzere Örnek1’e ait değerler mevcut durumdaki tüm değerlerin (hücrelerin) Örnek1’e ait oran olan 0,35 ayrı ayrı çarpılıp yuvarlanması ile elde edilmiştir. Formül satırında açık formül yapısı görülmektedir.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0,35	PERİYOTLAR											
2	GÜNLER	1	2	3	4	5	6	7	8	9	...	...	96
3	1	2	2	2	2	2	1	1	1	1	...	...	1
4	2	2	2	2	2	2	1	1	1	1	...	...	1
5	3	2	2	2	2	2	1	1	1	1	...	...	1
6	4	2	2	2	2	2	1	1	1	1	...	...	1
7	5	3	3	3	3	3	3	3	3	2	...	...	2
8	6	2	2	2	2	2	2	2	2	2	...	...	2
9	7	2	2	2	2	2	2	2	2	1	...	...	1
10													

Şekil 4.10. MS Excel’de mevcut değerlerden Örnek1’in oluşturulması.

Aynı işlemler tüm örnekler için Tablo 4.9.'da gösterilen oranlar ile tek tek oluşturulmuştur. Oluşturulan değerlerle mantıklı olması için toplam operatör sayıları ve bir vardiyaya atanabilecek maksimum operatör sayıları da Tablo 4.9.'daki gibi alınmıştır.

Tablo 4.9. Test örneklerini oluşturmada kullanılan parametreler

Örnek	Oran	Toplam Operatör Sayısı ($ A $)	Bir Vardiyaya Atanabilecek Maksimum Operatör Sayısı (γ)	Açıklama
1	0,35	27	10	
2	0,50	38	10	
3	0,65	51	10	
4	0,85	66	15	
5	1,00	78	15	Örnek 5 ve 6 için Oran ve $ A $ aynı, γ farklı
6	1,00	78	20	
7	1,20	90	30	Örnek 7 ve 8 için Oran ve γ aynı, $ A $ farklı
8	1,20	109	30	
9	1,80	135	40	
10	2,00	160	40	
11	2,30	185	40	

Öncelikle her bir test örneği Model 1 ile çözülmüştür. Tablo 4.10., test örnekleri için Model 1'in bazı girdi ve çıktı değerleri ile test sonuçlarını göstermektedir.

Tablo 4.10. Test örnekleri ve Model 1'e ait detaylar

Örnek	Periyot Başına Operatör İhtiyacı			Toplam Açılan Vardiya Sayısı	τ	Amaç Fonk. Değeri	Zaman (s)	Kısıt Sayısı	Karar Değişkeni Sayısı
	Min.	Ort.	Maks.						
1	0	3,54	9	23	100	9360	4,62	812	813
2	1	5,51	13	26	147	10806	4,97	812	813
3	1	7,58	18	27	196	11625	4,27	812	813
4	3	10,82	23	26	274	12282	3,97	812	813
5	2	11,40	26	28	299	14121	4,30	812	813
6	2	11,40	26	27	299	13821	4,68	812	813
7	3	14,07	32	27	367	14973	4,62	812	813
8	3	14,07	32	27	367	14973	4,58	812	813
9	4	20,99	47	27	546	18201	4,07	812	813
10	4	22,80	52	27	598	19542	4,07	812	813
11	5	26,66	60	27	701	21648	4,62	812	813

Min.: Minimum Değer, Ort.: Ortalama Değer, Maks.: Maksimum Değer

Tablo 4.10.'da 1 numaralı test örneği için, planlama ufku boyunca periyot başına minimum operatör talebinin 0 olduğu, periyot başına ortalama operatör talebinin 3,54 olduğu ve periyot başına maksimum operatör talep değerinin 9 olduğu anlaşılmaktadır.

Çözüm neticesinde toplam 23 vardiya açılmış olup, bu vardiyalara toplam 100 atama işlemi yapılmıştır. Tüm örnekler için optimum amaç fonksiyonu değerleri, çözüm süreleri, toplam kısıt ve karar değişkeni sayıları Tablo 4.10.'da gösterilmiştir. Tabloda da görüldüğü üzere tüm örnekler için toplam kısıt ve karar değişkeni sayıları aynıdır. Ortalama çözüm süresi 4,43 saniyedir.

Bu aşamadan sonra vardiyalara atanan operatör miktarlarında belirli oranlarda artışlar yapılarak (bir önceki bölümde anlatıldığı gibi) personel çizelgeleme problemi için gerekli olan girdiler oluşturulmuştur. Her bir örneğe ait Model 2a ve Model 2b için girdi değerleri ile test sonuçları Tablo 4.11.'de gösterilmiştir. CPLEX açısından ayar düzenlemesi önceki bölümde anlatıldığı şekilde yapılmıştır.

Her bir örnek için artış oranı %10'dan başlatılarak 5'er 5'er artırım yapılmıştır. Bu değerlerde veya ara değerlerde en küçük optimallığın yakalandığı artış oranına karşılık gelen optimal çözüm bilgileri tabloda gösterilmiştir.

Örneğin 1 numaralı test örneğinde %10 ile %25'lik artış oranları arasında her iki model açısından da uygun bir çözüm bulunamamıştır. En küçük optimal çözüme %26'lık artış oranında ulaşılmıştır. Çözüm neticesinde 141 atama işlemi olmuştur ($\tau=141$). Bu örnekte toplam 27 operatör yer almaktadır ($|A|=27$).

Örneğin 1 numaralı test örneğinde %10 ile %25'lik artış oranları arasında her iki model açısından da uygun bir çözüm bulunamamıştır. En küçük optimal çözüme %26'lık artış oranında ulaşılmıştır. Çözüm neticesinde 141 atama işlemi olmuştur ($\tau=141$). Bu örnekte toplam 27 operatör yer almaktadır ($|A|=27$).

Tablo 4.11. Oluşturulan test örnekleri için çözüm sonuçları

Ör.	Art.	τ	A	Model 2a (TSP)				Model 2b (KP)			
				Amaç Fonk.	Zaman (s)	Kısıt Sayısı	Karar Değiş. Sayısı	Amaç Fonk.	Zaman (s) (Hata Lim.)	Kısıt Sayısı	Karar Değiş. Sayısı
1	10	123	27	Olursuz ^a				Çözüm yok ^b			
	15	127	27	Olursuz ^a				Çözüm yok ^b			
	20	129	27	Olursuz ^a				Çözüm yok ^b			
	25	134	27	Olursuz ^a				Çözüm yok ^b			
	26*	141	27	13536	139,66	331764	444186	13536	0,34 (65,31)	39846	17034
	30	142	27	13632	131,67	331764	444186	13632	0,39 (72,13)	40074	17146
2	10	173	38	Olursuz ^a				Çözüm yok ^b			
	14	183	38	Olursuz ^a				Çözüm yok ^b			
	15*	186	38	17856	222,32	409314	541283	17856	0,75 (128,84)	70969	30992
3	10	223	51	Olursuz ^a				Çözüm yok ^b			
	15	240	51	Olursuz ^a				Çözüm yok ^b			
	20	243	51	Olursuz ^a				Çözüm yok ^b			
	23*	257	51	24672	367,77	500964	656034	24672	0,96 (194,44)	124906	56210
	25*	257	51	24672	362,65	500964	656034	24672	1,04 (198,79)	124906	56210
4	10	316	66	Olursuz ^a				Çözüm yok ^b			
	20	342	66	Olursuz ^a				Çözüm yok ^b			
	25	350	66	Olursuz ^a				Çözüm yok ^b			
	26*	357	66	34272	566,91	606714	788439	34272	1,20 (259,00)	213933	99108
	30	372	66	35712	559,43	606714	788439	35712	1,26 (188,05)	222033	103128
5	10	344	78	Olursuz ^a				Çözüm yok ^b			
	15	360	78	Olursuz ^a				Çözüm yok ^b			
	19*	371	78	35616	778,32	691314	894363	35616	2,65 (305,49)	262823	121604
	20*	371	78	35616	834,24	691314	894363	35616	2,49 (301,74)	262823	121604
6	10	338	78	Olursuz ^a				Çözüm yok ^b			
	15	358	78	Olursuz ^a				Çözüm yok ^b			
	19*	367	78	35232	856,23	691314	894363	35232	2,67 (317,92)	259342	120184
	20*	367	78	35232	861,55	691314	894363	35232	2,55 (311,13)	259342	120,184

Tablo 4.11. (devamı)

Ör.	Art.	τ	A	Model 2a (TSP)				Model 2b (KP)			
				Amaç Fonk.	Zaman (s)	Kısıt Sayısı	Karar Değiş. Sayısı	Amaç Fonk.	Zaman (s) (Hata Lim.)	Kısıt Sayısı	Karar Değiş. Sayısı
7	10	413	90	Olursuz ^a				Çözüm yok ^b			
	15	433	90	Olursuz ^a				Çözüm yok ^b			
	20	448	90	Olursuz ^a				Çözüm yok ^b			
	22*	459	90	44064	1074,75	775914	1000287	44064	3,79 (313,65)	365902	171936
	25	472	90	45312	1133,63	775914	1000287	45312	4,05 (297,34)	375418	176668
8	10	413	109	Olursuz ^a				Çözüm yok ^b			
	15	433	109	Olursuz ^a				Çözüm yok ^b			
	20	448	109	Olursuz ^a				Çözüm yok ^b			
	22*	459	109	44064	1440,44	909864	1168000	44064	5,38 (416,09)	441978	207846
	25	472	109	45312	1616,64	909864	1168000	45312	4,59 (388,22)	453470	213566
9	10	611	135	Olursuz ^a				Çözüm yok ^b			
	15	640	135	Olursuz ^a				Çözüm yok ^b			
	20*	667	135	64032	2670,55	1093164	1397502	64032	9,03 (504,03)	773218	370138
10	10	666	160	Olursuz ^a				Çözüm yok ^b			
	15	703	160	Olursuz ^a				Çözüm yok ^b			
	20	724	160	Olursuz ^a				Çözüm yok ^b			
	25	754	160	Olursuz ^a				Çözüm yok ^b			
	26*	768	160	73728	2967,78	1269414	1618177	73728	12,44 (832,95)	1045410	503232
	30	789	160	75744	3228,78	1269414	1618177	75744	12,89 (826,12)	1072542	516756
11	10	780	185	Olursuz ^a				Çözüm yok ^b			
	15	817	185	Olursuz ^a				Çözüm yok ^b			
	20	852	185	Zaman aşımı ^c				Çözüm yok ^b			
	25*	889	185	Zaman aşımı ^c				85344	20,67 (930,82)	1387842	671406

Kalın punto ile yazılan değerler optimaldir.

*En düşük optimal değere karşılık gelen artırma oranı

^aTam sayılı bir çözüm bulunamadı.

^bArama tamamlandı, çözüm bulunamadı.

^cZaman aşımı (3600 saniye).

Her iki model tüm örnekler açısından karşılaştırıldığında, Model 2b'nin Model 2a'dan çok daha etkin olduğu sonucu ortaya çıkmıştır. TSP modeli, tüm modellerde KP modelinden çok daha fazla sayıda kısıt ve karar değişkeni içermektedir. Yine tüm örnekler açısından TSP modelinde karar değişkeni sayısının kısıt sayısından fazla olduğu gözlenmiştir. Öte yandan, KP'deki kısıt sayısı karar değişkeni sayısından daha fazladır. Herhangi bir örnek ölçeğinde TSP modelindeki kısıt ve değişken sayısı, her bir örnek için artış oranından bağımsız iken, bu değerler, KP modelinde artış oranı ile doğru orantılı olarak ilişkilidir.

Tablo 4.11.'de de görüldüğü üzere KP temelli model saniyeler içinde çözüm üretebilmektedir. KP modelinde zaman sütunundaki parantezin dışındaki değer, optimum sonuca ulaşmak için gereken süreyi gösterirken, parantez içindeki değer hata sınırına ulaşma süresini göstermektedir. Ele alınan tüm test örnekleri kapsamında, TSP'de optimal sonuçlara ulaşma süresi KP'dekinin yaklaşık olarak 290 kat daha fazlasıdır. En büyük boyutlu probleme (11 numaralı örnek) TSP model belirtilen çalışma süresi içinde bir çözüm üretmez iken, KP model sadece 20,67 saniyede bir optimum çözüm üretebilmiştir.

CPLEX'te bu örnek için KP çözümünde 6,3 GB boyutunda bir bellek kullanımı göze çarpmıştır. Bu değer oldukça yüksektir. Yani mevcut CPLEX yapılandırması ve ayarları kullanılarak bu boyuttaki bir problemten daha büyük boyuttaki bir problemin çalıştırılması durumunda çözüm alınamayacağı ortadadır. Büyük boyuttaki problemler için çalışma süresinin artırılması durumunda TSP modeli ile, daha yüksek bellek kapasiteli bir bilgisayar kullanılması durumunda ise KP modeli ile uygun çözümler alınabileceğini belirtmekte fayda görülmektedir.

4.2.2. Kıyaslama test örnekleri ve elde edilen sonuçlar

KP temelli modelin daha objektif değerlendirilebilmesi için modelin herkese açık olan kıyaslama test örnekleri ile test edildiği, test etmek için model üzerinde gerekli yeniden düzenleme işlemlerinin yapıldığı daha önce belirtilmişti. Kullanılan test örneklerine www.schedulingbenchmarks.org adlı internet sitesinden

ulaşılabilmektedir. İlgili sitede her bir örnek için gereken tüm bilgiler (en iyi çözümler, alt sınırlar vb.) bulunmaktadır. Tablo 4.12.'de test örneklerine ait bazı özet bilgilere yer verilmiştir. Modeller çok küçük boyutlu olandan çok büyük boyutlu olana doğru artan düzendedir.

Tablo 4.12. Kıyaslama test örneklerine ait bazı değerler

Örnek	Toplam Hafta Sayısı	Toplam İşçi Sayısı	Toplam Vardiya Tipi Sayısı
1	2	8	1
2	2	14	2
3	2	20	3
4	4	10	2
5	4	16	2
6	4	18	3
7	4	20	3
8	4	30	4
9	4	36	4
10	4	40	5
11	4	50	6
12	4	60	10
13	4	120	18
14	6	32	4
15	6	45	6
16	8	20	3
17	8	32	4
18	12	22	3
19	12	40	5
20	26	50	6
21	26	100	8
22	52	50	10
23	52	100	16
24	52	150	32

Curtois ve Qu [9] çalışmalarında Ejection Chain adlı metasezgisel tabanlı bir model, Branch and Price adlı bir model ve Gurobi v5.6.3 adlı optimizasyon programında yazılmış tam sayılı programlama tabanlı bir model olmak üzere 3 farklı modele yer vermiştir. Geliştirilen KP temelli model sözü edilen kıyaslama test örneklerini çözecek şekilde düzenlenip ilgili testler gerçekleştirilmiştir.

KP temelli model ile [9]'daki modellerin hesaplamalı karşılaştırma sonuçları Tablo 4.13.'de gösterilmiştir. Tablodan da görüleceği üzere, Branch and Price isimli model küçük ve orta ölçekli kıyaslama test örneklerinde oldukça etkindir. Bununla birlikte, aynı model daha büyük örnekler (13 numaralı örnek ve sonrası) için hafıza yetersizliği nedeniyle çözüm üretmekte başarısız olmaktadır.

Öte yandan, KP temelli model 14-19 numara arasındaki örnekler için çözüm getirmeyi başarmıştır. Fakat KP model de 20-24 numara arasındaki örnekler için bellek yetersizliği hatası vermekte, bu örneklerle çözüm üretememektedir. Bu örneklerle Gurobi v5.6.3. de yazılan model de çözümler bulamamıştır.

Ejection Chain adlı yöntem genel olarak optimal çözüme en uzak çözümleri bulmaktadır. Ayrıca çözüme ulaşma noktasında zamansal anlamda çok da etkin olduğu söylenemez. Fakat, çok iyi sonuçlar olmasada 21 numaralı örneğe kadar bir çözüm üretebilmiştir.

Geliştirilen model ilk dört örnek için makul çalışma süreleri içinde optimal çözümler üretebilmiştir. Daha büyük örnekler için ise optimal değere veya bilinen en iyi alt sınır değerine yakın çözümler bulmuştur. Geliştirilen modelin çözüm üretebildiği örneklerin tümünde Ejection Chain adlı yöntemden daha iyi çözümler ürettiği göze çarpmaktadır.

Genel olarak Gurobi v5.6.3. de yazılan model optimal ya da optimale en yakın çözüme ulaşmıştır. Ancak, geliştirilen KP temelli model 15, 18 ve 19. örnekler için en iyi sonuçları vermiştir.

Kıyaslanan modellerin çözüm üretebildikleri örnekler kapsamında bilinen alt sınır ile buldukları çözümler arasındaki boşluk ortalama olarak Ejection Chain için %24,42, Branch and Price için %3,93, Gurobi 5.6.3 için %9,70 ve KP Temelli Model için ise %8,61 şeklindedir. Ejection Chain 4 farklı örnekte, Branch and Price 6 farklı örnekte, Gurobi 5.6.3 14 farklı örnekte ve KP Temelli Model ise 7 farklı örnekte bilinen alt

sınıra yaklaşımda en iyi olma durumuna sahiptir (Tabloda altı çizili değerleri içeren hücrelere karşılık gelen örnekler ve yöntemler).

Tablo 4.13. KP temelli model ile diğer modellerin karşılaştırma sonuçları

Ör.	Alt Sınır	Ejection Chain			Branch and Price			Gurobi 5.6.3			KP Temelli Model		
		10 dakika	60 dakika	% B.	Çöz. Değ.	Zaman (s)	% B.	Çöz. Değ.	Zaman (s)	% B.	Çöz. Değ.	Zaman (s)	% B.
1	607	607	607	<u>0,00</u>	607	0,27	<u>0,00</u>	607	1,62	<u>0,00</u>	607	0,77	<u>0,00</u>
2	828	923	837	1,08	828	0,13	<u>0,00</u>	828	5,22	<u>0,00</u>	828	62,69	<u>0,00</u>
3	1001	1003	1003	0,20	1001	0,45	<u>0,00</u>	1001	13,54	<u>0,00</u>	1001	158,33	<u>0,00</u>
4	1716	1719	1718	0,12	1716	1,5	<u>0,00</u>	1716	158,99	<u>0,00</u>	1716	518,48	<u>0,00</u>
5	1143	1439	1358	15,83	1160	25,61	1,47	1143	1520,2	<u>0,00</u>	1180	3600,2	3,14
6	1950	2344	2258	13,64	1952	10,46	0,10	1950	440,93	<u>0,00</u>	2014	3600,6	3,18
7	1056	1284	1269	16,78	1058	93,73	0,19	1056	2152,5	<u>0,00</u>	1091	3600,5	3,21
8	1281	2529	2260	43,32	1308	3600	<u>2,06</u>	1323	3599,8	3,17	1522	3601,8	15,83
9	247	474	463	46,65	439	76,99	<u>43,74</u>	439	3599,9	<u>43,74</u>	452	3602,9	45,35
10	4631	4999	4797	3,46	4631	113,44	0,00	4631	244,2	<u>0,00</u>	4666	3604,5	0,75
11	3443	3967	3661	5,95	3443	19,11	0,00	3443	109,92	<u>0,00</u>	3455	3610,8	0,35
12	4040	5611	5211	22,47	4046	1336,4	0,15	4040	2303,8	<u>0,00</u>	4409	3676,9	8,37
13	1346	8707	3037	<u>55,68</u>	Yetersiz Hafıza		-	3109	3600,6	56,71	Yetersiz Hafıza		-
14	1277	2542	1847	30,86	Yetersiz Hafıza		-	1280	3600,1	<u>0,23</u>	1345	3606,7	5,06
15	3806	6049	5935	35,87	Yetersiz Hafıza		-	4964	3600	23,33	4548	3625,6	<u>16,31</u>
16	3211	4343	4048	20,68	3323	265,02	3,37	3233	3600	<u>0,68</u>	3332	3601,6	3,63
17	5726	7835	7835	26,92	Yetersiz Hafıza		-	5851	3600	<u>2,14</u>	5911	3608	3,13
18	4351	6404	6404	32,06	Yetersiz Hafıza		-	4760	3600	8,59	4668	3604,7	<u>6,79</u>
19	2945	6522	5531	46,75	Yetersiz Hafıza		-	5420	3605,9	45,66	4900	3626,1	<u>39,90</u>
20	4743	23531	9750	<u>51,35</u>	Yetersiz Hafıza		-	-	3600,1	-	Yetersiz Hafıza		-
21	20868	38294	36688	<u>43,12</u>	Yetersiz Hafıza		-	-	3600,2	-	Yetersiz Hafıza		-
22	-	-	51686	-	Yetersiz Hafıza		-	-	3600,2	-	Yetersiz Hafıza		-
23	-	-	54384	-	Yetersiz Hafıza		-	-	3600,4	-	Yetersiz Hafıza		-
24	?	-	156858	-	Yetersiz Hafıza		-	Yetersiz Hafıza		-	Yetersiz Hafıza		-

Koyu punto ile yazılan değerler optimaldir.

Alt Sınır, Bilinen en iyi alt sınır değerini ifade etmektedir.

% B.=Bulunan çözüm değeri ile alt sınır değeri arasındaki boşluğu % olarak ifade eden değer

Altı çizili olarak yazılan değerler ilgili örnekteki alt sınırı yaklaşıma noktasında en iyi çözümü göstermektedir.

BÖLÜM 5. TARTIŞMA VE SONUÇ

Çağrı merkezleri, işgücü gibi pahalı bir kaynağın yoğun olarak kullanıldığı iş birimleridir. Bu iş birimlerindeki temel amaç, minimum işletme ve işgücü maliyetleri ile maksimum müşteri ve çalışan memnuniyetini sağlamaktır. Bu hedefi karşılamak için, firmalar çeşitli teknikler kullanmak zorundadır. İşgücü çizelgeleme modelleri de bu açıdan yararlı araçlardan biri olarak kabul edilebilir.

Bu tezde çağrı merkezleri için entegre bir vardiya ve personel çizelgeleme problemi ele alınmıştır. Vardiya çizelgeleme problemi kapsamlı bir TSP modeli ile formüle edilmiştir. Bu model, periyot başına gereken operator ihtiyacını ve vardiya tanımlarını girdi olarak alarak periyot başına fazlalık operatör miktarını (maliyetini), karşılanamayan talep miktarını (maliyetini) ve vardiya açma maliyetini minimize etmeyi amaçlamaktadır. Model çıktı olarak planlanlama dönemi içerisinde hangi vardiyaların açılacağını ve her bir vardiyaya kaç operatör atanacağını vermektedir. Bu çıktı verileri personel çizelgeleme modellerinin girdilerini oluşturmaktadır.

Personel çizelgeleme problemini çözmek için ise kıyaslama yapmak adına TSP ve KP tabanlı iki ayrı model geliştirilmiştir. Bu modeller vardiya çizelgeleme probleminin çözümü neticesinde açılacak olan vardiyalara hangi operatörlerin atanacağını, her bir operatörün mola zamanının ve izin gününün belirlenmesini sağlamaktadır.

Geliştirilen modeller; gerçek yaşam verileri, bir dizi senaryo ile oluşturulan test örnekleri ve genel kıyaslama test örnekleri üzerinde çalıştırılmıştır. Deneysel sonuçların çapraz karşılaştırmaları yapılmıştır. Bu karşılaştırmalarda daha çok personel çizelgeleme problemi kapsamında geliştirilen modellere odaklanılmıştır.

Elde edilen sonuçlara göre KP modelleme, modelleme kolaylığı, esnekliği ve oldukça hızlı hesaplama zamanı ile TSP'ye göre daha ön plana çıkmaktadır. Ek olarak KP'de TSP'ye göre mola aralıklarına operatörlerin daha homojen yayılımı dikkat çeken bir diğer nokta olmuştur.

Ayrıca, bu tip problemlerin çözümü için KP'nin etkin bir şekilde kullanılabileceği kıyaslama test örnekleri ile kanıtlanmıştır. Zira geliştirilen KP temelli modelin kıyaslama test örneklerinin test edildiği diğer yöntemlerle yeterince rekabetçi olduğu ortadadır. KP modelleri bu avantajlarını TSP'ye göre daha yüksek bir açıklayıcı güce sahip bir bildirimsel (declarative) mantık dili kullanılmasından almaktadır.

KP tabanlı modellerin çözümünde kullanılan bellek artışına bağlı olarak belli bir noktadan sonra çözüm üretilmez duruma gelinebilir. Yani KP modellerin limitasyonlarından biri model büyüklüğünün artışına bağlı olarak kullanılan bellek miktarının aşırı artmasıdır. Bu durum sadece kullanılan bilgisayarın donanımsal kapasitesi ile alakalıdır (donanımsal sınırlar). Daha yüksek kapasiteli bir bilgisayar kullanımı bu sorunu çözecektir.

Bu çalışmadaki tüm modeller 8 GB kapasiteli bir dizüstü bilgisayarda çalıştırılmıştır. Öte yandan literatürdeki çalışmalarda 48 GB belleğe sahip bilgisayarların kullanıldığı da görülmektedir. Bu açıdan önerilen KP modelin yüksek miktarda belleğe sahip bir bilgisayarda çalıştırılması ile kabul edilebilir çalışma zaman aralığında uygun veya optimal çözümler alınabilecektir. TSP açısından ise zamansal sınırlama (belirli bir çalışma süresi konulması) yüzünden uygun çözüm üretilmemesi sorunu göze çarpmaktadır (yazılımsal sınırlar). Çalışma süresinin artırılması ile uygun çözümler bulunabilir.

Bu çalışmada geliştirilen modellerin kullanımını sadece çağrı merkezleri ile sınırlamak yanlış olur. Hatta denilebilir ki bu tip bir çizelgeleme için çağrı merkezleri en zor alanlardan biridir. Zira bu alan çakışan vardiya düzenleri ve kısa periyotlarda çizelgeleme gibi zorluklar içermektedir. Bu açıdan modeller, parametrelerinde yapılacak değişimlerle çağrı merkezleri dışındaki üretim ve servis sistemlerinde de

kolaylıkla uygulanabilir. Özellikle KP tabanlı modeller bu dönüşümde oldukça esneklik sağlamaktadır.

Gelecek çalışmalarda modele ek gerçek yaşam kısıtları (ilgili gün belirli bir vardiyada çalışan bir operatörün ertesi günde belirli bir vardiyalarda çalışmaması, molaya şu periyotta çıktıysa bir sonraki mola ile arasında en az şu kadar periyot olsun vb.) eklenebilir. Yetenek, kıdem ve verimlilik vb. farklılıklar çalışmaya dahil edilebilir. Planlama dönemi uzunluğu (aylık çizelgeleme gibi) ya da gün için periyot uzunluğu (10 veya 30 dakikalık aralıklarla planlama) farklı alınabilir. Çalışan tercihlerinin dikkate alınması da düşünülebilir.



KAYNAKLAR

- [1] Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., De Boeck, L., Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3), 367–385, 2013.
- [2] Gans, N., Koole, G., Mandelbaum, A., Telephone Call Centers: Tutorial, Review, and Research Prospects. *Manufacturing & Service Operations Management*, 5(2), 79–141, 2003.
- [3] Dietz, D.C., Practical scheduling for call center operations. *Omega*, 39(5), 550–557, 2011.
- [4] Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D., Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1), 3–27, 2004.
- [5] Bhulai, S., Koole, G., Pot, A., Simple Methods for Shift Scheduling in Multiskill Call Centers. *Manufacturing & Service Operations Management*, 10(3), 411–420, 2008.
- [6] Aksin, Z., Armony, M., Mehrotra, V., The Modern Call Center: A Multi-Disciplinary Perspective on Operations Management Research. *Production and Operations Management*, 16(6), 665–688, 2009.
- [7] Sungur, B., Özgüven, C., Kariper, Y., Shift scheduling with break windows, ideal break periods, and ideal waiting times. *Flexible Services and Manufacturing Journal*, 29(2), 203–222, 2017.
- [8] www.schedulingbenchmarks.org, Erişim Tarihi: 15.01.2017.
- [9] Curtois, T., Qu, R., Technical report: Computational results on new staff scheduling benchmark instances. 2017.
- [10] Brucker, P., Qu, R., Burke, E., Personnel scheduling: Models and complexity. *European Journal of Operational Research*, 210(3), 467–473, 2011.

- [11] Edie, L.C., Traffic Delays at Toll Booths. *Journal of the Operations Research Society of America*, 2(2), 107–138, 1954.
- [12] Dantzig, G.B., Letter to the Editor—A Comment on Edie’s “Traffic Delays at Toll Booths.” *Journal of the Operations Research Society of America*, 2(3), 339–341, 1954.
- [13] Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D., An Annotated Bibliography of Personnel Scheduling and Rostering. *Annals of Operations Research*, 127(1–4), 21–144, 2004.
- [14] Alfares, H.K., Operator staffing and scheduling for an IT-help call centre. *European J. of Industrial Engineering*, 1(4), 414, 2007.
- [15] Ásgeirsson, E.I., Sigurðardóttir, G.L., Near-optimal MIP solutions for preference based self-scheduling. *Annals of Operations Research*, 239(1), 273–293, 2016.
- [16] Atlason, J., Epelman, M.A., Henderson, S.G., Optimizing Call Center Staffing Using Simulation and Analytic Center Cutting-Plane Methods. *Management Science*, 54(2), 295–309, 2008.
- [17] Atlason, J., Epelman, M.A., Henderson, S.G., Call Center Staffing with Simulation and Cutting Plane Methods. *Annals of Operations Research*, 127(1–4), 333–358, 2004.
- [18] Avramidis, A.N., Gendreau, M., L’Ecuyer, P., Pisacane, O., Simulation-based optimization of agent scheduling in multiskill call centers. 5th Annual International Industrial Simulation Conference (ISC-2007), London, 255–263, 2007.
- [19] Avramidis, A.N., Chan, W., L’Ecuyer, P., Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions*, 41(6), 483–497, 2009.
- [20] Canon, C., Personnel scheduling in the Call Center industry. *4OR*, 5(1), 89–92, 2007.
- [21] Castillo, I., Joro, T., Li, Y.Y., Workforce scheduling with multiple objectives. *European Journal of Operational Research*, 196(1), 162–170, 2009.
- [22] Cezik, M.T., L’Ecuyer, P., Staffing Multiskill Call Centers via Linear Programming and Simulation. *Management Science*, 54(2), 310–323, 2008.

- [23] Defraeye, M., Van Nieuwenhuyse, I., A branch-and-bound algorithm for shift scheduling with stochastic nonstationary demand. *Computers & Operations Research*, 65, 149–162, 2016.
- [24] Dudin, S., Kim, C., Dudina, O., Baek, J., Queueing System with Heterogeneous Customers as a Model of a Call Center with a Call-Back for Lost Customers. *Mathematical Problems in Engineering*, 2013(3), 1–13, 2013.
- [25] Ertogral, K., Bamuqabel, B., Developing staff schedules for a bilingual telecommunication call center with flexible workers. *Computers & Industrial Engineering*, 54(1), 118–127, 2008.
- [26] Excoffier, M., Gicquel, C., Jouini, O., A joint chance-constrained programming approach for call center workforce scheduling under uncertain call arrival forecasts. *Computers & Industrial Engineering*, 96, 16–30, 2016.
- [27] Gong, J., Yu, M., Tang, J., Li, M., Staffing to Maximize Profit for Call Centers with Impatient and Repeat-Calling Customers. *Mathematical Problems in Engineering*, 2015, 1–10, 2015.
- [28] Green, L. V., Kolesar, P.J., Whitt, W., Coping with Time-Varying Demand When Setting Staffing Requirements for a Service System. *Production and Operations Management*, 16(1), 13–39, 2007.
- [29] Helber, S., Henken, K., Profit-oriented shift scheduling of inbound contact centers with skills-based routing, impatient customers, and retries. *OR Spectrum*, 32(1), 109–134, 2010.
- [30] Hojati, M., Near-optimal solution to an employee assignment problem with seniority. *Annals of Operations Research*, 181(1), 539–557, 2010.
- [31] Ibrahim, R., L'Ecuyer, P., Forecasting Call Center Arrivals: Fixed-Effects, Mixed-Effects, and Bivariate Models. *Manufacturing & Service Operations Management*, 15(1), 72–85, 2013.
- [32] Ingolfsson, A., Campello, F., Wu, X., Cabral, E., Combining integer programming and the randomization method to schedule employees. *European Journal of Operational Research*, 202(1), 153–163, 2010.
- [33] Koole, G., van der Sluis, E., Optimal Shift Scheduling with a Global Service Level Constraint. *IIE Transactions*, 35(11), 1049–1055, 2003.

- [34] Kyngäs, N., Kyngäs, J., Nurmi, K., Optimizing Large-Scale Staff Rostering Instances. *Proceedings of the International Multi Conference of Engineers and Computer Scientists*, Hong Kong, 1524–1531, 2012.
- [35] Liao, S., Delft, C. Van, Koole, G., Dallery, Y., Jouini, O., Call center capacity allocation with random workload. 863–868, 2009.
- [36] Mattia, S., Rossi, F., Servilio, M., Smriglio, S., Staffing and scheduling flexible call centers by two-stage robust optimization. *Omega*, 72, 25–37, 2017.
- [37] Millán-Ruiz, D., Hidalgo, J.I., Forecasting Call Centre Arrivals. *Journal of Forecasting*, 32(7), 628–638, 2013.
- [38] Nah, J.E., Kim, S., Workforce planning and deployment for a hospital reservation call center with abandonment cost and multiple tasks. *Computers & Industrial Engineering*, 65(2), 297–309, 2013.
- [39] Örmeci, E.L., Salman, F.S., Yücel, E., Staff rostering in call centers providing employee transportation. *Omega*, 43, 41–53, 2014.
- [40] Pot, A., Bhulai, S., Koole, G., A Simple Staffing Method for Multiskill Call Centers. *Manufacturing & Service Operations Management*, 10(3), 421–428, 2008.
- [41] Restrepo, M.I., Gendron, B., Rousseau, L.-M., A two-stage stochastic programming approach for multi-activity tour scheduling. *European Journal of Operational Research*, 262(2), 620–635, 2017.
- [42] Robbins, T.R., Harrison, T.P., A simulation based scheduling model for call centers with uncertain arrival rates. 2008 Winter Simulation Conference, Miami, 2884–2890, 2008.
- [43] Robbins, T.R., Harrison, T.P., A stochastic programming model for scheduling call centers with global Service Level Agreements. *European Journal of Operational Research*, 207(3), 1608–1619, 2010.
- [44] Kilincli Taskiran, G., Zhang, X., Mathematical models and solution approach for cross-training staff scheduling at call centers. *Computers & Operations Research*, 87, 258–269, 2017.
- [45] Wallace, R.B., Whitt, W., A Staffing Algorithm for Call Centers with Skill-Based Routing. *Manufacturing & Service Operations Management*, 7(4), 276–294, 2005.

- [46] Weinberg, J., Brown, L.D., Stroud, J.R., Bayesian Forecasting of an Inhomogeneous Poisson Process With Applications to Call Center Data. *Journal of the American Statistical Association*, 102(480), 1185–1198, 2007.
- [47] Bianchi, L., Jarrett, J., Choudary Hanumara, R., Improving forecasting for telemarketing centers by ARIMA modeling with intervention. *International Journal of Forecasting*, 14(4), 497–504, 1998.
- [48] Taylor, J.W., A Comparison of Univariate Time Series Methods for Forecasting Intraday Arrivals at a Call Center. *Management Science*, 54(2), 253–265, 2008.
- [49] Koole, G., Mandelbaum, A., Queueing Models of Call Centers: An Introduction. *Annals of Operations Research*, 113(1/4), 41–59, 2002.
- [50] Avramidis, A.N., Chan, W., Gendreau, M., L'Ecuyer, P., Pisacane, O., Optimizing daily agent scheduling in a multiskill call center. *European Journal of Operational Research*, 200(3), 822–832, 2010.
- [51] Avramidis, A.N., Deslauriers, A., L'Ecuyer, P., Modeling Daily Arrivals to a Telephone Call Center. *Management Science*, 50(7), 896–908, 2004.
- [52] Bhulai, S., Dynamic routing policies for multiskill call centers. *Probability in the Engineering and Informational Sciences*, 23(1), 101–119, 2009.
- [53] Aktekin, T., Call center service process analysis: Bayesian parametric and semi-parametric mixture modeling. *European Journal of Operational Research*, 234(3), 709–719, 2014.
- [54] Brown, L. *et al.*, Statistical Analysis of a Telephone Call Center. *Journal of the American Statistical Association*, 100(469), 36–50, 2005.
- [55] Sencer, A., Basarir Ozel, B., A simulation-based decision support system for workforce management in call centers. *Simulation*, 89(4), 481–497, 2013.
- [56] Archawaporn, L., Wongseree, W., Erlang C model for evaluate incoming call uncertainty in automotive call centers. 2013 International Computer Science and Engineering Conference (ICSEC), Bangkok, 109–113, 2013.
- [57] Thompson, G.M., Labor staffing and scheduling models for controlling service levels. *Naval Research Logistics*, 44(8), 719–740, 1997.
- [58] Henderson, W.B., Berry, W.L., Determining optimal shift schedules for telephone traffic exchange operators. *Decision Sciences*, 8(1), 239–255, 1977.

- [59] Aykin, T., Optimal Shift Scheduling with Multiple Break Windows. *Management Science*, 42(4), 591–602, 1996.
- [60] Aykin, T., A comparative evaluation of modeling approaches to the labor shift scheduling problem. *European Journal of Operational Research*, 125(2), 381–397, 2000.
- [61] Thompson, G.M., Pullman, M.E., Scheduling workforce relief breaks in advance versus in real-time. *European Journal of Operational Research*, 181(1), 139–155, 2007.
- [62] Al-Yakoob, S.M., Sherali, H.D., Multiple Shift Scheduling of Hierarchical Workforce with Multiple Work Centers. *Informatica*, 18(3), 325–342, 2007.
- [63] Firat, M., Hurkens, C. a J., An improved MIP-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3), 363–380, 2012.
- [64] Peng, Y., Lu, D., Chen, Y., A Constraint Programming Method for Advanced Planning and Scheduling System with Multilevel Structured Products. *Discrete Dynamics in Nature and Society*, 2014, 1–7, 2014.
- [65] Demiriz, A., Bagherzadeh, N., Alhussein, A., Using constraint programming for the design of network-on-chip architectures. *Computing*, 97(6), 579–592, 2015.
- [66] Trilling, L., Guinet, A., Magny, D. Le, Nurse scheduling using integer linear programming and constraint programming. *IFAC Proceedings Volumes*, 39(3), 671–676, 2006.
- [67] Métivier, J.-P., Boizumault, P., Loudni, S., Solving Nurse Rostering Problems Using Soft Global Constraints. 15th international conference on Principles and practice of constraint programming, Lisbon, 73–87, 2009.
- [68] Qu, R., He, F., A Hybrid Constraint Programming Approach for Nurse Rostering Problems. *İçinde: Applications and Innovations in Intelligent Systems XVI. 1. Baskı*, Springer, London, 211–224, 2009.
- [69] He, F., Qu, R., A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research*, 39(12), 3331–3343, 2012.
- [70] Rahimian, E., Akartunalı, K., Levine, J., A hybrid integer and constraint programming approach to solve nurse rostering problems. *Computers & Operations Research*, 82, 83–94, 2017.

- [71] Cipriano, R., Di Gaspero, L., Dovier, A., Hybrid Approaches for Rostering: A Case Study in the Integration of Constraint Programming and Local Search. *İçinde: Lecture Notes in Computer Science*. 4030. Baskı, Springer, Berlin, 110–123, 2006.
- [72] Bertels, S., Fahle, T., A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. *Computers & Operations Research*, 33(10), 2866–2890, 2006.
- [73] Yunes, T.H., Moura, A. V, de Souza, C.C., Hybrid Column Generation Approaches for Urban Transit Crew Management Problems. *Transportation Science*, 39(2), 273–288, 2005.
- [74] Laporte, G., Pesant, G., A general multi-shift scheduling system. *Journal of the Operational Research Society*, 55(11), 1208–1217, 2004.
- [75] Demasse, S., Pesant, G., Rousseau, L.-M., A Cost-Regular Based Hybrid Column Generation Approach. *Constraints*, 11(4), 315–333, 2006.
- [76] Côté, M.-C., Gendron, B., Rousseau, L.-M., Modeling the Regular Constraint with Integer Programming. *İçinde: Lecture Notes in Computer Science*. 4510. Baskı, Springer, Berlin, 29–43, 2007.
- [77] Côté, M.-C., Gendron, B., Quimper, C.-G., Rousseau, L.-M., Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1), 54–76, 2011.
- [78] Chapados, N., Joliveau, M., Rousseau, L.-M., Retail Store Workforce Scheduling by Expected Operating Income Maximization. *İçinde: Lecture Notes in Computer Science*. 6697. Baskı, Springer, Berlin, 53–58, 2011.
- [79] Li, H., Womer, K., A Decomposition Approach for Shipboard Manpower Scheduling. *Military Operations Research*, 14(3), 67–90, 2009.
- [80] Larson, J., Johansson, M., Carlsson, M., An Integrated Constraint Programming Approach to Scheduling Sports Leagues with Divisional and Round-Robin Tournaments. *İçinde: Integration of AI and OR Techniques in Constraint Programming*. 8451. Baskı, Springer International Publishing, Cham, 144–158, 2014.
- [81] Rossi, F., Beek, P. Van, Walsh, T., Constraint Programming. *İçinde: Handbook of Constraint Programming*. 1. Baskı, Elsevier, Amsterdam, 1–31, 2006.

- [82] Hillier, F.S., Lieberman, G.J., The incorporation of constraint programming. İçinde: Introduction to Operations Research. 9. Baskı, McGraw-Hill, New York, 515–521, 2010.
- [83] Baptiste, P., Le Pape, C., Nuijten, W., Introduction. İçinde: Constraint-Based Scheduling. 9. Baskı, Springer US, Boston, 1–18, 2001.
- [84] Soto, R. *et al.*, Top- k Based Adaptive Enumeration in Constraint Programming. Mathematical Problems in Engineering, 2015, 1–12, 2015.
- [85] Kanet, J.J., Ahire, S., Gorman, M.F., Constraint Programming for Scheduling. İçinde: Handbook of Scheduling - Algorithms, Models, and Performance Analysis. Chapman and Hall/CRC, New York, 1038–1060, 2004.
- [86] www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/usrcoptimizer.pdf, Erişim Tarihi: 20.01.2015.
- [87] Sahraeian, R., Namakshenas, M., On the optimal modeling and evaluation of job shops with a total weighted tardiness objective: Constraint programming vs. mixed integer programming. Applied Mathematical Modelling, 39(2), 955–964, 2015.
- [88] Sel, C., Bilgen, B., Bloemhof-Ruwaard, J.M., van der Vorst, J.G.A.J., Multi-bucket optimization for integrated planning and scheduling in the perishable dairy supply chain. Computers & Chemical Engineering, 77, 59–73, 2015.
- [89] https://www.researchgate.net/publication/328530729_Detailed_Scheduling_in_IBM_ILOG_CPLEX_Optimization_Studio, Erişim Tarihi: 11.03.2015.
- [90] www.ibm.com/support/knowledgecenter/SSSA5P_12.6.2/ilog.odms.studio.help/pdf/opl_langref.pdf, Erişim Tarihi: 15.04.2015.
- [91] Régin, J.-C., Solving Problems with CP: Four Common Pitfalls to Avoid. İçinde: 17th International Conference on Principles and Practice of Constraint Programming - CP 2011. Springer, Berlin, Germany, 3–11, 2011.

EKLER

EK A: Vardiya çizelgeleme problemi için geliştirilen modele (Model 1) ait IBM ILOG CPLEX kodları

```
/******  
* OPL 12.6.0.0 Model  
* Author: tturker & ademiriz  
* Model 1 Kodları  
******/
```

```
/***/DATA***/
```

```
{string} Vardiyalar = ...;
```

```
int Gun=...;  
range Gunler = 1..Gun;
```

```
int Periyot=...;  
range Periyotlar = 1..Periyot;
```

```
int Kapasite=...;  
int IsciPerMaliyet =...;  
int KarsilanmayandanPerMaliyet =...;  
int VarMaliyet=...;  
int varSure [Vardiyalar]=...;  
int PerBasIsGucuIhtiyaci [Gunler][Periyotlar] =...;  
int VarPeriyot [Vardiyalar][Periyotlar] =...;
```

```
/***/DECISION VARIABLES***/
```

```
dvar boolean VardiyaSecilirse [Gunler][Vardiyalar];  
dvar int+ VardiyayaAtanan [Gunler][Vardiyalar];  
dvar int+ Karsilanmayan [Gunler] [Periyotlar];
```

```
/***/DECISION EXPRESSIONS***/
```

```
dexpr int VardiyayaAtananPer [i in Gunler] [p in Periyotlar]= sum (m in Vardiyalar)  
VarPeriyot [m][p] * VardiyayaAtanan [i][m];
```

```
dexpr int Fazlalik [i in Gunler] [p in Periyotlar]= sum (m in Vardiyalar) VarPeriyot
[m][p] * VardiyayaAtanan [i][m]-PerBasIsGuculhtiyaci [i][p];
```

```
dexpr int TopFazlalik=sum (i in Gunler, p in Periyotlar) Fazlalik[i][p];
```

```
dexpr float TopVarMaliyet=sum (i in Gunler, m in Vardiyalar) (VarMaliyet
*VardiyaSecilirse[i][m]);
```

```
dexpr float TopKarsilanmayanMaliyet=sum (i in Gunler, p in Periyotlar)
(KarsilanmayandanPerMaliyet * Karsilanmayan [i][p]);
```

```
dexpr float TopFazlalikMaliyet=sum (i in Gunler, p in Periyotlar)
Fazlalik[i][p]*IsciPerMaliyet;
```

```
**** OBJECTIVE FUNCTION****
```

```
minimize (TopFazlalikMaliyet+TopVarMaliyet+TopKarsilanmayanMaliyet);
```

```
**** CONSTRAINTS****
```

```
subject to {
```

```
c1: // Her günün her bir periyodunda ihtiyaç duyulan operatör sayısını garantiler.
```

```
forall (i in Gunler, p in Periyotlar)
sum (m in Vardiyalar)
(VarPeriyot[m][p]*VardiyayaAtanan[i][m]+Karsilanmayan[i][p])>=
PerBasIsGuculhtiyaci [i][p];
```

```
c2: // Vardiya açıksa kapasite dahilinde atama yapar.
```

```
forall (i in Gunler, m in Vardiyalar)
VardiyaSecilirse[i][m] <=VardiyayaAtanan[i][m]<= Kapasite*VardiyaSecilirse[i][m];
```

```
/*c_ek: // Mevcut düzende haftasonu sadece 3 vardiya açık
```

```
VardiyaSecilirse [6][3] ==0;
VardiyaSecilirse [6][4] ==0;
VardiyaSecilirse [6][5] ==0;
VardiyaSecilirse [6][6] ==0;
VardiyaSecilirse [6][7] ==0;
VardiyaSecilirse [6][8] ==0;
VardiyaSecilirse [6][9] ==0;
VardiyaSecilirse [7][3] ==0;
VardiyaSecilirse [7][4] ==0;
VardiyaSecilirse [7][5] ==0;
```

```
VardiyaSecilirse [7][6] ==0;  
VardiyaSecilirse [7][7] ==0;  
VardiyaSecilirse [7][8] ==0;  
VardiyaSecilirse [7][9] ==0; */
```

```
};
```

```
execute DISPLAY
```

```
{  
writeln("Fazlalik=",Fazlalik);  
writeln("TopFazlalik=",TopFazlalik);  
writeln("TopVarMaliyet=",TopVarMaliyet);  
writeln("TopKarsilanmayanMaliyet=",TopKarsilanmayanMaliyet);  
writeln("TopFazlalikMaliyet=",TopFazlalikMaliyet);  
}
```



EK B: Personel çizelgeleme problemi için geliştirilen TSP modeline (Model 2a) ait IBM ILOG CPLEX kodları

```

/*****
* OPL 12.6.0.0 Model
* Author: tturker & ademiriz
* Model 2a Kodları
*****/

```

```

//***DATA***

```

```

{int} birmolavard1=asSet(3..8);
{int} birmolavard2=asSet(35..40);
{int} birmolavard3=asSet(39..44);
{int} birmolavard4=asSet(43..48);
{int} birmolavard5=asSet(47..52);
{int} birmolavard6=asSet(51..56);
{int} birmolavard7=asSet(55..60);
{int} birmolavard8=asSet(59..64);
{int} birmolavard9=asSet(63..68);
{int} birmolavard10=asSet(67..72);

{int} yemmolavard1=asSet(11..17);
{int} yemmolavard2=asSet(43..49);
{int} yemmolavard3=asSet(47..53);
{int} yemmolavard4=asSet(51..57);
{int} yemmolavard5=asSet(55..61);
{int} yemmolavard6=asSet(59..65);
{int} yemmolavard7=asSet(63..69);
{int} yemmolavard8=asSet(67..73);
{int} yemmolavard9=asSet(71..77);
{int} yemmolavard10=asSet(75..81);

{int} ikimolavard1=asSet(21..26);
{int} ikimolavard2=asSet(53..58);
{int} ikimolavard3=asSet(57..62);
{int} ikimolavard4=asSet(61..66);
{int} ikimolavard5=asSet(65..70);
{int} ikimolavard6=asSet(69..74);
{int} ikimolavard7=asSet(73..78);
{int} ikimolavard8=asSet(77..82);
{int} ikimolavard9=asSet(81..86);
{int} ikimolavard10=asSet(85..90);

{int} ucmolavard1=asSet(27..32);
{int} ucmolavard2=asSet(59..64);
{int} ucmolavard3=asSet(63..68);
{int} ucmolavard4=asSet(67..72);

```

```
{int} ucmolavard5=asSet(71..76);
{int} ucmolavard6=asSet(75..80);
{int} ucmolavard7=asSet(79..84);
{int} ucmolavard8=asSet(83..88);
{int} ucmolavard9=asSet(87..92);
{int} ucmolavard10=asSet(91..96);
```

```
{string} Vardiyalar = ...;
```

```
int VardiyaSure [Vardiyalar] = ...;
int IsciMaliyet =...;
```

```
int Isci=...;
range Isciler = 1..Isci;
```

```
int Gun=...;
range Gunler = 1..Gun;
```

```
int Periyot=...;
range Periyotlar = 1..Periyot;
```

```
int PerBasIsGuculhtiyaci [1..Gun][1..Periyot] =...;
```

```
{string} PPeriyotFaalVarKume [Periyotlar] = ...;
```

```
range mola1rangevard1=1..card(birmolavard1);
range mola1rangevard2=1..card(birmolavard2);
range mola1rangevard3=1..card(birmolavard3);
range mola1rangevard4=1..card(birmolavard4);
range mola1rangevard5=1..card(birmolavard5);
range mola1rangevard6=1..card(birmolavard6);
range mola1rangevard7=1..card(birmolavard7);
range mola1rangevard8=1..card(birmolavard8);
range mola1rangevard9=1..card(birmolavard9);
range mola1rangevard10=1..card(birmolavard10);
```

```
range mola2rangevard1=1..card(ikimolavard1);
range mola2rangevard2=1..card(ikimolavard2);
range mola2rangevard3=1..card(ikimolavard3);
range mola2rangevard4=1..card(ikimolavard4);
range mola2rangevard5=1..card(ikimolavard5);
range mola2rangevard6=1..card(ikimolavard6);
range mola2rangevard7=1..card(ikimolavard7);
range mola2rangevard8=1..card(ikimolavard8);
range mola2rangevard9=1..card(ikimolavard9);
range mola2rangevard10=1..card(ikimolavard10);
```

```
range mola3rangevard1=1..card(ucmolavard1);
```

```

range mola3rangevard2=1..card(ucmolavard2);
range mola3rangevard3=1..card(ucmolavard3);
range mola3rangevard4=1..card(ucmolavard4);
range mola3rangevard5=1..card(ucmolavard5);
range mola3rangevard6=1..card(ucmolavard6);
range mola3rangevard7=1..card(ucmolavard7);
range mola3rangevard8=1..card(ucmolavard8);
range mola3rangevard9=1..card(ucmolavard9);
range mola3rangevard10=1..card(ucmolavard10);

```

```

range molayrangevard1=1..card(yemmolavard1);
range molayrangevard2=1..card(yemmolavard2);
range molayrangevard3=1..card(yemmolavard3);
range molayrangevard4=1..card(yemmolavard4);
range molayrangevard5=1..card(yemmolavard5);
range molayrangevard6=1..card(yemmolavard6);
range molayrangevard7=1..card(yemmolavard7);
range molayrangevard8=1..card(yemmolavard8);
range molayrangevard9=1..card(yemmolavard9);
range molayrangevard10=1..card(yemmolavard10);

```

```

int VarAtananIsGucuSayisi [i in Gunler][m in Vardiyalar]=...;

```

```

/**DECISION VARIABLES**

```

```

dvar boolean IsciVarAtanmasi [Isciler][Gunler][Vardiyalar];

```

```

dvar boolean IsciBirMolavard1 [Isciler][Gunler][mola1rangevard1];
dvar boolean IsciBirMolavard2 [Isciler][Gunler][mola1rangevard2];
dvar boolean IsciBirMolavard3 [Isciler][Gunler][mola1rangevard3];
dvar boolean IsciBirMolavard4 [Isciler][Gunler][mola1rangevard4];
dvar boolean IsciBirMolavard5 [Isciler][Gunler][mola1rangevard5];
dvar boolean IsciBirMolavard6 [Isciler][Gunler][mola1rangevard6];
dvar boolean IsciBirMolavard7 [Isciler][Gunler][mola1rangevard7];
dvar boolean IsciBirMolavard8 [Isciler][Gunler][mola1rangevard8];
dvar boolean IsciBirMolavard9 [Isciler][Gunler][mola1rangevard9];
dvar boolean IsciBirMolavard10 [Isciler][Gunler][mola1rangevard10];

```

```

dvar boolean IsciYemMolavard1 [Isciler][Gunler][molayrangevard1];
dvar boolean IsciYemMolavard2 [Isciler][Gunler][molayrangevard2];
dvar boolean IsciYemMolavard3 [Isciler][Gunler][molayrangevard3];
dvar boolean IsciYemMolavard4 [Isciler][Gunler][molayrangevard4];
dvar boolean IsciYemMolavard5 [Isciler][Gunler][molayrangevard5];
dvar boolean IsciYemMolavard6 [Isciler][Gunler][molayrangevard6];
dvar boolean IsciYemMolavard7 [Isciler][Gunler][molayrangevard7];
dvar boolean IsciYemMolavard8 [Isciler][Gunler][molayrangevard8];
dvar boolean IsciYemMolavard9 [Isciler][Gunler][molayrangevard9];

```

```
dvar boolean IsciYemMolavard10 [Isciler][Gunler][mola1rangevard10];
```

```
dvar boolean IsciIkiMolavard1 [Isciler][Gunler][mola2rangevard1];
dvar boolean IsciIkiMolavard2 [Isciler][Gunler][mola2rangevard2];
dvar boolean IsciIkiMolavard3 [Isciler][Gunler][mola2rangevard3];
dvar boolean IsciIkiMolavard4 [Isciler][Gunler][mola2rangevard4];
dvar boolean IsciIkiMolavard5 [Isciler][Gunler][mola2rangevard5];
dvar boolean IsciIkiMolavard6 [Isciler][Gunler][mola2rangevard6];
dvar boolean IsciIkiMolavard7 [Isciler][Gunler][mola2rangevard7];
dvar boolean IsciIkiMolavard8 [Isciler][Gunler][mola2rangevard8];
dvar boolean IsciIkiMolavard9 [Isciler][Gunler][mola2rangevard9];
dvar boolean IsciIkiMolavard10 [Isciler][Gunler][mola2rangevard10];
```

```
dvar boolean IsciUcMolavard1 [Isciler][Gunler][mola3rangevard1];
dvar boolean IsciUcMolavard2 [Isciler][Gunler][mola3rangevard2];
dvar boolean IsciUcMolavard3 [Isciler][Gunler][mola3rangevard3];
dvar boolean IsciUcMolavard4 [Isciler][Gunler][mola3rangevard4];
dvar boolean IsciUcMolavard5 [Isciler][Gunler][mola3rangevard5];
dvar boolean IsciUcMolavard6 [Isciler][Gunler][mola3rangevard6];
dvar boolean IsciUcMolavard7 [Isciler][Gunler][mola3rangevard7];
dvar boolean IsciUcMolavard8 [Isciler][Gunler][mola3rangevard8];
dvar boolean IsciUcMolavard9 [Isciler][Gunler][mola3rangevard9];
dvar boolean IsciUcMolavard10 [Isciler][Gunler][mola3rangevard10];
```

```
/***/DECISION EXPRESSIONS****/
```

```
dexpr int IsciAtanmasiGun [j in Isciler][i in Gunler]=sum(m in Vardiyalar)
IsciVarAtanmasi [j][i][m]; //gerek olmayabilir ya da decision expression olabilir
```

```
dexpr int VarAtananIsGucuSayisi_asil [i in Gunler][m in Vardiyalar]=
sum(j in Isciler) IsciVarAtanmasi [j][i][m];
```

```
dexpr int VarAtananIsGucuSayisi_ [m in Vardiyalar][i in Gunler]=
sum(j in Isciler) IsciVarAtanmasi [j][i][m];
```

```
dexpr int GunAtananIsGucuSayisi [i in Gunler]=
sum(j in Isciler, m in Vardiyalar) IsciVarAtanmasi [j][i][m];
```

```
dexpr int PPeriyotBirinciMolaVerenSayisi [i in Gunler][p in Periyotlar]
=max1(0,sum(j in Isciler,m in mola1rangevard1:item(birmolavard1,m-1)==p)
IsciBirMolavard1[j][i][m])+
max1(0,sum(j in Isciler,m in mola1rangevard2:item(birmolavard2,m-1)==p)
IsciBirMolavard2[j][i][m])+
max1(0,sum(j in Isciler,m in mola1rangevard3:item(birmolavard3,m-1)==p)
IsciBirMolavard3[j][i][m])+
max1(0,sum(j in Isciler,m in mola1rangevard4:item(birmolavard4,m-1)==p)
IsciBirMolavard4[j][i][m])+
```

```

    maxl(0,sum(j in Isciler,m in mola1rangevard5:item(birmolavard5,m-1)==p)
IsciBirMolavard5[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola1rangevard6:item(birmolavard6,m-1)==p)
IsciBirMolavard6[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola1rangevard7:item(birmolavard7,m-1)==p)
IsciBirMolavard7[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola1rangevard8:item(birmolavard8,m-1)==p)
IsciBirMolavard8[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola1rangevard9:item(birmolavard9,m-1)==p)
IsciBirMolavard9[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola1rangevard10:item(birmolavard10,m-1)==p)
IsciBirMolavard10[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var1 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard1:item(birmolavard1,m-1)==p)
IsciBirMolavard1[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var2 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard2:item(birmolavard2,m-1)==p)
IsciBirMolavard2[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var3 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard3:item(birmolavard3,m-1)==p)
IsciBirMolavard3[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var4 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard4:item(birmolavard4,m-1)==p)
IsciBirMolavard4[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var5 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard5:item(birmolavard5,m-1)==p)
IsciBirMolavard5[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var6 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard6:item(birmolavard6,m-1)==p)
IsciBirMolavard6[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var7 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard7:item(birmolavard7,m-1)==p)
IsciBirMolavard7[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var8 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard8:item(birmolavard8,m-1)==p)
IsciBirMolavard8[j][i][m]);

```

```

dexpr int PPeriyotBirinciMolaVerenSayisi_Var9 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard9:item(birmolavard9,m-1)==p)
IsciBirMolavard9[j][i][m]);

```

```
dexpr int PPeriyotBirinciMolaVerenSayisi_Var10 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola1rangevard10:item(birmolavard10,m-1)==p)
IsciBirMolavard10[j][i][m]);
```

```
dexpr int IsciBirMolaPeriyotSayisi [j in Isciler] [i in Gunler]
=maxl(0,sum(m in mola1rangevard1) IsciBirMolavard1[j][i][m])+
maxl(0,sum(m in mola1rangevard2) IsciBirMolavard2[j][i][m])+
maxl(0,sum(m in mola1rangevard3) IsciBirMolavard3[j][i][m])+
maxl(0,sum(m in mola1rangevard4) IsciBirMolavard4[j][i][m])+
maxl(0,sum(m in mola1rangevard6) IsciBirMolavard6[j][i][m])+
maxl(0,sum(m in mola1rangevard7) IsciBirMolavard7[j][i][m])+
maxl(0,sum(m in mola1rangevard8) IsciBirMolavard8[j][i][m])+
maxl(0,sum(m in mola1rangevard9) IsciBirMolavard9[j][i][m])+
maxl(0,sum(m in mola1rangevard10) IsciBirMolavard10[j][i][m]);
```

```
dexpr int PPeriyotYemekMolaVerenSayisi [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in molayrangevard1:item(yemmolavard1,m-1)==p)
IsciYemMolavard1[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard1:item(yemmolavard1,m-1)==p-1)
IsciYemMolavard1[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard2:item(yemmolavard2,m-1)==p)
IsciYemMolavard2[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard2:item(yemmolavard2,m-1)==p-1)
IsciYemMolavard2[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard3:item(yemmolavard3,m-1)==p)
IsciYemMolavard3[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard3:item(yemmolavard3,m-1)==p-1)
IsciYemMolavard3[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard4:item(yemmolavard4,m-1)==p)
IsciYemMolavard4[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard4:item(yemmolavard4,m-1)==p-1)
IsciYemMolavard4[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard5:item(yemmolavard5,m-1)==p)
IsciYemMolavard5[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard5:item(yemmolavard5,m-1)==p-1)
IsciYemMolavard5[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard6:item(yemmolavard6,m-1)==p)
IsciYemMolavard6[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard6:item(yemmolavard6,m-1)==p-1)
IsciYemMolavard6[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard7:item(yemmolavard7,m-1)==p)
IsciYemMolavard7[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard7:item(yemmolavard7,m-1)==p-1)
IsciYemMolavard7[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard8:item(yemmolavard8,m-1)==p)
IsciYemMolavard8[j][i][m])+
maxl(0,sum(j in Isciler,m in molayrangevard8:item(yemmolavard8,m-1)==p-1)
IsciYemMolavard8[j][i][m])+
```

```

    maxl(0,sum(j in Isciler,m in molayrangevard9:item(yemmolavard9,m-1)==p)
IsciYemMolavard9[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard9:item(yemmolavard9,m-1)==p-1)
IsciYemMolavard9[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard10:item(yemmolavard10,m-1)==p)
IsciYemMolavard10[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard10:item(yemmolavard10,m-1)==p-1)
IsciYemMolavard10[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var1 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard1:item(yemmolavard1,m-1)==p)
IsciYemMolavard1[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard1:item(yemmolavard1,m-1)==p-1)
IsciYemMolavard1[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var2 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard2:item(yemmolavard2,m-1)==p)
IsciYemMolavard2[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard2:item(yemmolavard2,m-1)==p-1)
IsciYemMolavard2[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var3 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard3:item(yemmolavard3,m-1)==p)
IsciYemMolavard3[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard3:item(yemmolavard3,m-1)==p-1)
IsciYemMolavard3[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var4 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard4:item(yemmolavard4,m-1)==p)
IsciYemMolavard4[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard4:item(yemmolavard4,m-1)==p-1)
IsciYemMolavard4[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var5 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard5:item(yemmolavard5,m-1)==p)
IsciYemMolavard5[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard5:item(yemmolavard5,m-1)==p-1)
IsciYemMolavard5[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var6 [i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard6:item(yemmolavard6,m-1)==p)
IsciYemMolavard6[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard6:item(yemmolavard6,m-1)==p-1)
IsciYemMolavard6[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var7[i in Gunler][p in Periyotlar]
= maxl(0,sum(j in Isciler,m in molayrangevard7:item(yemmolavard7,m-1)==p)
IsciYemMolavard7[j][i][m])+

```

```

    maxl(0,sum(j in Isciler,m in molayrangevard7:item(yemmolavard7,m-1)==p-1)
    IsciYemMolavard7[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var8[i in Gunler][p in Periyotlar]
    =maxl(0,sum(j in Isciler,m in molayrangevard8:item(yemmolavard8,m-1)==p)
    IsciYemMolavard8[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard8:item(yemmolavard8,m-1)==p-1)
    IsciYemMolavard8[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var9[i in Gunler][p in Periyotlar]
    =maxl(0,sum(j in Isciler,m in molayrangevard9:item(yemmolavard9,m-1)==p)
    IsciYemMolavard9[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard9:item(yemmolavard9,m-1)==p-1)
    IsciYemMolavard9[j][i][m]);

```

```

dexpr int PPeriyotYemekMolaVerenSayisi_Var10[i in Gunler][p in Periyotlar]
    =maxl(0,sum(j in Isciler,m in molayrangevard10:item(yemmolavard10,m-1)==p)
    IsciYemMolavard10[j][i][m])+
    maxl(0,sum(j in Isciler,m in molayrangevard10:item(yemmolavard10,m-1)==p-1)
    IsciYemMolavard10[j][i][m]);

```

```

dexpr int IsciYemMolaPeriyotSayisi [j in Isciler] [i in Gunler]
    =maxl(0,sum(m in molayrangevard1) IsciYemMolavard1[j][i][m])+
    maxl(0,sum(m in molayrangevard2) IsciYemMolavard2[j][i][m])+
    maxl(0,sum(m in molayrangevard3) IsciYemMolavard3[j][i][m])+
    maxl(0,sum(m in molayrangevard4) IsciYemMolavard4[j][i][m])+
    maxl(0,sum(m in molayrangevard5) IsciYemMolavard5[j][i][m])+
    maxl(0,sum(m in molayrangevard6) IsciYemMolavard6[j][i][m])+
    maxl(0,sum(m in molayrangevard7) IsciYemMolavard7[j][i][m])+
    maxl(0,sum(m in molayrangevard8) IsciYemMolavard8[j][i][m])+
    maxl(0,sum(m in molayrangevard9) IsciYemMolavard9[j][i][m])+
    maxl(0,sum(m in molayrangevard10) IsciYemMolavard10[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi [i in Gunler][p in Periyotlar]
    =maxl(0,sum(j in Isciler,m in mola2rangevard1:item(ikimolavard1,m-1)==p)
    IsciIkiMolavard1[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard2:item(ikimolavard2,m-1)==p)
    IsciIkiMolavard2[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard3:item(ikimolavard3,m-1)==p)
    IsciIkiMolavard3[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard4:item(ikimolavard4,m-1)==p)
    IsciIkiMolavard4[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard5:item(ikimolavard5,m-1)==p)
    IsciIkiMolavard5[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard6:item(ikimolavard6,m-1)==p)
    IsciIkiMolavard6[j][i][m])+

```

```

    maxl(0,sum(j in Isciler,m in mola2rangevard7:item(ikimolavard7,m-1)==p)
IsciIkiMolavard7[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard8:item(ikimolavard8,m-1)==p)
IsciIkiMolavard8[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard9:item(ikimolavard9,m-1)==p)
IsciIkiMolavard9[j][i][m])+
    maxl(0,sum(j in Isciler,m in mola2rangevard10:item(ikimolavard10,m-1)==p)
IsciIkiMolavard10[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var1 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard1:item(ikimolavard1,m-1)==p)
IsciIkiMolavard1[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var2 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard2:item(ikimolavard2,m-1)==p)
IsciIkiMolavard2[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var3 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard3:item(ikimolavard3,m-1)==p)
IsciIkiMolavard3[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var4 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard4:item(ikimolavard4,m-1)==p)
IsciIkiMolavard4[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var5 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard5:item(ikimolavard5,m-1)==p)
IsciIkiMolavard5[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var6 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard6:item(ikimolavard6,m-1)==p)
IsciIkiMolavard6[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var7 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard7:item(ikimolavard7,m-1)==p)
IsciIkiMolavard7[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var8 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard8:item(ikimolavard8,m-1)==p)
IsciIkiMolavard8[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var9 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard9:item(ikimolavard9,m-1)==p)
IsciIkiMolavard9[j][i][m]);

```

```

dexpr int PPeriyotIkinciMolaVerenSayisi_Var10 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola2rangevard10:item(ikimolavard10,m-1)==p)
IsciIkiMolavard10[j][i][m]);

```

```

dexpr int IsciIkiMolaPeriyotSayisi [j in Isciler] [i in Gunler]
=maxl(0,sum(m in mola2rangevard1) IsciIkiMolavard1[j][i][m])+
maxl(0,sum(m in mola2rangevard2) IsciIkiMolavard2[j][i][m])+
maxl(0,sum(m in mola2rangevard3) IsciIkiMolavard3[j][i][m])+
maxl(0,sum(m in mola2rangevard4) IsciIkiMolavard4[j][i][m])+
maxl(0,sum(m in mola2rangevard5) IsciIkiMolavard5[j][i][m])+
maxl(0,sum(m in mola2rangevard6) IsciIkiMolavard6[j][i][m])+
maxl(0,sum(m in mola2rangevard7) IsciIkiMolavard7[j][i][m])+
maxl(0,sum(m in mola2rangevard8) IsciIkiMolavard8[j][i][m])+
maxl(0,sum(m in mola2rangevard9) IsciIkiMolavard9[j][i][m])+
maxl(0,sum(m in mola2rangevard10) IsciIkiMolavard10[j][i][m]);

```

```

dexpr int PPeriyotUcuncuMolaVerenSayisi [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard1:item(ucmolavard1,m-1)==p)
IsciUcMolavard1[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard2:item(ucmolavard2,m-1)==p)
IsciUcMolavard2[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard3:item(ucmolavard3,m-1)==p)
IsciUcMolavard3[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard4:item(ucmolavard4,m-1)==p)
IsciUcMolavard4[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard5:item(ucmolavard5,m-1)==p)
IsciUcMolavard5[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard6:item(ucmolavard6,m-1)==p)
IsciUcMolavard6[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard7:item(ucmolavard7,m-1)==p)
IsciUcMolavard7[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard8:item(ucmolavard8,m-1)==p)
IsciUcMolavard8[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard9:item(ucmolavard9,m-1)==p)
IsciUcMolavard9[j][i][m])+
maxl(0,sum(j in Isciler,m in mola3rangevard10:item(ucmolavard10,m-1)==p)
IsciUcMolavard10[j][i][m]);

```

```

dexpr int PPeriyotUcuncuMolaVerenSayisi_Var1 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard1:item(ucmolavard1,m-1)==p)
IsciUcMolavard1[j][i][m]);

```

```

dexpr int PPeriyotUcuncuMolaVerenSayisi_Var2 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard2:item(ucmolavard2,m-1)==p)
IsciUcMolavard2[j][i][m]);

```

```

dexpr int PPeriyotUcuncuMolaVerenSayisi_Var3 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard3:item(ucmolavard3,m-1)==p)
IsciUcMolavard3[j][i][m]);

```

```

dexpr int PPeriyotUcuncuMolaVerenSayisi_Var4 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard4:item(ucmolavard4,m-1)==p)
IsciUcMolavard4[j][i][m]);

```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var5 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard5:item(ucmolavard5,m-1)==p)
IsciUcMolavard5[j][i][m]);
```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var6 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard6:item(ucmolavard6,m-1)==p)
IsciUcMolavard6[j][i][m]);
```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var7 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard7:item(ucmolavard7,m-1)==p)
IsciUcMolavard7[j][i][m]);
```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var8 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard8:item(ucmolavard8,m-1)==p)
IsciUcMolavard8[j][i][m]);
```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var9 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard9:item(ucmolavard9,m-1)==p)
IsciUcMolavard9[j][i][m]);
```

```
dexpr int PPeriyotUcuncuMolaVerenSayisi_Var10 [i in Gunler][p in Periyotlar]
=maxl(0,sum(j in Isciler,m in mola3rangevard10:item(ucmolavard10,m-1)==p)
IsciUcMolavard10[j][i][m]);
```

```
dexpr int IsciUcMolaPeriyotSayisi [j in Isciler][i in Gunler]
=maxl(0,sum(m in mola3rangevard1) IsciUcMolavard1[j][i][m])+
maxl(0,sum(m in mola3rangevard2) IsciUcMolavard2[j][i][m])+
maxl(0,sum(m in mola3rangevard3) IsciUcMolavard3[j][i][m])+
maxl(0,sum(m in mola3rangevard4) IsciUcMolavard4[j][i][m])+
maxl(0,sum(m in mola3rangevard5) IsciUcMolavard5[j][i][m])+
maxl(0,sum(m in mola3rangevard6) IsciUcMolavard6[j][i][m])+
maxl(0,sum(m in mola3rangevard7) IsciUcMolavard7[j][i][m])+
maxl(0,sum(m in mola3rangevard8) IsciUcMolavard8[j][i][m])+
maxl(0,sum(m in mola3rangevard9) IsciUcMolavard9[j][i][m])+
maxl(0,sum(m in mola3rangevard10) IsciUcMolavard10[j][i][m]);
```

```
dexpr int PPeriyotAtananSayisi_Toplam [i in Gunler][p in Periyotlar]
=sum(m in PPeriyotFaalVarKume[p]:card(PPeriyotFaalVarKume[p])>0)
VarAtananIsGucuSayisi[i][m];
```

```
dexpr int PPeriyotAtananSayisi_Var1 [i in Gunler][p in Periyotlar]
=sum(m in PPeriyotFaalVarKume[p]:m=="00-08") VarAtananIsGucuSayisi[i][m];
```

```
dexpr int PPeriyotAtananSayisi_Var2 [i in Gunler][p in Periyotlar]
=sum(m in PPeriyotFaalVarKume[p]:m=="08-16") VarAtananIsGucuSayisi[i][m];
```

```
dexpr int PPeriyotAtananSayisi_Var3 [i in Gunler][p in Periyotlar]
=sum(m in PPeriyotFaalVarKume[p]:m=="09-17") VarAtananIsGucuSayisi[i][m];
```

```

dexpr int PPeriyotAtananSayisi_Var4 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="10-18") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var5 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="11-19") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var6 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="12-20") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var7 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="13-21") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var8 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="14-22") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var9 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="15-23") VarAtananIsGucuSayisi[i][m];

dexpr int PPeriyotAtananSayisi_Var10 [i in Gunler][p in Periyotlar]
= sum(m in PPeriyotFaalVarKume[p]:m=="16-00") VarAtananIsGucuSayisi[i][m]; //
Bunları artık biliyoruz

dexpr int IsciTopSaat [j in Isciler]=sum (i in Gunler, m in Vardiyalar)
IsiVarAtanmasi [j][i][m] * VardiyaSure [m];

**** OBJECTIVE FUNCTION****

minimize sum (j in Isciler, i in Gunler, m in Vardiyalar)
(IsciMaliyet * VardiyaSure [m] * IsciVarAtanmasi [j][i][m]);

**** CONSTRAINTS****

subject to {

c1: // Çalışanların herhangi bir günde en fazla bir vardiyaya atanmalarına izin verir.

forall (j in Isciler, i in Gunler)
sum (m in Vardiyalar) IsciVarAtanmasi [j][i][m] <= 1;

c2: // Operatörlerin planlama ufkunda en az 1 gün tatil yapmalarını sağlar.

forall (j in Isciler) sum (m in Vardiyalar, i in Gunler)
IsiVarAtanmasi [j][i][m] <= (Gun-1);

c3: // Operatörlerin bütün dinlenme molalarını vermelerini sağlar . Ayrıca bu
molaların verildiği periyotları belirler.

```

```
forall (j in Isciler, i in Gunler )
```

```
{
  IsciBirMolaPeriyotSayisi [j] [i]<=1;
  IsciYemMolaPeriyotSayisi [j] [i]<=1;
  IsciIlkiMolaPeriyotSayisi [j] [i]<=1;
  IsciUcMolaPeriyotSayisi [j] [i]<=1;
}
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="00-08")
```

```
sum( m in mola1rangevard1) IsciBirMolavard1 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="08-16" )
```

```
sum( m in mola1rangevard2) IsciBirMolavard2 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="09-17")
```

```
sum( m in mola1rangevard3) IsciBirMolavard3 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="10-18")
```

```
sum( m in mola1rangevard4) IsciBirMolavard4 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="11-19")
```

```
sum( m in mola1rangevard5) IsciBirMolavard5 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="12-20")
```

```
sum( m in mola1rangevard6) IsciBirMolavard6 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="13-21")
```

```
sum( m in mola1rangevard7) IsciBirMolavard7 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="14-22")
```

```
sum( m in mola1rangevard8) IsciBirMolavard8 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="15-23")
```

```
sum( m in mola1rangevard9) IsciBirMolavard9 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="16-00")
```

```
sum( m in mola1rangevard10) IsciBirMolavard10 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="00-08")
```

```
sum( m in molayrangevard1) IsciYemMolavard1 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="08-16")
```

```
sum( m in molayrangevard2) IsciYemMolavard2 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="09-17")
```

```
sum( m in molayrangevard3) IsciYemMolavard3 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="10-18")
sum( m in molayrangevard4) IsciYemMolavard4 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="11-19")
sum( m in molayrangevard5) IsciYemMolavard5 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="12-20")
sum( m in molayrangevard6) IsciYemMolavard6 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="13-21")
sum( m in molayrangevard7) IsciYemMolavard7 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="14-22")
sum( m in molayrangevard8) IsciYemMolavard8 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="15-23")
sum( m in molayrangevard9) IsciYemMolavard9 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="16-00")
sum( m in molayrangevard10) IsciYemMolavard10 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="00-08")
sum( m in mola2rangevard1) IscilkiMolavard1 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="08-16")
sum( m in mola2rangevard2) IscilkiMolavard2 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="09-17")
sum( m in mola2rangevard3) IscilkiMolavard3 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="10-18")
sum( m in mola2rangevard4) IscilkiMolavard4 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="11-19")
sum( m in mola2rangevard5) IscilkiMolavard5 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="12-20")
sum( m in mola2rangevard6) IscilkiMolavard6 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="13-21")
sum( m in mola2rangevard7) IsciIlkiMolavard7 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="14-22")
sum( m in mola2rangevard8) IsciIlkiMolavard8 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="15-23")
sum( m in mola2rangevard9) IsciIlkiMolavard9 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="16-00")
sum( m in mola2rangevard10) IsciIlkiMolavard10 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="00-08")
sum( m in mola3rangevard1) IsciUcMolavard1 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="08-16")
sum( m in mola3rangevard2) IsciUcMolavard2 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="09-17")
sum( m in mola3rangevard3) IsciUcMolavard3 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="10-18")
sum( m in mola3rangevard4) IsciUcMolavard4 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="11-19")
sum( m in mola3rangevard5) IsciUcMolavard5 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="12-20")
sum( m in mola3rangevard6) IsciUcMolavard6 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="13-21")
sum( m in mola3rangevard7) IsciUcMolavard7 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="14-22")
sum( m in mola3rangevard8) IsciUcMolavard8 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="15-23")
sum( m in mola3rangevard9) IsciUcMolavard9 [j][i][m]==IsciVarAtanmasi [j][i][k];
```

```
forall (j in Isciler, i in Gunler, k in Vardiyalar : k=="16-00")
sum( m in mola3rangevard10) IsciUcMolavard10 [j][i][m]==IsciVarAtanmasi
[j][i][k];
```

c4: // İlgili gün ilgili vardiyaya atanacak operatör sayısını, vardiya çizelgeleme probleminin çözümü neticesinde elde edilen “vardiyalardaki gün başına operatör sayısına” eşitler.

```
forall (i in Gunler,m in Vardiyalar)
sum (j in Isciler) IsciVarAtanmasi [j][i][m] == VarAtananIsGucuSayisi [i][m];
```

c5: // Planlama dönemindeki günlerde tüm gün içi periyotlarda olması gereken kadar operatörün çalıştırılması sağlanır.

```
forall (i in Gunler, p in Periyotlar )
{
PPeriyotBirinciMolaVerenSayisi[i][p]>=0;
PPeriyotYemekMolaVerenSayisi[i][p]>=0;
PPeriyotIkinciMolaVerenSayisi[i][p]>=0;
PPeriyotUcuncuMolaVerenSayisi[i][p]>=0;

maxl (0, sum (m in PPeriyotFaalVarKume[p]:card (PPeriyotFaalVarKume[p])>0)
VarAtananIsGucuSayisi_asil[i][m])-PPeriyotBirinciMolaVerenSayisi[i][p]-
PPeriyotYemekMolaVerenSayisi[i][p]-PPeriyotIkinciMolaVerenSayisi[i][p]-
PPeriyotUcuncuMolaVerenSayisi[i][p])>=(PerBasIsGucuIhtiyaci[i][p]);
};
};
```

EK C: Personel çizelgeleme problemi için geliştirilen KP modeline (Model 2b) ait IBM ILOG CPLEX kodları

```

/*****
* OPL 12.6.0.0 Model
* Author: tturker & ademiriz
* Model 2b Kodları
*****/

```

```
using CP;
```

```
//***DATA***
```

```
int agent=...;
range agents = 1..agent;
```

```
int day=...;
range days=1..day;
```

```
int shift_ =...;
range shifts=1..shift_;
```

```
int period=...;
range periods=1..period;
```

```
int cost=...;
```

```
int req[i in days][p in periods]=...;
```

```
int nbassigned[i in days][m in shifts]=...;
```

```
int totassgn=sum(i in days,m in shifts) nbassigned[i][m];
range tottasks=1..(4*totassgn);
```

```
int startbreak1[shifts]=...;
int endbreak1[shifts]=...;
```

```
int startbreak2[shifts]=...;
int endbreak2[shifts]=...;
```

```
int startbreak3[shifts]=...;
int endbreak3[shifts]=...;
```

```
int startbreakM[shifts]=...;
int endbreakM[shifts]=...;
```

```
tuple shift{
int id;
```

```

int start;
int end;
};
{shift} dshifts=...;

tuple ashift{
int id;
int day;
int shift;
};
{ashift} ashifts=...;

tuple task {
key int id;
int day;
int type;
int shift;
int tsize;
int start;
int end;
};
{task} tasks = {<i,0,0,0,0,0,0> | i in tottasks};

execute INITIALIZE{
var j1=1;
for (var i in days){
  for(var s in shifts){
    for (var k=0;k<nbassigned[i][s];k++){
      Opl.item(tasks,j1-1).day=i;
      Opl.item(tasks,j1-1).type=1;
      Opl.item(tasks,j1-1).shift=s;
      Opl.item(tasks,j1-1).tsize=1;
      Opl.item(tasks,j1-1).start=startbreak1[s];
      Opl.item(tasks,j1-1).end=endbreak1[s];
      j1++;
    }
  }
}
var j2=1+totassgn;

for (var i in days){
  for(var s in shifts){
    for (var k=0;k<nbassigned[i][s];k++){
      Opl.item(tasks,j2-1).day=i;
      Opl.item(tasks,j2-1).type=2;
      Opl.item(tasks,j2-1).shift=s;
      Opl.item(tasks,j2-1).tsize=2;
      Opl.item(tasks,j2-1).start=startbreakM[s];
      Opl.item(tasks,j2-1).end=endbreakM[s];
      j2++;
    }
  }
}

```

```

    }}}
    var j3=1+2*totassgn;

    for (var i in days){
        for(var s in shifts){
            for (var k=0;k<nbassigned[i][s];k++){
                Opl.item(tasks,j3-1).day=i;
                Opl.item(tasks,j3-1).type=3;
                Opl.item(tasks,j3-1).shift=s;
                Opl.item(tasks,j3-1).tsize=1;
                Opl.item(tasks,j3-1).start=startbreak2[s];
                Opl.item(tasks,j3-1).end=endbreak2[s];
                j3++;
            }
        }
    }
    var j4=1+3*totassgn;

    for (var i in days){
        for(var s in shifts){
            for (var k=0;k<nbassigned[i][s];k++){
                Opl.item(tasks,j4-1).day=i;
                Opl.item(tasks,j4-1).type=4;
                Opl.item(tasks,j4-1).shift=s;
                Opl.item(tasks,j4-1).tsize=1;
                Opl.item(tasks,j4-1).start=startbreak3[s];
                Opl.item(tasks,j4-1).end=endbreak3[s];
                j4++;
            }
        }
    }
};

tuple Allocation {
    int taskid;
    int agent;
};
{Allocation} allocs =
    { <m.id, r> | m in tasks, r in agents };//

/**DECISION VARIABLES**/

dvar interval actworkers[a in ashifts][r in agents] optional in
item(dshifts,a.shift-1).start..item(dshifts,a.shift-1).end size
(item(dshifts,a.shift-1).end-item(dshifts,a.shift-1).start);

dvar interval titasks[m in tasks] in m.start..m.end size m.tsize;
dvar interval tiallocs[a in allocs] optional;
dvar sequence workers[i in ashifts][r in agents] in
all (a in allocs, t in tasks: a.agent == r && t.id==a.taskid && t.day==i.day &&
t.shift==i.shift) tiallocs[a];

```

```
/**DECISION EXPRESSIONS**/
```

```
cumulFunction assigned [i in days] =
    sum(r in agents,a in ashifts:a.day==i) pulse(actworkers[a][r],1);
```

```
cumulFunction breaks[i in days] =
    sum(t in tasks :t.day==i) pulse(ttasks[t], 1);
```

```
cumulFunction reqs [i in days] =
    sum(p in periods) pulse(p-1,p,req[i][p]);
```

```
/** SETTINGS **/
```

```
execute {var f = cp.factory;
    var phase =f.searchPhase(actworkers);
    cp.setSearchPhases (phase)}
```

```
/** OBJECTIVE FUNCTION **/
```

```
minimize sum (i in ashifts,r in agents) cost*lengthOf(actworkers[i][r]);
```

```
/** CONSTRAINTS **/
```

```
subject to {
```

c1: // İlgili gün ilgili vardiyaya atanacak operatör sayısını, vardiya çizelgeleme probleminin çözümü neticesinde elde edilen gün başına vardiyalardaki operatör sayısına eşitler.

```
forall(i in ashifts)
    sum(r in agents)(presenceOf(actworkers[i][r]))==nbassigned[i.day][i.shift];
```

c2: // Operatörlerin planlama ufkunda herhangi bir günde en fazla bir vardiyaya atanmalarına izin verilmektedir.

```
forall(r in agents,i in days)
    sum(a in ashifts:a.day==i)(presenceOf(actworkers[a][r]))<=1;
```

c3: // Operatörlerin planlama ufkunda en az 1 gün tatil yapmalarına olanak sağlamaktadır.

```
forall(r in agents)
    sum(a in ashifts)(presenceOf(actworkers[a][r]))<=day-1;
```

c4: // Operatörlerin atandıkları vardiyadaki her bir görevden (moladan) sadece bir tane yapmasını garantiler (Operatörlerin atandıkları vardiyada istenen tüm molalarını vermeleri garantilenmiş olur).

```
forall(r in agents,i in ashifts)
  sum(m in tasks, a in allocs:m.id==
    a.taskid&&a.agent==r && m.shift==i.shift&&m.day==i.day&&m.type==1)
  presenceOf(tiallocs[a])==(presenceOf(actworkers[i][r]));
```

```
forall(r in agents,i in ashifts)
  sum(m in tasks, a in allocs:m.id==
    a.taskid&&a.agent==r && m.shift==i.shift&&m.day==i.day&&m.type==2)
  presenceOf(tiallocs[a])==(presenceOf(actworkers[i][r]));
```

```
forall(r in agents,i in ashifts)
  sum(m in tasks, a in allocs:m.id==
    a.taskid&&a.agent==r && m.shift==i.shift&&m.day==i.day&&m.type==3)
  presenceOf(tiallocs[a])==(presenceOf(actworkers[i][r]));
```

```
forall(r in agents,i in ashifts)
  sum(m in tasks, a in allocs:m.id==
    a.taskid&&a.agent==r && m.shift==i.shift&&m.day==i.day&&m.type==4)
  presenceOf(tiallocs[a])==(presenceOf(actworkers[i][r]));
```

c5: // Her bir görevin sadece bir operatör tarafından yapılmasını sağlar.

```
forall(m in tasks)
  presenceOf(titasks[m])==sum(a in allocs: m.id==a.taskid) presenceOf(tiallocs[a]);
```

c6: // Her bir görevin sadece bir operatör tarafından yapılmasını sağlar.

```
forall(m in tasks)
  synchronize(titasks[m],all(a in allocs: m.id==a.taskid) tiallocs[a]);
```

c7: //Bir operatörün görev aldığı (molada olduğu) zaman aralığında aynı zaman dilimine gelebilecek başka bir görevde kullanılması engellenmiş olur.

```
forall(i in ashifts,r in agents)
  noOverlap(workers[i][r]);
```

c8: // Planlama dönemindeki günlerin gün içi periyotlarında yeterli sayıda operatörün çalıştırılmasını sağlar.

```
forall (i in days)
  assigned[i]-breaks[i]-reqs[i]>=0;
```

```
};
```

EK D: Kıyaslama test örnekleri için geliştirilen KP modeline (Model 3b) ait IBM ILOG CPLEX kodları

```

/*****
* OPL 12.6.0.0 Model
* Author: tturker & ademiriz
* Model 3b Kodları
*****/

```

```
using CP;
```

```
//***DATA***
```

```
{string} agents = ...;
```

```
int day=...;
range days=0..day;
```

```
int week=...;
range weeks=1..week;
```

```
{string} shift=...;
{string} nextforbid [shift]=...;
```

```
int start [shift]=...;
int end [shift]=...;
```

```
tuple req{
int day;
string shift;
int req;
};
{req} reqs=...;
```

```
tuple task {
key int id;
int day;
string shift;
int start;
int end;
int tsize;
};
{task} tasks =...;
```

```
tuple onrequest {
string agent;
int day ;
string shift;
```

```

int weight;
};
{onrequest} onrequests = ...;

tuple offrequest {
string agent;
int day ;
string shift;
int weight;
};
{offrequest} offrequests = ...;

tuple Allocation {
int taskid;
string agent;
};
{Allocation} allocs =
{ <m.id, r> | m in tasks, r in agents};

{int} day_off [r in agents]=...;

int maxworktime [r in agents]=...;
int minworktime [r in agents]=...;
int maxshift [r in agents][m in shift]=...;
int maxconsecutiveshifts [r in agents]=...;
int minconsecutiveshifts [r in agents]=...;
int minconsecutivedaysoff [r in agents]=...;
int maxweekend [r in agents]=...;

int reqsTASK [t in tasks] =
sum(a in reqs: a.day==t.day && a.shift==t.shift) (a.req);

/***/DECISION VARIABLES****

dvar int+ vbelow [t in tasks];
dvar int+ vabove [t in tasks];

dvar boolean actworkers[t in tasks][r in agents];
dvar boolean weekendwork [r in agents][w in weeks];

dvar interval titasks[t in tasks] in t.start..t.end size t.size;
dvar interval tiallocs[a in allocs] optional;

dvar sequence workers[r in agents][t in tasks] in
all(a in allocs: a.agent == r && t.id==a.taskid) tiallocs[a];

```

/**DECISION EXPRESSIONS**/

```
dexpr int assigned [t in tasks]=sum (r in agents) actworkers[t][r];
dexpr int Aassigned [t in tasks]=sum (r in agents,a in allocs: a.agent==r &&
t.id==a.taskid) presenceOf(tiallocs[a]);
dexpr int below[t in tasks]=Aassigned[t]-reqsTASK [t];
```

```
cumulFunction Fassigned [t in tasks]=sum(a in allocs:t.id==a.taskid)
pulse(tiallocs[a],1);
dexpr int Fassigned_[t in tasks]=sum(a in allocs:t.id==a.taskid)
heightAtStart(tiallocs[a],Fassigned[t]);
```

```
cumulFunction FreqsTASK [t in tasks]=sum(a in reqs: t.day==a.day &&
t.shift==a.shift) pulse(titasks[t],a.req);
dexpr int FreqsTASK_[t in tasks]=sum(a in reqs: t.day==a.day && t.shift==a.shift)
heightAtStart(titasks[t],FreqsTASK[t]);
```

```
cumulFunction Fmaxshift [r in agents][s in shift]= pulse(start[s],
end[s],maxshift[r][s]);
```

```
cumulFunction assignRtoD [r in agents][i in days]=
sum(t in tasks,a in allocs:t.day==i &&a.agent==r &&
t.id==a.taskid)stepAtStart(tiallocs[a],1);
dexpr int VassignRtoD [r in agents][i in days]=
sum(t in tasks,a in allocs:t.day==i &&a.agent==r &&
t.id==a.taskid)heightAtStart(tiallocs[a],assignRtoD[r][i]);
```

```
cumulFunction assignRtoDtoS [r in agents][i in days][s in shift]=
sum(t in tasks,a in allocs:t.shift==s && t.day==i && a.agent==r &&
t.id==a.taskid)stepAtStart(tiallocs[a],1);
```

```
cumulFunction maxshiftF [r in agents][s in shift]=
sum(t in tasks,a in allocs: t.shift==s && a.agent==r && t.id==a.taskid)
stepAtStart (tiallocs[a],1);
```

```
cumulFunction topworktimeF [r in agents]= sum(a in allocs:
a.agent==r)stepAtStart(tiallocs[a],60*lengthOf(tiallocs[a]));
dexpr int topworktimeF_ [r in agents]= sum(a in allocs:
a.agent==r)(heightAtStart(tiallocs[a],topworktimeF[r]));
```

```
dexpr int topworktimeA [r in agents]= sum(a in allocs:
a.agent==r)lengthOf(tiallocs[a])*60;
dexpr int topworktime [r in agents]= sum(t in
tasks)(lengthOf(titasks[t])*60*actworkers[t][r]);
```

```
int penalty_not_assign [r in agents][i in days][m in shift]=sum (o in onrequests:
o.agent==r && o.day==i && o.shift==m) (o.weight);
```

```

int penalty_assign [r in agents][i in days][m in shift]=sum (o in offrequests:
o.agent==r && o.day==i && o.shift==m) (o.weight);

dexpr int sum_penalty_not_assign= sum (r in agents,i in days,m in shift, t in
tasks:t.shift==m && t.day==i) (penalty_not_assign [r][i][m]*(1-actworkers[t][r]));

dexpr int sum_penalty_assign= sum (r in agents,i in days,m in shift, t in tasks:
t.shift==m && t.day==i) (penalty_assign [r][i][m]*actworkers[t][r]);

dexpr int sum_below=sum (t in tasks) (100*vbelow[t]);
dexpr int sum_above=sum (t in tasks) (1* vabove[t]);

dexpr int sum_penalty_not_assignR [r in agents][i in days]= sum (m in shift, t in
tasks :t.shift==m && t.day==i) (penalty_not_assign [r][i][m]*(1-(actworkers[t][r])));
dexpr int sum_penalty_assignR[r in agents][i in days]= sum (m in shift, t in tasks
:t.shift==m && t.day==i) (penalty_assign [r][i][m]*((actworkers[t][r])));

dexpr int jISCI_Pen_Assign [r in agents] [ i in days]=sum (t in tasks, a in allocs, m in
shift:t.shift==m && t.day==i && a.agent==r && t.id==a.taskid)(penalty_assign
[r][i][m]*(presenceOf(tiallocs[a])));
dexpr int jISCI_Pen_NOT_Assign [r in agents] [ i in days]=sum (t in tasks, a in
allocs, m in shift:t.shift==m && t.day==i && a.agent==r && t.id==a.taskid)
(penalty_not_assign [r][i][m]*(1-presenceOf(tiallocs[a])));

**** SETTINGS****

execute {var f = cp.factory;
    var phase1 =f.searchPhase(actworkers);
    var phase2 =f.searchPhase(tiallocs);
    cp.setSearchPhases (phase1,phase2);
}

**** OBJECTIVE FUNCTION****

minimize
    sum_penalty_not_assign+
    sum_penalty_assign+
    sum (t in tasks) (100*vbelow[t])+
    sum (t in tasks) (1 *vabove[t]);

**** CONSTRAINTS****

subject to {

c1: //An employee cannot be assigned more than one shift on a single day.

```

```
forall(r in agents, i in days)
    assignRtoD[r][i]<=1;
```

c2://Shift rotation. A minimum amount of rest is required after each shift. Therefore certain shifts cannot follow others. For example, an early shift cannot follow a late shift.

```
forall(r in agents, i in 0..day-1, s in shift, k in nextforbid[s])
    assignRtoDtoS [r][i][s]+assignRtoDtoS [r][i+1][k]<=1;
```

c3://The maximum numbers of shifts of each type that can be assigned to employees. For example, some employees will have contracts which do not allow them to work night shifts or only a maximum number of night shifts.

```
forall(r in agents, m in shift)
    maxshiftF [r][m]<=maxshift [r][m];
```

c4://Minimum and maximum work time. The total minutes worked by each employee must be between a minimum and maximum.

```
forall(r in agents)
    {minworktime [r]<= topworktime [r];
    maxworktime [r]>= topworktime [r];}
```

c5://Maximum consecutive shifts. The maximum number of shifts an employee can work without a day off.

```
forall(r in agents, i in 0..(day-maxconsecutiveshifts[r]))
    count(all( j in (i..(i+maxconsecutiveshifts[r])), t in tasks: t.day==j) actworkers
[t][r],1) <=maxconsecutiveshifts[r];
```

c6://Minimum consecutive shifts. This can be modelled by preventing every sequence of consecutive shifts below the minimum. For example, if the minimum number of consecutive shifts is four then we must not allow any of the sequences: {off-on-off, off-on-on-off, off-on-on-on-off} where off is a day without a shift and on is a day with a shift assigned.

```
forall(r in agents, s in 1..(minconsecutiveshifts[r]-1), i in 0..(day-(s+1)))
    count(all( t in tasks: t.day==i)actworkers[t][r],1)+
    s+
    count(all( t in tasks: t.day==(i+s+1))actworkers[t][r],1)>
    count(all( j in ((i+1)..(i+s)), t in tasks: t.day==j)actworkers[t][r],1);
```

c7://Minimum consecutive days off. This can be modelled in a similar way to the minimum consecutive shifts constraint. For example, if the minimum number of consecutive days off is three then we must not allow any of the sequences: {on-off-on, on-off-off-on}.

```
forall(r in agents,s in 1..(minconsecutivedaysoff[r]-1), i in 0..(day-(s+1)))
```

```
(1-count(all( t in tasks: t.day==i)actworkers[t][r],1))+
count(all( j in ((i+1)..(i+s)),t in tasks: t.day==j)actworkers[t][r],1)+
(1-count(all( t in tasks: t.day==(i+s+1))actworkers[t][r],1))>0;
```

c8://Maximum number of weekends. A weekend is considered as being worked if the employee has a shift on the Saturday or the Sunday.

```
forall(r in agents, w in weeks)
    weekendwork[r][w]==maxl(sum( t in tasks: t.day==(7*w-2))actworkers[t][r],
        sum( t in tasks: t.day==(7*w-1))actworkers[t][r]);
```

```
forall(r in agents)
    sum( w in weeks)weekendwork[r][w]<=maxweekend[r];
```

c9://Days off. These are days that employees cannot work because, for example, they are on vacation.

```
forall(r in agents, k in day_off[r])
    sum( t in tasks, a in allocs :t.day==k && a.agent==r && t.id==a.taskid)
    presenceOf(tiallocs[a])==0;
```

c10://Cover requirements.

```
forall (t in tasks)
    (Assigned [t]-reqsTASK[t])==vabove[t]-vbelow[t];
```

c11://

```
forall(r in agents, t in tasks,a in allocs: t.id==a.taskid && a.agent==r)
    presenceOf(tiallocs[a])==(actworkers[t][r]);
```

```
forall(r in agents, t in tasks,a in allocs: t.id==a.taskid && a.agent==r)
    lengthOf(tiallocs[a])==lengthOf(titasks[t])*(actworkers[t][r]);
```

c12://

```
forall(t in tasks)
    synchronize(titasks[t],all(a in allocs: t.id==a.taskid) tiallocs[a]);
```

c13://

```
forall(r in agents, t in tasks)
    noOverlap(workers[r][t]);
```

```
};
```

EK E: Mevcut durumun Model 1 ile çözülmesi sonucu ortaya çıkan IBM ILOG CPLEX sonuçları

*** KOMUT DOSYASI GÜNLÜĞÜ ***

// solution (optimal) with objective 15501
Solving time ~= 4.87

Fazlalik= [[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 3 3 3 3 1 1 1
1 9 9 9 9 8 8 7 7 6 6 6 6 0 0 0 0 1 1 1 2 2 3 3 3 5 6 7 7 7 7
8 8 1 2 2 2 4 4 5 7 0 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4]
[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 5
5 5 5 3 3 3 3 10 10 10 10 9 9 8 8 6 6 6 6 0 0 0 0 1 1 1 2 2 3 3 3
3 4 5 5 5 5 6 6 0 1 1 1 3 3 4 5 0 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4]
[0 0 2 2 2 3 3 3 3 3 3 3 4 4 4 4 5 5 5 5 4 4 4 4 3 3 3 3 3 3 3 3
3 3 3 3 3 3 9 9 9 9 1 2 0 0 4 4 3 3 3 5 5 6 7 7 5 7 8 8 8 8 7 7
7 7 7 7 7 7 3 3 3 3 4 6 7 9 0 0 0 0 1 1 1 1 3 3 3 3 4 4 4 4]
[0 0 1 1 1 1 1 1 2 2 2 2 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 1 1 1 1 5
5 5 5 3 3 3 3 7 6 6 6 5 5 5 5 1 1 0 0 0 0 1 1 0 0 1 1 2 1 0 0 1 0
1 1 2 3 1 3 0 0 0 0 2 3 5 5 0 0 1 1 2 2 2 2 2 2 3 3 3 4 4 4]
[0 0 0 0 0 0 0 0 1 1 1 2 2 2 2 3 3 3 3 3 1 1 1 1 0 0 0 0 0 0 0 0 2
1 0 0 0 0 1 0 8 6 7 6 8 6 6 6 8 1 1 1 4 4 4 4 5 5 5 5 5 5 5 8 8
8 8 9 11 13 14 3 5 7 8 3 5 7 8 0 0 0 0 1 1 1 1 2 2 2 2 4 4 4 4]
[0 0 0 0 0 1 1 1 1 1 2 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 0 0 0 1 0
10 10 9 9 9 9 10 10 9 10 9 9 9 9 9 2 2 1 1 1 0 0 0 0 2 1 2 1 2 2 1
2 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 6 6 6 6 7 7 7 7 9 9 9 9 9 9 9]
[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 5
5 5 4 4 4 4 3 3 2 2 2 2 2 2 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 1 1 1 1 2 2 4 4 0 0 0 0 2 2 2 2 3 4 4 5 5 6 6 6]]

TopFazlalik=2067

TopVarMaliyet=9300

TopKarsilanmayanMaliyet=0

TopFazlalikMaliyet=6201

*** ÇÖZÜMLER ***

// solution (optimal) with objective 15501

// Quality Incumbent solution:

// MILP objective 1.5501000000e+004
// MILP solution norm |x| (Total, Max) 2.44940e+004 2.38470e+004
// MILP solution error (Ax=b) (Total, Max) 0.00000e+000 0.00000e+000
// MILP x bound error (Total, Max) 0.00000e+000 0.00000e+000
// MILP x integrality error (Total, Max) 0.00000e+000 0.00000e+000
// MILP slack bound error (Total, Max) 0.00000e+000 0.00000e+000
// MILP indicator slack bound error (Total, Max) 0.00000e+000 0.00000e+000

EK F: Önerilen durumun Model 1 ile çözülmesi sonucu ortaya çıkan IBM ILOG CPLEX sonuçları

**** KOMUT DOSYASI GÜNLÜĞÜ****

// solution (optimal) with objective 13437
Solving time ~= 4.95

Fazlalik= [[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 8 8 8 8 6
6 6 6 5 5 5 4 4 3 3 6 6 6 6 0 0 0 1 1 1 2 2 3 3 3 0 1 2 2 2 2
3 3 5 6 6 6 8 8 9 1 1 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4]
[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 1 1 1 1 8
8 8 8 6 6 6 5 5 5 4 4 3 3 6 6 6 6 0 0 0 1 1 1 2 2 3 3 3 0 1
2 2 2 2 3 3 5 6 6 6 8 8 9 1 0 0 0 0 1 1 1 2 2 2 3 3 3 4 4]
[0 0 2 2 2 3 3 3 3 3 3 4 4 4 4 5 5 5 5 4 4 4 3 3 3 3 3 3 3 8
8 8 8 8 8 8 7 7 7 7 3 4 2 2 1 1 0 0 0 2 2 3 4 4 2 4 5 5 5 0 0
0 0 0 0 0 3 3 3 3 0 2 3 5 1 1 1 1 2 2 2 2 4 4 4 4 5 5 5 5]
[0 0 1 1 1 1 1 1 2 2 2 2 4 4 4 4 4 4 4 4 4 4 2 2 2 2 1 1 1 1 5
5 5 5 3 3 3 3 2 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 1 1 2 1 0 0 1 0
1 1 2 3 1 3 5 5 5 5 7 8 10 10 0 0 1 1 2 2 2 2 2 3 3 3 4 4 4]
[0 0 0 0 0 0 0 1 1 1 2 2 2 2 3 3 3 3 3 1 1 1 1 0 0 0 0 0 0 0 8
7 6 6 6 6 7 6 8 6 7 6 8 6 6 6 7 0 0 0 3 3 3 3 4 4 4 4 4 4 1 1
1 1 2 4 6 7 2 4 6 7 2 4 6 7 0 0 0 1 1 1 1 2 2 2 2 4 4 4 4]
[0 0 0 0 0 1 1 1 1 1 2 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 0 0 0 4
4 4 3 3 3 3 4 4 3 4 3 3 3 3 3 2 2 1 1 1 0 0 0 0 2 1 2 1 2 1 2 1
1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 3 3 3 3 3 3 3 3]
[0 0 1 1 1 1 1 1 3 3 3 3 3 3 3 4 4 4 4 3 3 3 3 3 3 3 3 1 1 1 1 5
5 5 4 4 4 4 3 3 2 2 2 2 2 2 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 1 1 1 1 2 2 4 4 0 0 0 0 2 2 2 2 3 4 4 5 5 6 6 6 6]]

TopFazlalik=1779

TopVarMaliyet=8100

TopKarsilanmayanMaliyet=0

TopFazlalikMaliyet=5337

**** ÇÖZÜMLER****

// solution (optimal) with objective 13437

// Quality Incumbent solution:

// MILP objective 1.3437000000e+004

// MILP solution norm |x| (Total, Max) 2.44690e+004 2.38470e+004

// MILP solution error (Ax=b) (Total, Max) 5.68434e-014 1.77636e-015

// MILP x bound error (Total, Max) 0.00000e+000 0.00000e+000

// MILP x integrality error (Total, Max) 1.77636e-015 1.77636e-015

// MILP slack bound error (Total, Max) 0.00000e+000 0.00000e+000

// MILP indicator slack bound error (Total, Max) 0.00000e+000 0.00000e+000

Root node processing (before b&c):

Real time = 43.38 sec. (104814.92 ticks)

Parallel b&c, 4 threads:

Real time = 0.00 sec. (0.00 ticks)

Sync time (average) = 0.00 sec.

Wait time (average) = 0.00 sec.

Total (root+branch&cut) = 43.38 sec. (104814.92 ticks)

**** KOMUT DOSYASI GÜNLÜĞÜ****

// solution (optimal) with objective 37056

Solving time ≈ 797.23

i. Gün m Vardiyasında Çalışanlar (Satırlar GÜN, Sütunlar VARDİYA)

```
[[{8 10 19 20 28 36 53 57 71} {3 6 15 29 30 39 41 44 45 49 50 55 56 59 60 61
  64 67 74} {} {} {} {1 5 12 14 18 26 27 32 33 34 35 37 40 46 47
  48 52 65} {} {} {} {2 4 9 11 13 16 21 31 42 43 51 54 58 62 68
  69 70 72 76}]
[{15 21 30 45 49 59 75 76} {1 4 5 9 16 18 26 28 31 37 39 41 42 47
  54 60 62 63 67 68 70} {} {} {} {6 7 10 11 12 17 20 24 33 35 38
  44 51 52} {} {} {} {2 8 13 14 19 23 25 34 46 50 53 55 58 65 69
  71 72 73}]
[{11 19 23 43 46 57 63 67 68} {1 6 10 13 14 18 21 28 31 32 44 49 51
  55 71 74} {} {} {9 15 25 29 30 33 40 41 48 54 59 62 64 75} {}
  {} {} {2 7 8 12 17 20 22 26 27 34 36 37 39 42 45 50 53 69 73}]
[{1 7 23 35 47 56 58 59 62 74} {2 5 10 16 17 20 21 28 30 32 37 48
  50 57 60 65} {} {} {} {3 12 18 19 27 31 33 34 38 41 54 63 68}
  {} {} {} {13 14 15 29 40 42 49 51 52 66 67 69 72 73 75 76}]
[{14 23 25 27 36 37 44 64 70 74} {2 3 5 11 15 17 19 26 31 32 33 41
  58 60 63 69 73 75} {} {} {} {4 7 9 18 20 22 35 45 49 50 53 54
  55 57 61} {} {} {} {1 10 13 21 24 28 29 34 38 39 42 43 46 47 51
  52 62 65 66 76}]
[{4 8 24 29 30 62 65 71 73} {6 16 22 23 27 31 40 43 46 48 59 68 74}
  {} {} {} {17 28 32 34 35 57 63 75} {} {} {} {9 14 25 26 36 37
  44 51 53 55 58 60 70 76}]
[{3 9 10 22 27 44 57 68 72} {2 8 12 17 20 30 35 42 45 47 50 53 54
  56 58 66 75 76} {} {} {} {} {} {} {} {} {16 33 40 43 46 48 59 60
  61 63 65 71 74}]]
```

1. Gün 00-08 Vardiyasında Çalışanlar= {8 10 19 20 28 36 53 57 71}

1. Gün 08-16 Vardiyasında Çalışanlar= {3 6 15 29 30 39 41 44 45 49 50 55 56 59 60 61 64 67 74}

1. Gün 09-17 Vardiyasında Çalışanlar= {}

*

*

*

7. Gün 14-22 Vardiyasında Çalışanlar= {}
 7. Gün 15-23 Vardiyasında Çalışanlar= {}
 7. Gün 16-00 Vardiyasında Çalışanlar= {16 33 40 43 46 48 59 60 61 63 65 71 74}

i. Gün m Vardiyasında Toplam Çalışan Sayısı (Satırlar GÜN, Sütunlar VARDİYA)

[[9 19 0 0 0 18 0 0 0 19]
 [8 21 0 0 0 14 0 0 0 18]
 [9 16 0 0 14 0 0 0 0 19]
 [10 16 0 0 0 13 0 0 0 16]
 [10 18 0 0 0 15 0 0 0 20]
 [9 13 0 0 0 8 0 0 0 14]
 [9 18 0 0 0 0 0 0 0 13]]

1. Gün 00-08 Vardiyasında Toplam 9 işçi çalışmaktadır.
 1. Gün 08-16 Vardiyasında Toplam 19 işçi çalışmaktadır.
 1. Gün 09-17 Vardiyasında Toplam 0 işçi çalışmaktadır.

*
 *
 *

7. Gün 14-22 Vardiyasında Toplam 0 işçi çalışmaktadır.
 7. Gün 15-23 Vardiyasında Toplam 0 işçi çalışmaktadır.
 7. Gün 16-00 Vardiyasında Toplam 13 işçi çalışmaktadır.

1. Gün Çalışanlar= {1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 18 19 20 21 26 27 28 29
 30 31 32 33 34
 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
 60 61 62 64 65 67 68 69 70 71 72 74 76}

*
 *
 *

7. Gün Çalışanlar= {2 3 8 9 10 12 16 17 20 22 27 30 33 35 40 42 43 44 45 46 47 48
 50 53 54 56
 57 58 59 60 61 63 65 66 68 71 72 74 75 76}

1. Gün Çalışmayanlar= {7 17 22 23 24 25 38 63 66 73 75}

*
 *
 *

7. Gün Çalışmayanlar= {1 4 5 6 7 11 13 14 15 18 19 21 23 24 25 26 28 29 31 32 34
 36 37 38 39 41 49 51 52 55 62 64 67 69 70 73}

j. İşçinin Toplam Çalıştığı Süre (Sütunlar İŞÇİ)

[40 48 32 32 32 32 32 40 48 48 32 40 40 48 40 40 48 40 32 40 24 32
 40 48 48 40 48 48 40 48 48 48 32 48 24 32 40 40 48 40 48 40 40 40
 48 48 32 48 48 40 24 48 48 48 48 24 48 48 24 32 48 40 32 40 32 40
 48 48 48]

j. İşçinin i. Gün Çalışıp Çalışmadığı (Satırlar İŞÇİ, Sütunlar GÜN)

```

[[1 1 1 1 1 0 0]
 [1 1 1 1 1 0 1]
 [1 0 0 1 1 0 1]
 *
 *
 [1 0 1 1 1 1 1]
 [0 1 1 1 1 1 1]
 [1 1 0 1 1 1 1]]

```

1. İşçi {1 2 3 4 5}. günlerinde çalışmakta {6 7}. günde/günlerinde çalışmamaktadır.
2. İşçi {1 2 3 4 5 7}. günlerinde çalışmakta {6}. günde/günlerinde çalışmamaktadır.
3. İşçi {1 4 5 7}. günlerinde çalışmakta {2 3 6}. günde/günlerinde çalışmamaktadır.

*

*

*

74. İşçi {1 3 4 5 6 7}. günlerinde çalışmakta {2}. günde/günlerinde çalışmamaktadır.
75. İşçi {2 3 4 5 6 7}. günlerinde çalışmakta {1}. günde/günlerinde çalışmamaktadır.
76. İşçi {1 2 4 5 6 7}. günlerinde çalışmakta {3}. günde/günlerinde çalışmamaktadır.

j. İşçinin i. Gün Çalıştığı Vardiya (Satırlar İŞÇİ, Sütunlar GÜN)

```

[{"12-20"} {"08-16"} {"08-16"} {"00-08"} {"16-00"} {} {}]
[{"16-00"} {"16-00"} {"16-00"} {"08-16"} {"08-16"} {} {"08-16"}]
[{"08-16"} {} {} {"12-20"} {"08-16"} {} {"00-08"}]
[{"16-00"} {"08-16"} {} {} {"12-20"} {"00-08"} {}]

```

*

*

*

```

[{"08-16"} {} {"08-16"} {"00-08"} {"00-08"} {"08-16"} {"16-00"}]
[{} {"00-08"} {"11-19"} {"16-00"} {"08-16"} {"12-20"} {"08-16"}]
[{"16-00"} {"00-08"} {} {"16-00"} {"16-00"} {"16-00"} {"08-16"}]]

```

1. İşçi 1. Gün 12-20 Vardiyasında çalışmakta ve ilk molasını {54}. periyotta, yemek molasını {65}. periyotta, ikinci molasını {73}. periyotta ve üçüncü molasını {75}. periyotta vermektedir.

1. İşçi 2. Gün 08-16 Vardiyasında çalışmakta ve ilk molasını {40}. periyotta, yemek molasını {45}. periyotta, ikinci molasını {58}. periyotta ve üçüncü molasını {59}. periyotta vermektedir.

*

*

*

76. İşçi 6. Gün 16-00 Vardiyasında çalışmakta ve ilk molasını {69}. periyotta, yemek molasını {80}. periyotta, ikinci molasını {90}. periyotta ve üçüncü molasını {95}. periyotta vermektedir.

{ }
 {46} {16 59} {40 48 65} {43 71} {33 60 63} {61 74} { } { } { } { }
 { } { }]]

1. Gün 21. Periyotta {8 19}. işçi/işçiler ikinci molaya çıkmaktadır.

1. Gün 22. Periyotta {20}. işçi/işçiler ikinci molaya çıkmaktadır.

*

*

*

7. Gün 89. Periyotta {33 60 63}. işçi/işçiler ikinci molaya çıkmaktadır.

7. Gün 90. Periyotta {61 74}. işçi/işçiler ikinci molaya çıkmaktadır.

i. Gün p. Periyot Üçüncü Molasına Çıkanlar (Satırlar GÜN, Sütunlar PERİYOT)

[{ }
 { } {10} {8 20 28} {36 57} {19} { } {53 71} { } { } { } { } { } { }
 { }
 44 49} {41 50 59 60 67} {15} {29 45} {39 61} {6 55 56 64 74} { }
 { }
 65} {18 26 33} {32 34 35 48} { }
 13 31 51} {2 69 72} {16 21 42 43 62 70} {9 68} {11 76} {54 58}]

*

*

*

[{ }
 { } { } { } { } {22 68} {9 57} { } { } {3 10} {27 44 72} { } { } { } { }
 { }
 { } {35 42 53} {56 58} {17 66} {8 54 76} {2 12 45 50} {20 30 47
 75} { }
 { } { } { } { } { } { } {48 60} {46 59} {16 61 63} {33 43} {40 65} {71
 74}]

1. Gün 27. Periyotta {10}. işçi/işçiler üçüncü molaya çıkmaktadır.

1. Gün 28. Periyotta {8 20 28}. işçi/işçiler üçüncü molaya çıkmaktadır.

*

*

*

7. Gün 95. Periyotta {40 65}. işçi/işçiler üçüncü molaya çıkmaktadır.

7. Gün 96. Periyotta {71 74}. işçi/işçiler üçüncü molaya çıkmaktadır.

j. İŞÇİNİN i. GÜN Birinci Molaya Çıktığı Periyot (Satırlar İŞÇİ, Sütunlar GÜN)

[{54} {40} {40} {6} {71} { } { }]
 [{70} {67} {69} {40} {35} { } {40}]
 [{35} { } { } {51} {35} { } {3}]
 [{72} {40} { } { } {55} {7} { }]

*

*

*

[{ } {4} {52} {68} {37} {51} {35}]

[{70} {6} {} {67} {71} {69} {38}]]

j.İŞÇİNİN i. GÜN Yemek Molasına Çıktığı Periyot (Satırlar İŞÇİ, Sütunlar GÜN)

[[{65} {45} {48} {13} {76} {} {}]

[{76} {81} {76} {45} {47} {} {43}]

[{45} {} {} {64} {49} {} {17}]

*

*

*

[{} {16} {60} {79} {45} {59} {48}]

[{77} {16} {} {76} {79} {80} {46}]]

j.İŞÇİNİN i. GÜN İkinci Molaya Çıktığı Periyot (Satırlar İŞÇİ, Sütunlar GÜN)

[[{73} {58} {58} {25} {88} {} {}]

[{88} {90} {88} {54} {54} {} {53}]

[{58} {} {} {72} {53} {} {21}]

[{90} {57} {} {} {74} {23} {}]

*

*

*

[{} {24} {66} {90} {55} {73} {54}]

[{86} {21} {} {90} {90} {90} {55}]]

j.İŞÇİNİN i. GÜN Üçüncü Molaya Çıktığı Periyot (Satırlar İŞÇİ, Sütunlar GÜN)

[[{75} {59} {64} {30} {94} {} {}]

[{92} {91} {96} {59} {60} {} {63}]

[{59} {} {} {80} {60} {} {31}]

*

*

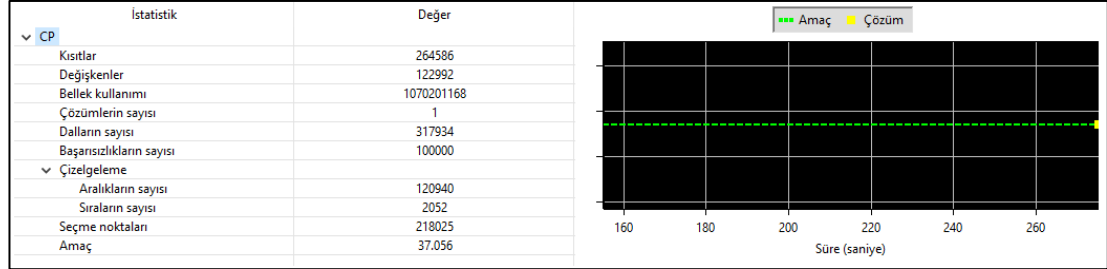
*

[{} {31} {76} {93} {60} {79} {64}]

[{95} {32} {} {93} {91} {95} {62}]]

EK H: Personel çizelgeleme probleminin %22'lik artış olan durumunun Model 2b ile çözülmesi sonucu ortaya çıkan IBM ILOG CPLEX sonuçları

////** İSTATİSTİKLER**



////** İŞLEMÇİ GÜNLÜĞÜ**

```
! -----
! Minimization problem- 122.992 variables, 264.586 constraints, 1 phase
! FailLimit      = 100.000
! SearchType     = Restart
! Workers        = 1
! Initial process time: 0,59s (0,00s extraction + 0,58s propagation)
! . Log search space : 685.261,6 (before), 685.261,6 (after)
! . Memory usage    : 610,9 MB (before), 752,3 MB (after)
! Using sequential search.
! -----
!
!      Best Branches Non-fixed   Branch decision
!          1.000   29.300      on actworkers#19#56
!          2.000   18.492      on tiallocs#34654
!          3.000   18.998      on tiallocs#88031
!          4.000   18.672      on tiallocs#45191
!          5.000    9.889      on tiallocs#16524
!          6.000    4.193      on tiallocs#110560
!          7.000    4.186      on tiallocs#2842
!          8.000    7.787      on tiallocs#108092
!          9.000    3.404      on tiallocs#18393
!         10.000    1.124      on tiallocs#83847
!
! * 37.056 10.491 7,07s -
!    37.056 11.000 27.332      on actworkers#5#9
!    37.056 12.000 25.850      on actworkers#7#53
!    37.056 13.000 25.850      on actworkers#3#72
!    37.056 14.000 25.850      on actworkers#3#72
!    37.056 15.000 25.850      on actworkers#22#43
!    37.056 16.000 25.846      on actworkers#19#33
!    37.056 17.000 25.854      on actworkers#10#39
!    37.056 18.000  1.776      on tiallocs#74073
!    37.056 19.000    2      F on tiallocs#97265
! Time = 15,83s, Average fail depth = 477, Memory usage = 1,1 GB
```

! Best Branches Non-fixed Branch decision

37.056	20.000	2.232	F	-
37.056	21.000	1.794		on tiallocs#112254
37.056	22.000	2	F	on tiallocs#116670
37.056	23.000	3.088		on tiallocs#1794
37.056	24.000	6.798	F	on tiallocs#11829
37.056	25.000	6.798		on tiallocs#12046
37.056	26.000	6.798	F	on tiallocs#11998
37.056	27.000	1.358		on tiallocs#93634
37.056	28.000	2	F	on tiallocs#117063
37.056	29.000	1.777		on tiallocs#108029
37.056	30.000	2	F	on tiallocs#117063
37.056	31.000	2.178		on tiallocs#44558
37.056	32.000	170		on tiallocs#105630
37.056	33.000	2	F	on tiallocs#30444
37.056	34.000	2.928		on tiallocs#29700
37.056	35.000	894		on tiallocs#61562
37.056	36.000	2	F	on tiallocs#110832
37.056	37.000	1.302		on tiallocs#54292
37.056	38.000	2	F	on tiallocs#96542
37.056	39.000	720		on tiallocs#90339

! Time = 35,60s, Average fail depth = 763, Memory usage = 1,1 GB

! Best Branches Non-fixed Branch decision

37.056	40.000	2	F	on tiallocs#116796
37.056	41.000	3.088	F	on actworkers#23#69
37.056	42.000	5.971	F	on actworkers#23#56
37.056	43.000	5.958	F	on actworkers#15#12
37.056	44.000	2.090		on tiallocs#12293
37.056	45.000	90		on tiallocs#100917
37.056	46.000	2	F	on tiallocs#109679
37.056	47.000	3.088	F	on actworkers#3#52
37.056	48.000	1.360		on tiallocs#32606
37.056	49.000	2		on tiallocs#20226
37.056	50.000	3.430		on tiallocs#48963
37.056	51.000	3.430		on tiallocs#11746
37.056	52.000	3.430	F	on tiallocs#12456
37.056	53.000	3.430		on tiallocs#11998
37.056	54.000	3.430	F	on tiallocs#11998
37.056	55.000	3.430	F	on tiallocs#12456
37.056	56.000	3.430		on tiallocs#53810
37.056	57.000	3.335		on tiallocs#116087
37.056	58.000	2	F	on tiallocs#110186
37.056	59.000	4.796	F	-

! Time = 44,54s, Average fail depth = 648, Memory usage = 1,1 GB

! Best Branches Non-fixed Branch decision

37.056	60.000	5.954	F	on actworkers#3#25
37.056	61.000	5.943		on actworkers#26#48
37.056	62.000	5.942	F	on actworkers#26#34

37.056	63.000	2.742		on tiallocs#35034
37.056	64.000	254		on tiallocs#81089
37.056	65.000	2	F	on tiallocs#44862
37.056	66.000	4.113		on tiallocs#49110
37.056	67.000	4.113		on tiallocs#49198
37.056	68.000	4.113		on tiallocs#44862
37.056	69.000	4.113		on tiallocs#49110
37.056	70.000	4.113	F	on tiallocs#48369
37.056	71.000	2.690		on tiallocs#58923
37.056	72.000	634		on tiallocs#33865
37.056	73.000	2	F	on tiallocs#101136
37.056	74.000	3.095		on actworkers#5#59
37.056	75.000	18.402		on tiallocs#11883
37.056	76.000	18.402	F	on tiallocs#11959
37.056	77.000	18.402		on tiallocs#11959
37.056	78.000	18.402	F	on tiallocs#11883
37.056	79.000	18.402	F	on tiallocs#12042
! Time = 61,52s, Average fail depth = 626, Memory usage = 1,1 GB				
!	Best Branches	Non-fixed	Branch decision	
37.056	80.000	18.402		on tiallocs#12388
37.056	81.000	2.262		on tiallocs#6277
37.056	82.000	262		on tiallocs#63392
37.056	83.000	2		on actworkers#19#74
37.056	84.000	3.958	F	on tiallocs#40742
37.056	85.000	3.958	F	on tiallocs#40742
37.056	86.000	1.451		on tiallocs#37214
37.056	87.000	2	F	on tiallocs#110264
37.056	88.000	2.996	F	-
37.056	89.000	1.437		on tiallocs#50251
37.056	90.000	2		on tiallocs#93436
37.056	91.000	570		on tiallocs#37600
37.056	92.000	2	F	on tiallocs#101437
37.056	93.000	2.448		on tiallocs#114584
37.056	94.000	446		on tiallocs#76369
37.056	95.000	2		on tiallocs#82538
37.056	96.000	2		on tiallocs#26728
37.056	97.000	704		on tiallocs#95440
37.056	98.000	2	F	on tiallocs#101263
37.056	99.000	1.200		on tiallocs#28847
! Time = 77,41s, Average fail depth = 664, Memory usage = 1,1 GB				
!	Best Branches	Non-fixed	Branch decision	
37.056	100k	2	F	on tiallocs#106116
37.056	101k	3.050	F	-
37.056	102k	1.086		on tiallocs#3546
37.056	103k	2	F	on tiallocs#110384
37.056	104k	2.896		on actworkers#3#69
37.056	105k	25.850		on actworkers#19#42
37.056	106k	2.086		on tiallocs#57197

37.056	107k	84		on tiallocs#117223
37.056	108k	2		on tiallocs#94092
37.056	109k	2	F	on tiallocs#117062
37.056	110k	2.028		on tiallocs#3205
37.056	111k	24		on tiallocs#109859
37.056	112k	2		on tiallocs#57244
37.056	113k	2.506		on tiallocs#88454
37.056	114k	504		on tiallocs#46833
37.056	115k	2	F	on tiallocs#91512
37.056	116k	2.368		on tiallocs#1415
37.056	117k	362		on tiallocs#71949
37.056	118k	2		on tiallocs#24706
37.056	119k	20		on tiallocs#109588

! Time = 96,72s, Average fail depth = 689, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	120k	2	F	on tiallocs#110907
37.056	121k	2.986		on tiallocs#73195
37.056	122k	18.402	F	on tiallocs#45135
37.056	123k	18.402		on tiallocs#12421
37.056	124k	18.402	F	on tiallocs#12403
37.056	125k	18.402	F	on tiallocs#44375
37.056	126k	18.402	F	on tiallocs#44375
37.056	127k	18.402	F	on tiallocs#12179
37.056	128k	1.566		on tiallocs#98881
37.056	129k	2	F	on tiallocs#92051
37.056	130k	2.434	F	-
37.056	131k	1.478		on tiallocs#107398
37.056	132k	2	F	on tiallocs#109859
37.056	133k	2.940	F	on actworkers#10#73
37.056	134k	8.117		on tiallocs#6277
37.056	135k	6.001	F	on tiallocs#44082
37.056	136k	6.001	F	on tiallocs#48494
37.056	137k	6.001	F	on tiallocs#44712
37.056	138k	1.274		on tiallocs#116267
37.056	139k	2	F	on tiallocs#114109

! Time = 112,12s, Average fail depth = 708, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	140k	2.728	F	-
37.056	141k	2.857		on tiallocs#54118
37.056	142k	2		on tiallocs#93363
37.056	143k	741		on tiallocs#38144
37.056	144k	2	F	on tiallocs#11739
37.056	145k	2.640		on tiallocs#11609
37.056	146k	1.826		on tiallocs#60344
37.056	147k	2	F	on tiallocs#100950
37.056	148k	1.083		on tiallocs#25414
37.056	149k	2		on tiallocs#1718
37.056	150k	689		on tiallocs#67164

37.056	151k	2		on tiallocs#9438
37.056	152k	2	F	on tiallocs#96875
37.056	153k	2.260		on tiallocs#25196
37.056	154k	2	F	on tiallocs#117321
37.056	155k	2.368	F	-
37.056	156k	1.458		on tiallocs#106962
37.056	157k	2	F	on tiallocs#116994
37.056	158k	2.630		on tiallocs#85375
37.056	159k	628		on tiallocs#51248

! Time = 122,64s, Average fail depth = 706, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	160k	2		on actworkers#7#22
37.056	161k	6.897	F	on tiallocs#30697
37.056	162k	6.897	F	on tiallocs#49430
37.056	163k	6.897		on tiallocs#12466
37.056	164k	6.897		on actworkers#19#36
37.056	165k	18.402	F	on tiallocs#12387
37.056	166k	1.248		on tiallocs#41189
37.056	167k	2		on tiallocs#20438
37.056	168k	2	F	on tiallocs#114013
37.056	169k	2.144	F	-
37.056	170k	1.548		on tiallocs#116201
37.056	171k	2		on tiallocs#63881
37.056	172k	488		on tiallocs#42000
37.056	173k	2		on tiallocs#34659
37.056	174k	302		on tiallocs#75709
37.056	175k	2		on tiallocs#108158
37.056	176k	2		on tiallocs#23237
37.056	177k	2.752		on tiallocs#52567
37.056	178k	750		on tiallocs#108674
37.056	179k	2	F	on tiallocs#109588

! Time = 139,83s, Average fail depth = 713, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	180k	2.794		on tiallocs#85343
37.056	181k	762		on tiallocs#109060
37.056	182k	2		on tiallocs#98201
37.056	183k	540		on tiallocs#51767
37.056	184k	2		on tiallocs#94389
37.056	185k	2	F	on tiallocs#113326
37.056	186k	2.550	F	-
37.056	187k	1.490		on tiallocs#94493
37.056	188k	2		on tiallocs#82129
37.056	189k	776		on tiallocs#42906
37.056	190k	2	F	on tiallocs#91765
37.056	191k	1.238		on tiallocs#22727
37.056	192k	2		on actworkers#0#15
37.056	193k	17.512		on tiallocs#39867
37.056	194k	1.606		on tiallocs#89194

37.056	195k	2		on tiallocs#26810
37.056	196k	904		on tiallocs#61967
37.056	197k	2	F	on tiallocs#92562
37.056	198k	2.930		on tiallocs#47857
37.056	199k	889		on tiallocs#13167

! Time = 153,49s, Average fail depth = 726, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	200k	2		on tiallocs#31463
37.056	201k	122		on tiallocs#109959
37.056	202k	2		on tiallocs#106748
37.056	203k	152		on tiallocs#116627
37.056	204k	2		on actworkers#25#66
37.056	205k	25.846	F	on actworkers#7#47
37.056	206k	25.846	F	on actworkers#7#43
37.056	207k	25.846		on actworkers#7#47
37.056	208k	25.850	F	on actworkers#7#12
37.056	209k	25.850		on actworkers#7#44
37.056	210k	25.850	F	on actworkers#19#33
37.056	211k	25.850	F	on actworkers#10#20
37.056	212k	43.844	F	on actworkers#10#32
37.056	213k	43.844		on actworkers#10#26
37.056	214k	25.854	F	on actworkers#7#38
37.056	215k	25.854	F	on actworkers#19#68
37.056	216k	25.854		on actworkers#19#15
37.056	217k	25.854		on actworkers#7#7
37.056	218k	25.854	F	on actworkers#19#75
37.056	219k	25.854	F	on actworkers#19#27

! Time = 169,53s, Average fail depth = 701, Memory usage = 1,1 GB

!	Best Branches	Non-fixed		Branch decision
37.056	220k	2.821		on tiallocs#40303
37.056	221k	2.821		on tiallocs#30125
37.056	222k	2.821	F	on tiallocs#48591
37.056	223k	2.276		on tiallocs#20118
37.056	224k	268		on tiallocs#63436
37.056	225k	2	F	on tiallocs#110907
37.056	226k	2.160		on tiallocs#57312
37.056	227k	136		on tiallocs#109730
37.056	228k	2		on tiallocs#31568
37.056	229k	2	F	on tiallocs#116421
37.056	230k	2.664	F	-
37.056	231k	3.079		on tiallocs#102843
37.056	232k	2	F	on tiallocs#113895
37.056	233k	5.052	F	-
37.056	234k	1.394		on tiallocs#36563
37.056	235k	2	F	on tiallocs#117110
37.056	236k	920		on tiallocs#12738
37.056	237k	2	F	on actworkers#19#54
37.056	238k	25.850	F	on actworkers#11#24

```

37.056 239k 25.850 on actworkers#19#9
! Time = 180,87s, Average fail depth = 701, Memory usage = 1,1 GB
! Best Branches Non-fixed Branch decision
37.056 240k 25.850 on actworkers#11#60
37.056 241k 25.850 on actworkers#11#63
37.056 242k 25.850 on actworkers#11#74
37.056 243k 2.388 on tiallocs#15452
37.056 244k 40 on tiallocs#96584
37.056 245k 2 F on tiallocs#116667
37.056 246k 1.374 on tiallocs#89455
37.056 247k 2 on tiallocs#27274
37.056 248k 212 on tiallocs#109706
37.056 249k 2 on tiallocs#79642
37.056 250k 2 on actworkers#15#29
37.056 251k 7.858 on tiallocs#102241
37.056 252k 5.227 on tiallocs#58785
37.056 253k 2.508 on tiallocs#6907
37.056 254k 470 on tiallocs#71288
37.056 255k 2 F on tiallocs#114013
37.056 256k 3.088 on tiallocs#97739
37.056 257k 5.740 on tiallocs#11483
37.056 258k 5.740 on tiallocs#12427
37.056 259k 5.740 F on tiallocs#49582
! Time = 207,19s, Average fail depth = 691, Memory usage = 1,1 GB
! Best Branches Non-fixed Branch decision
37.056 260k 5.269 F on tiallocs#44546
37.056 261k 1.044 on tiallocs#22747
37.056 262k 2 on tiallocs#53589
37.056 263k 264 on tiallocs#72096
37.056 264k 2 on tiallocs#111799
37.056 265k 2 F on tiallocs#105169
37.056 266k 1.549 on tiallocs#45570
37.056 267k 2 on tiallocs#59347
37.056 268k 680 on tiallocs#95439
37.056 269k 2 on actworkers#19#9
37.056 270k 2.756 on tiallocs#76871
37.056 271k 722 on tiallocs#42819
37.056 272k 2 on tiallocs#52984
37.056 273k 150 on tiallocs#116573
37.056 274k 2 on tiallocs#116087
37.056 275k 2 F on tiallocs#92620
37.056 276k 2.570 on tiallocs#11155
37.056 277k 400 on tiallocs#67476
37.056 278k 2 F on tiallocs#116730
37.056 279k 1.998 on tiallocs#20480
! Time = 227,71s, Average fail depth = 697, Memory usage = 1,1 GB
! Best Branches Non-fixed Branch decision
37.056 280k 2 F on tiallocs#105060

```

37.056	281k	1.436		on tiallocs#103458
37.056	282k	2		on tiallocs#14386
37.056	283k	18.402		on actworkers#21#46
37.056	284k	18.402		on actworkers#8#66
37.056	285k	17.512	F	on tiallocs#40672
37.056	286k	17.512	F	on tiallocs#40672
37.056	287k	17.512		on tiallocs#30725
37.056	288k	2.829		on tiallocs#73251
37.056	289k	793		on tiallocs#109417
37.056	290k	2		on tiallocs#36218
37.056	291k	2	F	on tiallocs#117212
37.056	292k	2.870		on tiallocs#47765
37.056	293k	836		on tiallocs#90149
37.056	294k	2		on tiallocs#24012
37.056	295k	2.064		on tiallocs#53728
37.056	296k	58		on tiallocs#92765
37.056	297k	2		on tiallocs#62077
37.056	298k	2		on tiallocs#28307
37.056	299k	437		on tiallocs#72175

! Time = 249,80s, Average fail depth = 708, Memory usage = 1,1 GB

! Best Branches Non-fixed Branch decision

37.056	300k	2	F	on tiallocs#105169
37.056	301k	2.554		on tiallocs#1883
37.056	302k	520		on tiallocs#43145
37.056	303k	2		on tiallocs#19733
37.056	304k	312		on tiallocs#67636
37.056	305k	2	F	on tiallocs#92411
37.056	306k	2.336		on tiallocs#27396
37.056	307k	278		on tiallocs#84697
37.056	308k	2		on tiallocs#11759
37.056	309k	46		on tiallocs#110681
37.056	310k	2		on tiallocs#8873
37.056	311k	2		on tiallocs#81903
37.056	312k	2.038		on tiallocs#11925
37.056	313k	38		on tiallocs#105905
37.056	314k	2	F	on tiallocs#105060

! -----

! Search terminated by limit, 1 solution found.

! Best objective : 37.056

! Number of branches : 314.934

! Number of fails : 100.000

! Total memory usage : 1,1 GB (1,1 GB CP Optimizer + 0,1 GB Concert)

! Time spent in solve : 263,06s (263,06s engine + 0,00s extraction)

! Search speed (br. / s) : 1.208,6

! -----

ÖZGEÇMİŞ

Turgay Türker, 09.09.1985’de Samsun’da doğdu. İlk, orta ve lise eğitimini Samsun’un Çarşamba ilçesinde tamamladı. 2002 yılında Çarşamba Lisesi’nden mezun oldu. 2003 yılında başladığı Kocaeli Üniversitesi Endüstri Mühendisliği Bölümü’nü 2008 yılında bölüm birincisi olarak bitirdi. 2010 yılında Karabük Üniversitesi Fen Bilimleri Enstitüsü Makine Mühendisliği Ana Bilim Dalı (ABD)’nda yüksek lisans eğitime başladı. Bu eğitimini 2012 yılında tamamladı ve akabinde aynı yıl doktora eğitimine Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Ana Bilim Dalı (ABD)’nda devam etti. Evli ve 1 çocuk sahibidir.