TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MONTAJ HATTI BESLEME İŞLEMLERİNİN ÇİZELGELENMESİ İÇİN BİR ÇÖZÜM YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Melis ÖZCAN

Endüstri Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Kadir ERTOĞRAL

AĞUSTOS 2016

Fen Bilimleri Enstitüsü Onayı

_____

Prof. Dr. Osman EROĞUL

Müdür

Bu tezin Yüksek Lisans derecesinin tüm gereksinimlerini sağladığını onaylarım.

_____

Prof. Dr. Tahir HANALİOĞLU

Anabilim Dalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 121311116 numaralı Yüksek Lisans Öğrencisi **Melis ÖZCAN** 'ın ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"MONTAJ HATTI BESLEME İŞLEMLERİNİN ÇİZELGELENMESİ İÇİN BİR ÇÖZÜM YAKLAŞIMI"** başlıklı tezi **15.08.2016** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı: Doç. Dr. Kadir ERTOĞRAL _____

TOBB Ekonomive Teknoloji Üniversitesi

Jüri Üyeleri: **Doç. Dr. Hakan GÜLTEKİN (Başkan)** _____

TOBB Ekonomive Teknoloji Üniversitesi

Doç. Dr. Metin DAĞDEVİREN _____

Gazi Üniversitesi

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Melis ÖZCAN

ÖZET

Yüksek Lisans Tezi

MONTAJ HATTI BESLEME İŞLEMLERİNİN ÇİZELGELENMESİ İÇİN BİR ÇÖZÜM YAKLAŞIMI

Melis ÖZCAN

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Kadir ERTOĞRAL

Ağustos 2016

Bu çalışmada ele alınan problem, bir bulaşık makinesi üretim fabrikasındaki montaj hatları malzeme beslemeyle ilgili gerçek hayat probleminden yola çıkılarak oluşturulmuştur. Problem montaj hatlarına malzeme besleme operasyonunun çizelgelenmesidir. Sistemde paralel birden fazla montaj hattı bulunmaktadır ve montaj hatlarının her iki yanında iş istasyonları bulunabilmektedir. Montaj hatlarında bir vardiya boyunca farklı modeller üretilir. Montaj istasyonlarına malzemeler ve parçalar merkezi bir depodan taşıyıcı personel tarafından transpallet denilen taşıma ekipmanı kullanılarak iki farklı ebattaki kutular içerisinde taşınır. Her bir taşıyıcı sorumlu olduğu istasyonların malzeme ve parça ihtiyacını karşılar. Montaj besleme çizelgesinin oluşturulması iki ana problem şeklinde ele alınmıştır. İlk problem üretim planına bağlı olarak her iş istasyonu için vardiya boyunca gerekecek malzeme ve parça ihtiyacı taşınma işinin taşıyıcılara dengeli bir şekilde dağıtılmasıdır. İkinci ana problem ise vardiya boyunca malzeme dağıtım turlarının her taşıyıcı için ayrı ayrı planlamasıdır. İlk problem için farklı taşıyıcılara atanan, vardiya boyunca taşınacak kutu sayılarının mutlak farklarını en azlayan ve bunu yaparken taşıyıcılara atanan iş istasyonlarının belirli bir komşulukta olması kısıtını göz önüne alan matematiksel bir model geliştirilmiştir. Bu problem için bir de etkin sezgisel bir yaklaşım geliştirilerek, optimal çözümlerle karşılaştırılmıştır. İkinci problemde ise iki amaç söz konusudur. Bunlar; 1-Taşıyıcının vardiya boyunca yaptığı malzeme besleme tur sayısının azaltılması. 2-Montaj istasyonlarındaki stok yığılmalarının en azlanmasıdır.

Malzeme besleme tur sayısının minimizasyonu malzeme besleme eforunu, harcanan enerjiyi, fabrika içinde oluşan trafik yoğunluğunu en azlama anlamında önemli bir amaçtır. Vardiya boyunca istasyonlarda biriken stokların azaltılması da istasyondaki işlemlerin verimli bir şekilde devamı, malzeme akış trafiğinin rahatlaması ve istasyonlardaki stok alanının limitli olması açılarından önemli bir amaçtır. Bu iki tezat amaçtan ilkini amaç ve ikincisini kısıt olarak göz önüne alan, vardiyanın eşit zaman dilimlerine bölünmesine ve bu zaman dilimlerinde yapılacak malzeme besleme turlarının planlanmasına dayalı bir matematiksel model geliştirilmiştir. Bu ikinci problemin NP-zor sınıfına ait olduğu gösterilmiştir. Bu problemin gerçek hayatta karşılaşılan büyüklüklerinin kısa sürede çözümü amacıyla sezgisel bir yaklaşım da tasarlanmış ve bulaşık makinesi fabrikasından alınan gerçek problem datasına dayalı örnek bir set problem üzerinde sezgiselin etkinliği gösterilmiştir.

Anahtar Kelimeler: Montaj hattı, Malzeme besleme çizelgeleme

ABSTRACT

Master Thesis

A SOLUTION APPROACH FOR SCHEDULING ASSEMBLY LINE FEEDING OPERATIONS

Melis ÖZCAN

TOBB EconomicsandTechnologyUniversity
Institute of Natural and Applied Sciences
Industrial Engineering

Supervisor: Assoc. Prof. Kadir ERTOĞRAL

August 2016

The problem addressed in this thesis is motivated by a real problem in the assembly lines of a plant that produces washing machines. The problem is the scheduling of the operations for feeding the materials to the assembly lines. There are several assembly lines in the system with, potentially, stations in both sides of the line. Multiple models are produced in the lines in a single shift. The component and parts used in the assembly are transfered from a central depot location to the lines in two different size containers using the material handling equipment called transpallets. Each transporter carries material to the stations that are assigned to his responsibility. The scheduling of material feeding to the lines is handled as two separate problems. First problem is about balancing the workload of transporters to carry the required materials to the lines during a shift based on the production plan. The second main problem is the scheduling of the tours to carry the required materials for each transporter separately. For the first problem, we developed a mathematical model that minimizes the absolute differences of the assigned containers to different transporters in a shift. We also suggested an effective heuristic approach for the load balancing problem and compared its results with optimal solutions. The second main problem has two objectives; First objective is minimizing the number of tours that the transporter makes in a shift, while the second objective is minimizing the inventory accumulations at the assembly stations. For the second problem we introduced a mathematical model based on dividing the shift into equal time intervals

This model takes the tour minimization as the objective while assuming a limit for the inventory at the stations as a constraint. The problem is shown to be NP-hard. We also suggested a heuristic solution approach for the problem in order to solve the bigger problems of real life in short amount of time and we showed the effectiveness of the solution approach on a set of problems based on a real data set obtained from the washing machine producing plant.

Keywords: Assembly line balancing**,** Material feed scheduling

TEŞEKKÜR

Öncelikle bugüne kadar üstümden emeğini, sevgisini ve desteğini eksik etmeyen en başta eşime ve tüm aileme teşekkür ederim. Çalışmamda bana bilgi ve deneyimleriyle yol gösteren danışman hocam Doç. Dr. Kadir ERTOĞRAL'a değerli katkılarından dolayı teşekkürü bir borç bilirim. Kıymetli zamanlarını ayırıp tezimi okuyan ve tavsiyelerde bulunan tez jürimin saygıdeğer üyelerine, yüksek lisans eğitimimde bana sağladıkları katkı ve emeklerinden dolayı TOBB Ekonomi ve Teknoloji Üniversitesi Endüstri Mühendisliği Bölümü öğretim üyelerine de teşekkür ederim. Ayrıca, burs sağladığı için TOBB Ekonomi ve Teknoloji Üniversitesi'ne teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

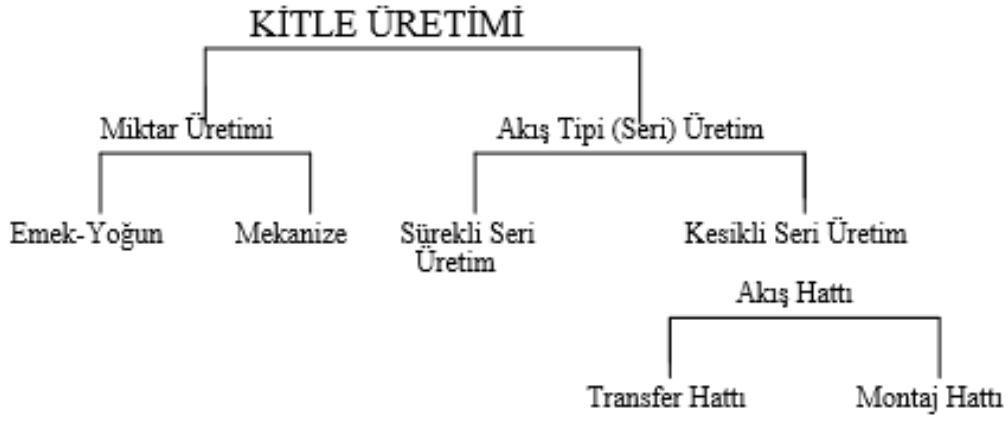
ÇİZELGE LİSTESİ

Sayfa

1. GİRİŞ

Günümüzde insan ihtiyaçlarının doğa tarafından tam olarak karşılanamamasından dolayı üretim kavramı ortaya çıkmıştır. Bazı ekonomistler üretimi fayda yaratmak olarak tanımlarken mühendislere göre fiziksel varlık üzerinde, o varlığın değerini arttıran bir değişiklik yapmak veya hammadde/yarı mamülleri kullanılır hale getirmektir. Üretim sistemleri, iş akışına ve üretim miktarlarına göre 3'e ayrılmaktadır. Bunlar; siparişe göre üretim, parti üretimi ve seri üretimdir. [1]

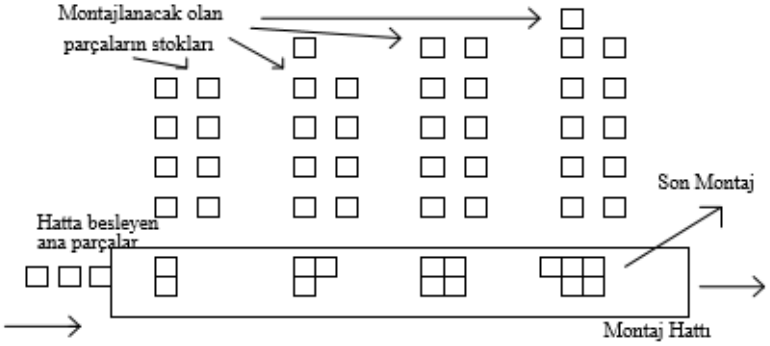
Sanayinin gösterdiği gelişmeyle birlikte üretim elemanları yaygın olarak kullanılmaya başlanmıştır ve bu da büyük miktarlarda ürün üretimi ile sonuçlanmıştır. Bu büyük miktarlı üretim nedeniyle kitle üretim sistemleri "miktar üretim" olarak da nitelendirilir. Daha sonra ise karmaşık ürün üretimine başlanılmıştır ve ürün artık akışı olan bir üretim teknolojisi ile üretilir bu nedenle "akış üretimi" olarak da adlandırılır. Kitle üretim sistemleri Şekil 1.1'de ki gibi gösterilebilir. [2]



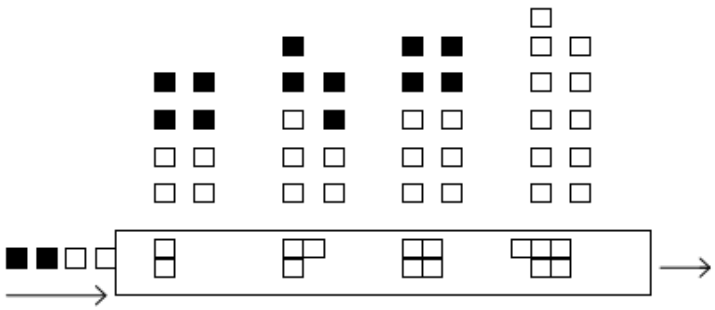
Şekil 1.1 Kitle Üretim Sistemleri

Montaj hatları ilk olarak otomotiv sanayinde Henry Ford tarafından geliştirilmiş olup bir ürünün her bir parçasının belirli istasyonlarda ürün üzerine monte edilip hattın sonunda son ürün olarak fabrikadan çıktığı üretim hatlarıdır. Montaj hatları yapılması gereken işlemlerin sıralanması ile oluşur. İş istasyonlarındaki öncelik ilişkileri göz önüne alınarak sıralama yapılır. Montaj hatları model çeşitlerine göre tek modelli, çok modelli ve karışık modelli hatlar olmak üzere 3'e ayrılır.

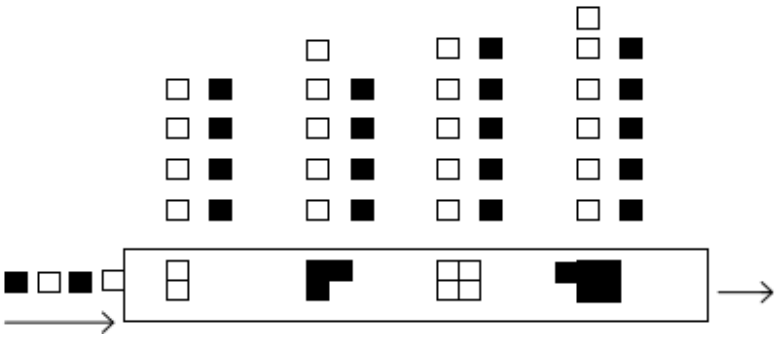
1. Tek Modelli Hatlar: Tek tip ürün üretimi, Şekil 1.2'de gösterilmiştir.
2. Çok Modelli Hatlar: Farklı ürünler, farklı modellerin üretimi, Şekil 1.3'de gösterilmiştir.
3. Karışık Modelli Hatlar: Montaj hattında aynı anda benzer tipteki modellerin üretimi için kullanılır, Şekil 1.4'de gösterilmiştir. Değişik işlem zamanına sahip ürünler olması nedeniyle iş akışının düzene sokulması gerekir, boş zamanlar artar ve yarı mamül stoğunda artış olur. [2]



Şekil 1.2.Tek Modelli Montaj Hattı [2]



Şekil 1.3.Çok Modelli Montaj Hattı [2]



Şekil 1.4.Karışık Modelli Montaj Hattı [2]

Bu tez çalışmasında beyaz eşya sektöründe önemli bir yere sahip bir firmanın bulaşık makinesi hattı ele alınmıştır. Fabrikada montaj işlemi 3 ayrı bantta yapılmaktadır. Tam boy, tam ankastre ve yarı ankastre olan ürün yelpazesinde yaklaşık 1.300 farklı model çeşidi vardır. 2005 yılında, JIPM (Japon Planlı Bakım) Enstitüsü'nden alınan TPM Mükemmellik Ödülü'yle üretimde rekabetçi yapısını tescilleyen işletme, esnek üretim yapısıyla rekabetçiliğini artırmaktadır. Üretiminin yaklaşık olarak %50'si ihraç edilmektedir. Fabrika, bu ihracatının çoğunu Avrupa Birliği ülkeleri olmak üzere 100'den fazla ülkeye yapmaktadır.

Bu çalışmanın ana amacı fabrika içi montaj bandında kullanılan malzemelerin dağıtım sisteminin iyileştirilmesi ve malzemecilere eşit yük dağılımı sağlanmasıdır. Bu kapsam altında alt amaçlar ise fabrika içerisinde montaj bantlarının yanlarında bulunan fazla stokları en aza indirmek ve malzemecilerin yaptığı tur sayısını minimize etmektir. Böylece fabrika içi trafik azalmış olur ve kaza riski de azalır. Montaj bantlarının temel işleyişi, taşınma biçimleri, montaj bantlarını besleyen malzemeciler ve bu malzemecilerin rotasyonları, istenmeyen durumların sebepleri gibi pek çok konuda yapılan gözlemler incelenmiştir. Sorunları en iyi şekilde yönetebilmek için matematiksel modeller oluşturulmuş ve optimal çözüm elde edilmiş, aynı zamanda daha kısa zamanda iyi çözümler elde edebilmek adına sezgisel algoritmalar geliştirilmiştir.

Montaj hatlarının analizi, montaj hatlarında dengeleme çalışmaları ve uygulamaları literatürde ve pratikte çokça çalışılmış olup montaj hattı besleme işlemlerinin çizelgelenmesi ile ilgili çok sayıda araştırma ve ya uygulama olmaması dikkat çekmektedir. Bu tez çalışmasında geliştirilecek yöntemler, matematiksel modeller ve sezgisel algoritmalar literatürdeki konu ile ilgili az sayıdaki kaynaklardan biri

olacaktır. Bölüm 2'de literatür incelemesi yapılmıştır. Bölüm 3'te ilk ana problemin tanımı, modelleme yaklaşımı ve matematiksel modeli verilmiştir. Matematiksel model dışında bir çözüm yaklaşımından bahsedilerek sezgisel algoritma adımları anlatılmıştır. Bölüm 3'ün devamında ikinci ana problemin tanımı, modelleme yaklaşımı ve matematiksel modeli verilmiştir. Problemin NP-zor sınıfında olması ve gerçek hayat problemlerinde optimizasyon zamanının uzunluğundan dolayı daha hızlı sonuç alınabilecek bir sezgisel algoritma geliştirilmiştir ve bu algoritmanın adımları anlatılmıştır. Bölüm 4'te aynı veri kümeleri ile çözülen matematiksel model ve sezgisel algoritma için çözüm süreleri ve sonuçlar karşılaştırılmıştır. Problemler ve sonuçlar ile ilgili genel değerlendirmeler yapılmıştır.

2. LİTERATÜR ÇALIŞMASI

Montaj hatları ilk kez 20 yy. başlarında Henry Ford tarafından Ford fabrikalarında kullandırılmaya başlandı. Üretimde kalitenin artması, verimliliğin artması ve insan gücüne dayalı sistemin azaltılmak istenmesiyle montaj hatlarına olan ilgi artmıştır ve zaman içinde iyileştirilmeye çalışılmıştır. En çok çalışılan problemler montaj hattı dengeleme problemleri olup çok geniş bir literatür aralığı vardır. [2]

Scholl ve Becker'in 2004 [3] yılında yayınladığı tarama niteliğinde olan literatür basit montaj hatlarını anlatmaktadır. 2006 yılında ise yine Scholl ve Becker [4] genel montaj hatları için sınıflandırmalar sunan bir literatür yayınlamışlardır.

Bu çalışma kapsamında ele alınan paralel montaj hatlarında malzeme besleme taşıyıcılar tarafından transpallet ile yapılmaktadır. Taşıyıcılara aynı oranda iş yükü dağılması, hat başındaki stokların düşürülmesi ve hem de taşıyıcılar tarafından toplamda yapılan tur sayısının azaltılmasını amaçlayan bir malzeme besleme çizelgelemesi çalışması yapılmıştır.

Literatürde yoğun olarak çalışılmış olan hat dengeleme probleminin aksine, malzeme besleme çizelgelenmesi probleminin bu çalışmada ele alındığı şekliyle üzerinde durulmamıştır. Bu sebeple montaj hattı besleme problemi için bir çizelgeleme yaklaşımı literatürdeki bu boşluğu doldurmaktadır. Gerçek hayatta çok çeşitli alanlarda fazlaca kullanılan montaj hatları için, her fabrikada ortaya çıkabilecek sorunlara eğilen ve uygulanabilir bir çözüm yöntemi geliştirilen bu çalışma literatürde çok az ele alınmış bir probleme ışık tutmaktadır.

Karma model montaj hatlarında olabilecek iki farklı ana amacın iş yükü ve malzeme teslimi ile ilgili olduğu 2007 yılında Boysen vd. [5] tarafından yapılan çalışmada belirtilmiştir. Karma model montaj hatları için, araştırma, sınıflandırma ve model incelemeleri ortaya koymuşlardır. Oldukça kapsamlı olan bu çalışmada istasyon ve hat karakteristikleri, olası tüm hedefler iki farklı ana amaç için tüm kombinasyonlarla incelenmiştir. Sıralama yaklaşımları 3 şekilde ayrılabilir, karma model sıralama,

araba sıralama ve sınıf sıralamadır. Her bir yaklaşım için teker teker tüm karakteristikler ve tüm amaç fonksiyonlarının analizi yapılmıştır.

Bu çalışma sonucunda çok amaçlı ve karma problemlerde sıralama ile ilgili geniş kapsamlı bir özet elde edilmiştir. Hiyerarşik bir sınıflandırma şeması çıkartılarak tüm problem tipleri belirli bir sistemle incelenmiştir.

2011 yılında Golz vd. [6] yaptığı çalışmada yüksek varyansı olan karışık model montaj hatlarında parça besleme problemi ele alınmıştır. Yüksek varyanslı karışık model montaj hatlarında en belirgin problemler, hat dengeleme, üretim çizelgeleme, üretim sırası belirleme, malzeme akışının kontrolü ve herhangi bir aksaklık durumunda tekrar çizelgelemedir. Bahsedilen sistemde fabrikada malzeme akışını kontrol ederken Kanban benzeri bir yaklaşım kullanılmaktadır. Malzeme kritik seviyeye düştüğünde sistemden bir uyarı gelmektedir ve normalde planlanmamış olan bir ara tur eklenerek eksilen parçaları tamamlamak için montaj hatlarına gidilmektedir. Bu sırada hem malzeme beklerken hat durabilmektedir yani verimlilik düşer hem de planlanmadan eklenen turlarla birlikte toplam tur sayısı yüksek sayılara ulaşmaktadır. Bunun önüne geçebilmek için 2 aşamalı bir algoritma geliştirmişlerdir. Algoritmanın ilk aşamasında tarihine ve parça kodlarına göre yapılması gereken işlemler ayrılmaktadır ve bu şekilde bir günlük veya bir vardiyalık taşıma emirleri oluşturulmaktadır. Daha sonra ise bu taşıma emirleri her bir taşıma emri mutlaka bir tura atanacak şekilde yerleştirilir. Bu sistemde her bir tur bir taşıyıcıya atanır ve bir taşıyıcı birden fazla tur yapamaz.

Bu problem optimizasyon programlarında makul bir zamanda çözülememektedir fakat bu algoritma ile yeterince iyi olmayan sonuçlar kısa sürede elde edilebilmektedir.

2009 yılında Battini vd. [7] tarafından yazılan literatürde karışık model montaj hatlarında besleme sistemi seçimi üzerine yoğunlaşılmıştır. Montaj hattı besleme sistemleri paletlerle istasyona, araba ile istasyona veya paket halinde montaj hattına şeklinde 3'e ayrılmaktadır. Paletlerle istasyona taşınan sistemde eğer malzemeye montaj hattında ihtiyaç kalmadıysa geri ana depoya taşınması gerekir ve bu nedenle bu taşıma sistemi aynı istasyonda kullanılacak malzemeler bir arada olduğunda anlamlıdır. Araba ile istasyona taşıma sisteminde tüm istasyonlarda ihtiyaç duyulacak tüm komponentler bir araya toplanır ve tek bir montaj hattına götürülür.

Bir önceki taşıma politikası ile kıyaslandığında yapılan taşıma sayısının azalmış olduğunu görürüz. Paket taşınması sisteminde ise her bir son ürün için gereken tüm malzemeler bir paket haline getirilir. Hazırlanan paket sayısı son ürün sayısına eşittir.

Tüm taşıma sistemleri için tek tek toplam kat edilen yollar ve toplam zaman hesaplayan eşitlikler verilmiştir. Bir diğer önemli kriter ise depolamanın nasıl yapılacağının seçilmesidir. Merkezi tek bir depodan malzemeler taşınabilir ya da montaj hatlarının yanında stok noktaları oluşturulabilir.

ANOVA (Analysis of Variance) kullanılarak tüm değişkenlerin tek tek etkisi incelenmiştir. Çalışma sonunda ise üretimin büyüklüğüne göre hangi sistemlerin kullanılması gerektiği detaylandırılmıştır. En küçük üretim miktarları için en iyi sistem paketlerin montaj hattına taşınması, orta büyüklükteki üretimler için arabayla istasyona taşınması ve büyük üretim miktarları için de paletlerle istasyona taşınması en uygun seçenekler olarak belirlenmiştir.

Bu tez kapsamında ele alınan problemde paralel montaj hatlarına yapılan malzeme taşıma işlemi için öncelikle tüm malzemecilerin dengeli bir iş yüküne sahip olması amaçlanmıştır. Malzemecilerin dengeli bir iş yüküne sahip olması sağlandıktan sonra iki ana amaç vardır, taşıyıcının bir vardiya boyunca yaptığı tur sayısının azaltılması ve montaj hatlarındaki stok yığılmalarının en azlanması. Literatürde taşıyıcılara dengeli yük atamak ve aynı zamanda tur sayısı ve stok yığılmalarını azaltmak amaçlarına sahip bir problem daha önce ele alınmamış olup, konu ile ilgili geliştirilen modeller ve algoritmalar literatürdeki boşluğu dolduracaktır.

Bu iki amaç birbirine tezattır. İlkini amaç ikincisini ise kısıt olarak ele aldığımız model sonucunda taşıyıcı için bir vardiya boyunca yapacağı malzeme besleme turlarının çizelgesi elde edilmektedir.

Rasgele araştırma algoritmaları ilk olarak Anderson [8] tarafından sunulmuştur ve daha sonra ise Karnopp [9] ve Rastrigin [10] konuyu ele almışlardır.

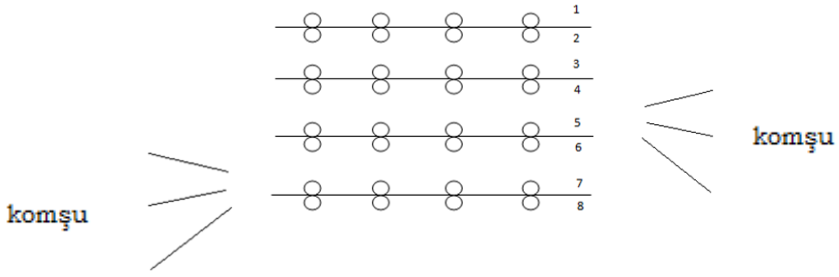
Neumann vd. [11] yaptığı çalışmada, rasgele araştırma algoritmaları ile büyük bir optimizasyon problemi sınıfı ele alınabilir. Hatta bu metotlar bazı durumlarda iyi sonuçlar verebilmektedir. Bu çalışma ile bu tür yaklaşımların çalışma mantığının nasıl olduğu, algoritma parametrelerini nasıl seçilmesi gerektiği anlatılmaktadır. Bu tez kapsamında ele alınan problemin sezgisel çözümünde de rasgele seçim algoritması kullanılmıştır.

3.PROBLEM TANIMI

Bu çalışmada çok hatlı, çok istasyonlu, birden fazla taşıyıcı bulunan bir fabrika içi montaj sisteminde, montaj hattı beslenmesi sırasında malzeme dağıtımının düzenlenmesi problemi ele alınmaktadır. Montaj hatlarında bir vardiya boyunca farklı modeller üretilir. Montaj istasyonlarına malzemeler ve parçalar merkezi bir depodan, taşıyıcı personel tarafından transpallet denilen taşıma ekipmanı kullanılarak taşınır. İncelenen her bir montaj hattı 2 sıradan oluşmaktadır. Her bir sıra üzerinde n adet istasyon bulunmaktadır. Her bir sıranın tanımlanacak bir m parametresine göre belirlenen komşu olduğu diğer sıralar vardır. Her bir taşıyıcı sadece komşu olan sıralar arasında dağıtım yapabilmektedir. Montaj besleme çizelgesinin oluşturulması iki ana problem şeklinde ele alınmıştır. İlk problem üretim planına bağlı olarak her iş istasyonu için vardiya boyunca gerekecek malzeme ve parça ihtiyacı taşınma işinin taşıyıcılara dengeli bir şekilde dağıtılmasıdır. İkinci ana problem ise vardiya boyunca malzeme dağıtım turlarının her taşıyıcı için ayrı ayrı planlamasıdır. İkinci problemde ise iki amaç söz konusudur. Bunlar; 1-Taşıyıcının vardiya boyunca yaptığı malzeme besleme tur sayısının azaltılması. 2-Montaj istasyonlarındaki stok yığılmalarının en aza indirilmesi.

Trafik yoğunluğunu azaltabilmek adına bir taşıyıcının birbirinden uzak hatlara aynı turda malzeme taşımasını engelleyici bir komşuluk kısıtı eklenmiştir. Komşuluk karar verici tarafından belirlenen bir m parametresi ile tanımlanır. Örneğin; hat sayısı 4, sıra sayısı 8 olan bir montaj hattında m=2 olarak tanımlandıysa kendisinden sonraki iki montaj hattı komşusudur. Yani, sıra 1,2,3,4,5 ve 6 birbirleriyle komşudur. Aynı zamanda sıra 3,4,5,6,7 ve 8 de birbirleriyle komşudur. Fakat sıra 1,2 ve 7,8 birbirleriyle komşu değillerdir. Bu nedenle bir taşıyıcı hem sıra 1 veya 2'deki bir istasyona hem de sıra 7 veya 8'deki bir istasyona aynı turda malzeme taşıyamaz.

Örneğimizdeki birbirine komşu olan/olmayan hatlar(istasyonlar) Şekil 3.1'de gösterilmiştir.



Şekil 3.1.Montaj Hattında Komşuluk Gösterimi

Bu problemde temel girdi olarak bir bulaşık makinesi üretim fabrikasının verileri kullanılmıştır. Bir vardiya her biri birer saatten oluşan 7 zaman dilimi içerir. Her bir zaman dilimi için, her bir istasyonun, her bir malzemeden ne kadar ihtiyacı olduğu bilinmektedir.

Montaj Hattı Girdileri

- Her bir montaj hattı için günlük üretim planları (her bir istasyonun hangi zaman diliminde hangi malzemeden ne kadar ihtiyacı olduğu)
- Her bir ürünün ürün ağaçları; söz konusu ürünü oluşturan alt ürünler ve malzemeler
- Ortalama tur süreleri; malzemecilerin tur süreleri dolayısıyla belirli bir zaman diliminde yapılacak tur sayısı

Malzemeci Girdileri

- Malzemeci sayısı
- Her bir malzemecinin taşıyabileceği maksimum ağırlık ve hacim bilgileri

Malzeme Girdileri

- Malzemenin adı
- Malzemenin kodu
- Sepetlerin hacmi
- Her bir malzemenin bir sepette kapladığı hacim

3.1. İlk Ana Problem

Üretim planına bağlı olarak her iş istasyonu için vardiya boyunca gerekecek malzeme ve parça ihtiyacı taşınma işinin taşıyıcılara dengeli bir şekilde dağıtılması problemidir. Farklı taşıyıcılara atanan, vardiya boyunca taşınacak sepet sayılarının mutlak farklarını en azlayan ve bunu yaparken taşıyıcılara atanan iş istasyonlarının belirli bir komşulukta olması kısıtını göz önüne alan matematiksel bir model geliştirilmiştir.

3.1.1.İlk ana problem matematiksel model

İlk problem matematiksel modelinde i indisi istasyonları, j indisi malzemecileri, K_i istasyonuna bir vardiyada götürülmesi gereken ürün miktarını ve z ise iki malzemecinin taşıdığı yük miktarlarının alabileceği en yüksek mutlak fark değerini gösterir.

Her bir taşıyıcı sadece komşu istasyonlara malzeme götürebilmektedir. Hangi istasyonların birbiri ile komşu olduğunu a_{ik} parametresi ile gösterilir. a_{ik}, i ve k istasyonları komşu ise 1, değilse 0 değerini atanır.

$$a_{ik}\begin{cases}1; i\ istasyonu\ ile\ k\ istasyonu\ komşu\ ise\\ 0; diğer\ durumda\end{cases}$$

Hangi istasyona hangi malzemecinin atandığı ise x_{ij} karar değişkeni ile tutulur. x_{ij}, i istasyonuna j malzemecisi atandıysa 1, atanmadıysa 0 değerini alır.

$$x_{ij}\begin{cases}1; i\ istasyonuna\ j\ malzmecisi\ atandı\ ise\\ 0; diğer\ durumda\end{cases}$$

Taşıyıcılara iş yükünün dengeli bir şekilde dağılmasını sağlayan matematiksel model aşağıdaki gibidir:

$$Min\ z \tag{3.1}$$

$$\sum_j x_{ij} = 1 \qquad \forall i \in \{1,..,I\} \tag{3.2}$$

$$x_{ij}(1 - a_{ik}) + x_{kj} \leq 1 \qquad \forall i,j,k,\ i \neq k \tag{3.3}$$

$$\sum_i k_i x_{ij} - \sum_i k_i X_{il} \leq z \qquad \forall j,l, \qquad j \neq l \tag{3.4}$$

$$-\sum_i k_i x_{ij} + \sum_i k_i x_{il} \leq z \qquad \forall j,l\ j \neq l \tag{3.5}$$

$$x_{ij} \in \{0,1\} \tag{3,6}$$

$$z \geq 0$$

Malzeme ve parça ihtiyacı taşınma işinin taşıyıcılara dengeli bir şekilde dağıtılması için farklı taşıyıcılara atanan, bir vardiya boyunca taşınacak sepet sayılarının mutlak farklarını en azlamak için (3.1) numaralı amaç fonksiyonu yazılmıştır. Her istasyonun ihtiyacı bir taşıyıcı tarafından karşılanmalıdır bu nedenle (3.2) numaralı kısıt eklenmiştir. Malzemecilerin birbirine komşu olmayan istasyonlardan sadece birine atanması gerekmektedir, bunu sağlayan (3.3) numaralı kısıttır. (3.4) ve (3.5) numaralı kısıtlar ise z nin maksimum iş yükü mutlak farkına eşit olması için eklenmiştir. (3.6) numaralı kısıt işaret kısıtıdır.

3.1.2. İlk ana problem çözüm yaklaşımları

Matematiksel model ile yapılan denemelerde görülmüştür ki bazı problemlerde çözüm süresi yaklaşık 40 dakikayı bulmaktadır. Çözüm süresinin uzunluğundan dolayı bu problem için bir sezgisel algoritma tasarlanmıştır.

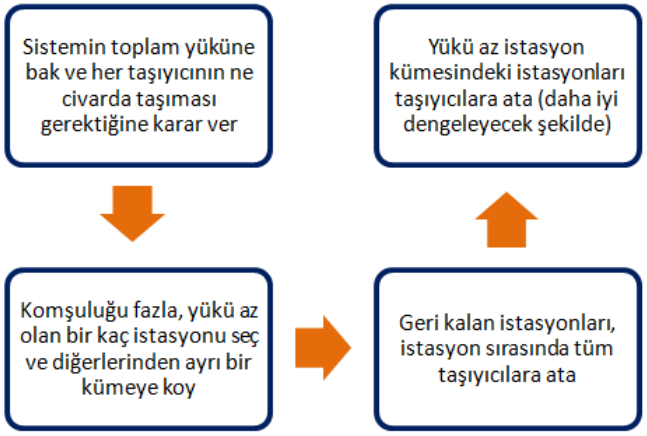
Sezgisel Algoritma

Sezgisel algoritmalar aslında günlük hayatta yol ayrımlarında sezgisel olarak seçim yapmak gibidir. Doğruluğunun ve en iyi sonucu vereceğinin hiçbir garantisi yoktur fakat karmaşık problemleri daha basite indirgeyerek hızlı bir çözüm elde edilmesine yardımcı olur. En iyi sonucu vermese de yeterince iyi bir sonuç almak ana

amaçlardandır. İlk ana problem için tasarlanan sezgisel algoritma aşağıda açıklanmıştır;

Sezgisel Algoritma Adımları-Başlangıç Çözümü

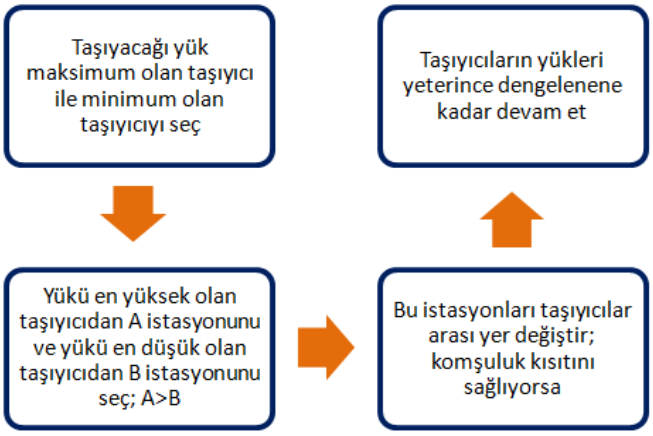
Sezgisel algoritmanın başlangıç çözümü bulma adımları Şekil 3.2'de gösterilmiştir. Sistemin toplam yükü; her bir istasyona bir vardiya boyunca götürülmesi gereken toplam malzeme ihtiyacıdır. Bu toplam ihtiyaç malzemeci sayısına bölünerek her bir malzemecinin eşit taşıdığı durumda ne kadar yük taşıdığı belirlenir. Sistemde denenen diğer parametrelere göre (hat sayısı, istasyon sayısı, seçilen yük aralığı) belirlenen bir sayıda yükü az fakat komşuluğu fazla olan istasyonlar farklı bir kümeye ayrılır. Hesaplanan ortalama yüke parametrelere göre belirlenecek bir sayı kadar az veya çok yük ataması yapıldığı zaman o taşıyıcı artık doldu kabul edilir. Ayrılan kümedeki istasyonlar yükü diğerlerinden az olan taşıyıcıya yükü çok istasyon verilecek şekilde atanır.



Şekil 3.2. Sezgisel Algoritma Başlangıç Çözümü Bulma Akış Diyagramı

Sezgisel Algoritma Adımları-İyileştirme

Sezgisel algoritmanın başlangıç çözümü bulma adımları Şekil 3.3'de gösterilmiştir. Son adımda, örneğin atama yapılan yük aralığı 1-10 seçilmiş ise taşıyıcılar arası taşınan toplam yük farkı maksimum 5 olacak şekilde atanmışsa yeterince dengeli kabul edilir.



Şekil 3.3. Sezgisel Algoritma Başlangıç İyileştirme Adımları Akış Diyagramı.

3.1.3. Kullanılan veri kümesi

Matematiksel model ve sezgisel model Çizelge 3.2.'de verilen parametreler ile çözdürülmüş ve yüzde dengesizlikler karşılaştırılmıştır. Denemeler yük ağırlıklarının uniform dağılımdan üretilmesiyle elde edilmiştir.

Çizelge 3.2. Matematiksel Model ve Sezgisel Model Verileri.

Deneme Sayısı	Yük Aralığı	Hat Sayısı	Sıra Sayısı	İstasyon Sayısı	Taşıyıcı Sayısı
10	1-20	4	8	(her sırada 4'er) 32	3
10	15-45	4	8	(her sırada 4'er) 32	3

3.1.4. Optimal ile sezgisel model sonuçlarının karşılaştırılması

Çizelge 3.2.'de gösterilen veriler ile matematiksel model ve sezgisel model çözdürülmüş olup Çizelge 3.3'de verilen sonuçlara ulaşılmıştır.

Çizelge3.3. Matematiksel Model ve Sezgisel Model Sonuçları

Yük	Deneme	Optimal % Dengesizlik	Süre	Algoritma % Dengesizlik	Süre
1-20	**1**	0,10	00:00:59:85	0,68	00:00:00:01
1-20	**2**	0,00	00:00:02:15	0,63	00:00:00:01
1-20	**3**	0,11	00:03:26:57	0,33	00:00:00:01
1-20	**4**	0,11	00:02:07:15	0,21	00:00:00:01
1-20	**5**	0,11	00:02:01:46	0,43	00:00:00:01

1-20	**6**	0,45	00:01:56:16	0,57	00:00:00:01
1-20	**7**	0,11	00:02:06:27	0,86	00:00:00:01
1-20	**8**	0,10	00:02:46:65	0,76	00:00:00:01
1-20	**9**	0,00	00:02:05:16	0,19	00:00:00:01
1-20	**10**	0,00	00:02:58:54	0,79	00:00:00:01
15-45	**1**	0,03	00:20:08:67	0,46	00:00:00:01
15-45	**2**	0,04	00:03:23:22	0,83	00:00:00:01
15-45	**3**	0,00	00:00:02:13	0,58	00:00:00:01
15-45	**4**	0,37	00:00:02:68	0,70	00:00:00:01
15-45	**5**	0,03	00:40:10:62	0,63	00:00:00:01
15-45	**6**	0,19	00:36:25:12	0,26	00:00:00:01
15-45	**7**	0,04	00:35:16:19	0,44	00:00:00:01
15-45	**8**	0,04	00:04:02:79	0,34	00:00:00:01
15-45	**9**	0,14	00:03:56:13	0,17	00:00:00:01
15-45	**10**	0,00	00:02:14:03	0,55	00:00:00:01
Ortalama		**0,0985**	**00:08:20:40**	**0,5205**	**00:00:00:01**

Çizelge 3.3'de gösterilen veriler incelendiğinde sezgisel modelin çözüm süresi matematiksel modele göre oldukça kısadır. Ortalama süre matematiksel modelde yaklaşık 8 dakikadır fakat bazı problemlerde çözüm süresi 40 dakika civarındadır.

Sezgisel algoritmada çözüm süresi saliselerle ifade edilirken iki modelin performansları kıyaslandığında ise sezgiselin % dengesizlik olarak ortalamada %0,519 verdiği görülmüştür ve bu kabul edilebilir bir dengesizlik yüzdesi olarak değerlendirilmiştir.

3.2. İkinci Ana Problem

İkinci ana problemde ana amaç bir veya birden fazla vardiya boyunca malzeme dağıtımının her bir taşıyıcı için ayrı ayrı çizelgelenmesidir. İkinci problemde ele alınan ilk amaç taşıyıcıların vardiya boyunca yaptığı malzeme besleme tur sayısının azaltılmasıdır. Montaj istasyonlarındaki stok yığılmalarının en azlanması ise kısıt olarak alınmıştır.

3.2.1 İkinci ana problem matematiksel modeli

İkinci ana problem iki ayrı amaç fonksiyonu için tasarlanmıştır. Birinci amaç fonksiyonu bir zaman dilimindeki hat başı stoğunu minimize edecek şekilde tasarlanmıştır. İkinci amaç fonksiyonu ise malzemecilerin yaptıkları toplam tur sayısının mümkün olan en düşük seviyeye çekilmesini amaçlamaktadır. Birinci modeldeki bir takım çıktılar ikinci modelde parametre olarak kullanılmıştır.

İkinci ana problem matematiksel modelinde parametreler aşağıdaki gibidir:

$P = \{1,...,n_p\}$ Malzeme kümesi

$I = \{1,...,n_I\}$ İstasyonlar kümesi

D_{jkt} $j \in P$ malzemesinin $k \in I$ istasyonunda t zaman dilimindeki talep miktarı (sepet)

F_i $i \in M$ malzemecisinin taşıyabildiği maksimum hacim $\left(m^3\right)$

V_j $j \in P$ malzemesinin bir sepetinin hacmi $\left(m^3\right)$

G_k $k \in I$ istasyonunda biriktirilebilecek maksimum stok miktarı (sepet)

S_{jk} $j \in P$ malzemesinin $k \in I$ istasyonunda bir önceki vardiyadan kalan miktarı(sepet)

K Bir vardiyada üretilebilen maksimum ürün miktarı($Big - M$)

n Bir vardiyadaki t zaman dilimlerinin sayısı

h_i $i \in M$ malzemecisinin bir t zaman diliminde yapabileceği maksimum tur sayısı

İkinci ana problem matematiksel modelinde karar değişkenleri aşağıdaki gibidir:

ϑ_{it} = $i \in M$ malzemecisinin t zaman dilimindeki tur sayısı

q_{jkt} = $j \in M$ malzemesinin $k \in I$ istasyonunda t zaman dilimi sonunda hat başındaki durumu

w_{jkt} = $j \in M$ malzemesinin $k \in I$ istasyonunda t zaman dilimi sonunda talep durumuna göre nihai hat başı stok değeri

x_{ijkt} = $i \in M$ malzemecisinin $j \in M_i$ malzemesinden $k \in I$ istasyonuna t zaman diliminde götürdüğü sepet miktarı

y_{ijktm} = i malzemecisinin j malzemesinden k istasyonuna t zaman dilimindeki m. turda taşıdığı sepet miktarı

α_{itm} = $\begin{cases} 1, & i \in M \text{ malzemecisi t diliminde } m. \text{ turu yaptıysa} \\ 0, & \text{değilse} \end{cases}$

z_{jkt} = $j \in M$ malzemesinin $k \in I$ istasyonunda t zaman dilimi sonundaki kümülatif sepet stoğu

İkinci ana problem matematiksel modelinde amaç fonksiyonu ile malzemecinin yaptığı toplam tur sayısının mümkün olan en düşük seviyeye çekilmesi amaçlanmaktadır ki bu amacın formülasyonu aşağıdadır;

$$min \sum_{i \in M} \sum_{t=1}^{n} \vartheta_{it}$$

İkinci ana problem matematiksel modelindeki kısıtlar ve açıklamaları aşağıda verilmiştir.

Taşınabilecek maksimum hacim kısıtı

Bir transpallet ile tek seferde birden fazla sepet taşınabilmektedir. Ancak tek seferde maksimum taşınabilecek hacim taşıma transpalletin özelliğine bağlıdır Bu kısıt tek seferde çoklu taşıma yapılacağı durumlarda maksimum hacmi aşacak kombinasyonları engellemektedir.

Bu kısıt aynı anda bir malzemenin birden çok sepetinin taşınması veya farklı malzemelerin kombinasyonlarından oluşan bir taşımada taşınan hacmin taşınabilecek maksimum hacimden düşük ya da eşit olmasını sağlar.

$$\sum_{k \in I} \sum_{j \in P} y_{ijktm} V_j \leq F_i \qquad \forall i \in M, t = 1,...,n, m = 1,...,h_i$$

Talebin karşılanmasını sağlayan kısıt

Bu kısıt t zaman diliminde ihtiyaç duyulacak malzemenin, t zaman diliminden önce gerekli istasyona taşınmasını sağlıyor.

$$z_{jk(t-1)} \geq D_{jkt} \qquad \forall j \in P, k \in I, t = 1,...,n$$

İstasyondaki birikmenin belli bir seviyenin altında kalmasını sağlayan kısıt

Her istasyon için izin verilen bir stok kapasitesi bulunmaktadır. Bu kısıt herhangi bir t zaman dilimi sonunda, istasyon başındaki birikmenin o istasyona özgü maksimum sınırın aşılmasını önlemektedir. Bu problemde bu değer 4, 5 ve 6 için denemeler yapılmıştır.

$$\sum_{j \in P} w_{jkt} \leq G_k \qquad \forall k \in I, t = 1,...,n$$

Bir turda bir istasyona malzeme gönderme kararı alınmadıysa o turda o istasyona malzeme gönderilmemesini sağlayan kısıt

Bir t dilimindeki bir turda bir istasyona belli bir malzeme gönderilecekse, öncelikle gönderme kararı alınması gerekmektedir.

$$\sum_{j \in P_i} \sum_{k \in I} y_{ijktm} \leq \alpha_{itm} K \qquad \forall i \in M, t = 1,...,n, m = 1,...,h_i$$

Bir t zaman dilimindeki tur sayısını veren bağıntı

Bir malzemeci bir t zaman diliminde birden çok tur yapabilmektedir. Bu bağıntı ile bütün turlar üzerinden t zaman dilimindeki toplam tur sayısı çıkarılmaktadır.

$$\vartheta_{it} = \sum_{m=1}^{h_i} \alpha_{itm} \qquad \forall i \in M, t = 1,...,n$$

Bir t zaman dilimi sonundaki birikmenin bağıntısı

Bir t zaman dilimi sonundaki hat başı stok, bir önceki zaman diliminden kalan stoğa o zaman diliminde taşınan miktarın eklenmesi ve o zaman diliminde tüketilen miktarın bu toplamdan düşülmesiyle bulunur.

$$z_{jk(t-1)} + \sum_{i \in M} x_{ijkt} - D_{jkt} = z_{jkt} \qquad \forall j \in P, k \in I, t = 1,...,n$$

Vardiya başlangıç stoku kısıtı

Vardiya başlangıcındaki istasyon başındaki stok bir önceki vardiya sonundaki stoğa eşittir**.**

$$z_{jk0} = S_{jk} \qquad \forall j \in P, k \in I$$

Bir t zaman diliminde taşınan malzemeyi veren bağıntı

Bir t zaman diliminde taşınan toplam malzeme o zaman diliminde yapılan tüm turlarda taşınan toplam malzeme miktarına eşittir.

$$\sum_{m=1}^{h_i} y_{ijktm} = x_{ijkt} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

Matematiksel Model Tutarlılığını Sağlamak için Eklenen Kısıtlar

$$f_{ijtm} \le \sum_{k \in I} y_{ijktm} * BigM \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$\sum_{k \in I} y_{ijktm} \le f_{ijtm} * BigM \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$\alpha_{itm} * BigM \ge \sum_{j \in P} f_{ijtm} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$f_{ijtm} + f_{idtm} \le 1 \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n, st[j] = st[d]$$

Model bütün halinde aşağıda verilmiştir;

$$min \sum_{i \in M} \sum_{t=1}^{n} \vartheta_{it}$$

Kısıtlar:

$$w_{jkt} = z_{jkt} - q_{jkt} \qquad \forall j \in P, k \in I, t = 1,...,n$$

$$\sum_{j \in P} w_{jkt} \le G_k \qquad \forall k \in I, t = 1,...,n$$

$$\sum_{k \in I} \sum_{j \in P} y_{ijktm} V_j \le F_i \qquad \forall i \in M, t = 1,...,n, m = 1,...,h_i$$

$$\sum_{j \in P_i} \sum_{k \in I} y_{ijktm} \le \alpha_{itm} K \qquad \forall i \in M, t = 1,...,n, m = 1,...,h_i$$

$$\vartheta_{it} = \sum_{m=1}^{h_i} \alpha_{itm} \qquad \forall i \in M, t = 1,...,n$$

$$z_{jk(t-1)} + \sum_{i \in M} x_{ijkt} - D_{jkt} = z_{jkt} \qquad \forall j \in P, k \in I, t = 1,...,n$$

$$z_{jk0} = S_{jk} \qquad \forall j \in P, k \in I$$

$$\sum_{m=1}^{h_i} y_{ijktm} = x_{ijkt} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$f_{ijtm} \le \sum_{k \in I} y_{ijktm} * BigM \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$\sum_{k \in I} y_{ijktm} \le f_{ijtm} * BigM \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$\alpha_{itm} * BigM \ge \sum_{j \in P} f_{ijtm} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$f_{ijtm} + f_{idtm} \le 1 \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n, st[j] = st[d]$$

$$x_{ijkt} \ge 0, \text{tamsayı} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n$$

$$z_{jkt} \ge 0 \qquad \forall j \in P, k \in I, t = 0,...,n$$

$$\alpha_{itm} \in \{0,1\} \qquad \forall i \in M, t = 1,...,n, m = 1,...,h_i$$

$$y_{ijktm} \ge 0, \text{tamsayı} \qquad \forall i \in M, j \in P, k \in I, t = 1,...,n, m = 1,...,h_i$$

$$\vartheta_{it} \ge 0, \text{tamsayı} \qquad \forall i \in M, t = 1,...,n$$

3.2.2. Problem Karmaşıklığı

Bu kısımda problemimizin özel bir halinin tek boyutlu kutulama (bin-packing) problemi olduğunu göstereceğiz. Tek boyutlu kutulama probleminin NP-hard olduğu Coffman *et al.* (1997) gösterilmiştir. [12] Tek boyutlu kutulama probleminde elimizde tamsayı olarak s_i boyutunda n adet obje vardır. Problem tüm objelerin m adet, kapasitesi B olan kutulara dağıtmasıdır. m in en azlanması problemin objektifidir.

Bizim problemimizin şu özel halini ele alalım; vardiya tek periyottan oluşsun, ve tur sayısı limiti sonsuz olsun. Turlarla ilgili süre limiti de sonsuz olsun. Her tur için olan kapasite kısıtı B, her istasyonun her malzemeden talepleri de tamsayı olarak s_i kutu olsun. Bu şartlarda oluşan tur sayısı azaltma problemimizin özel hali kutulama problemine dönüşmüş olur ve dolayısıyla bizim problemimizde Np-zordur.

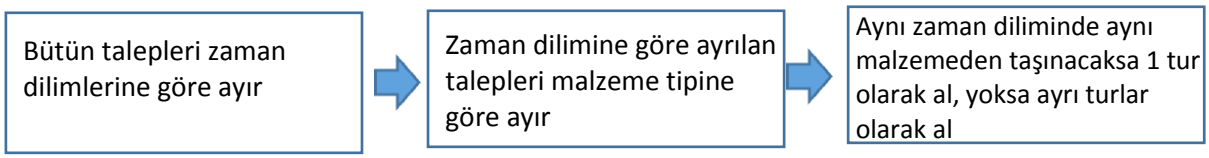
3.2.3. İkinci ana problem çözüm yaklaşımları

Sezgisel Algoritma

Ele alınan problem NP-Zor bir problem olduğundan büyük problemlerin çözümü için sezgisel bir algoritma geliştirilmiştir. İkinci ana problemin amacı olan bir veya birden fazla vardiya boyunca malzeme dağıtımının her bir taşıyıcı için ayrı ayrı çizelgelenmesi işlemi matematiksel model ile çözüldüğünde çok uzun zaman almaktadır ve problemimiz Np-zor sınıfındadır. Bu nedenle çözüm süresini kısaltacak ve matematiksel model ile yeterince yakın sonuçlar verebilecek bir sezgisel algoritma geliştirilmiştir. Sezgisel algoritma adımları aşağıda açıklanmıştır.

Sezgisel Algoritma Adımları-Başlangıç Çözümü

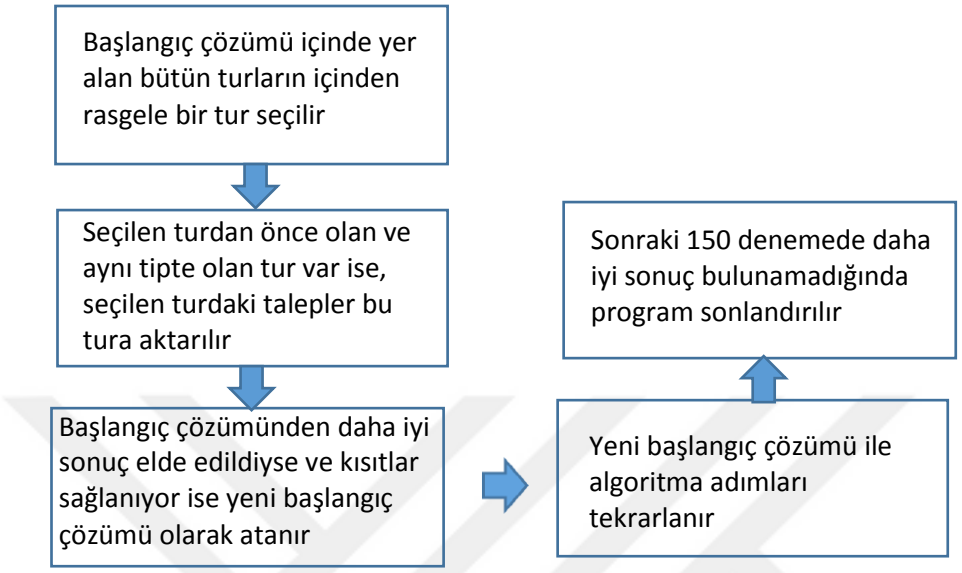
Sezgisel algoritmanın başlangıç çözümü bulma adımları Şekil 3.4'de gösterilmiştir.



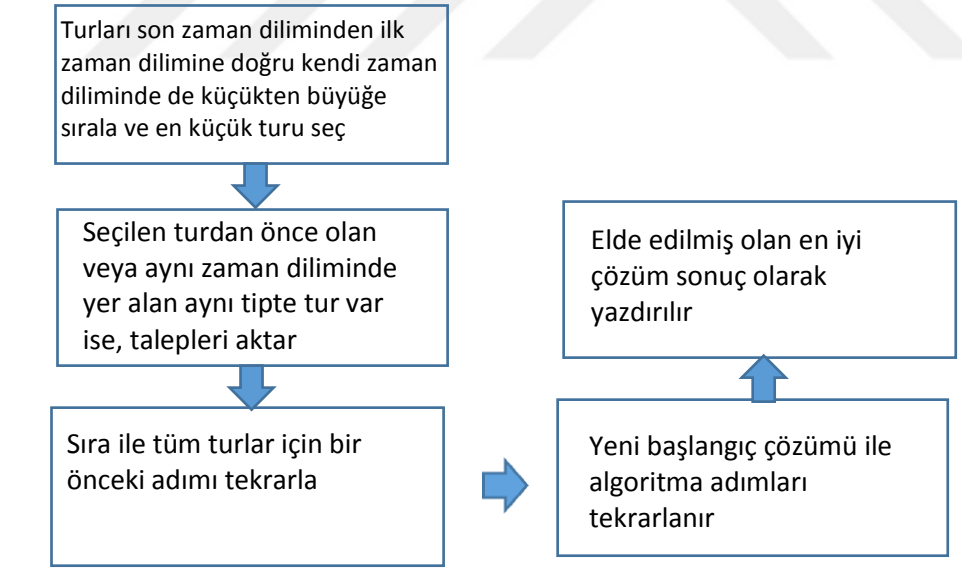
Şekil 3.4. Sezgisel Algoritma Başlangıç Çözümü Bulma Akış Diyagramı

Sezgisel Algoritma Adımları-İyileştirme Adımları

Sezgisel algoritmanın başlangıç çözümü bulma adımları Şekil 3.5 ve 3.6'da gösterilmiştir.



Şekil 3.5. Sezgisel Algoritma İyileştirme Adımları Akış Diyagramı



Şekil 3.6. Sezgisel Algoritma İyileştirme Adımları Akış Diyagramı

Bu problemde test datası olarak bir bulaşık makinesi üretim fabrikasının verileri ve bu veriden türetilen problemler kullanılmıştır. Bir vardiya her biri birer saatten oluşan 7 zaman dilimi içerir. Her bir zaman dilimi için, her bir istasyonun, her bir

malzemeden ne kadar ihtiyacı olduğu bilinmektedir üretim planına göre hesaplanır ve bilinmektedir.

Bir gündeki zaman dilimi sayısı 7, bir zaman diliminde bir taşıyıcının yapabileceği maksimum tur sayısı h, istasyon sayısı I, taşıyıcı sayısı M, hat başında birikebilecek maksimum sepet sayısı ise G ile gösterilmektedir. Yapılan denemelerde kullanılan parametre değerleri ve istasyon talepleri için orijinal verinin hangi aralıklarda değiştirilerek tekrar deneme yapıldığı Çizelge 3.4'te verilmiştir.

Fabrikadan alınan, her bir istasyonda hangi malzemeden hangi zaman diliminde sepet cinsinden ne kadar ihtiyaç olduğu bilgisi ile model çözdürülmüştür. Bu orijinal veri kümesinin değerleri %20 ve %40 oranında arttırılarak, %20 ve %40 oranında azaltılarak yeni veri kümeleri oluşturulmuş ve problem çözdürülmüştür. daha sonra orijinal istasyon talepleri (-%30,+%30) aralığında rasgele üretilen değerler ile çarpılarak 30 farklı veri kümesi oluşturulmuş ve amaç fonksiyonu değerleri incelenmiştir.

Çizelge 3.4.Yapılan Denemelerde Kullanılan Değerler

Deneme Sayısı	**N**	**h**	**I**	**G**	**İstasyon Talepleri Veri Aralığı**
1	7	6	52	4,5,6	Orijinal Veri
1	7	6	52	4,5,6	Orijinal Veri*1,1
1	7	6	52	4,5,6	Orijinal Veri*1,2
1	7	6	52	4,5,6	Orijinal Veri*0,9
1	7	6	52	4,5,6	Orijinal Veri*0,8
30	7	6	52	4,5,6	Orijinal Veri* (-%30,+%30)

3.2.4. Matematiksel model ve sezgisel algoritma sonuçlarının karşılaştırılması

Çizelge 3.4'de belirtilen değerlerle hem matematiksel model hem de sezgisel algoritma çözdürülmüştür. Matematiksel modelin ve sezgisel algoritmanın her bir veri kümesi için hat başı maksimum stok miktarı 6, 5 ve 4 iken çözüm zamanları ve

amaç fonksiyon değeri olan tur sayısı sırasıyla Çizelge 3.5, Çizelge 3.6 ve Çizelge 3.7'de gösterilmiştir.

Matematiksel Model ve Sezgisel Algoritma Deneme Sonuçları

Hat başı maksimum stok miktarı 6 için yapılan deneme sonuçları aşağıdaki gibidir;

Çizelge 3.5. Hat Başı Stok Miktarı 6 için Deneme Sonuçları

Veri Aralığı	Deneme Sayısı	Algoritma Çözüm Zamanı	Algoritma Amaç Fonk. Değeri	Mat. Model Çözüm Zamanı	Optimal Amaç Fonk. Değeri	Sapma
Orijinal Veri	Deneme 1	00:00.00:02	9	00:45:27:19	9	0,00%
Orijinal Veri*1,2	Deneme 1	00:00.00:02	10	00:39:56:73	10	0,00%
Orijinal Veri*1,4	Deneme 1	00:00.00:02	11	00:43:59:61	10	10,00%
Orijinal Veri*0,8	Deneme 1	00:00.00:02	9	00:43:41:96	9	0,00%
Orijinal Veri*0,6	Deneme 1	00:00.00:02	9	00:47:52:20	8	12,50%
Orijinal Veri* (-%30,+%30)	Deneme 1	00:00.00:02	10	00:55:20:80	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 2	00:00.00:02	9	00:51:57:06	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 3	00:00.00:02	9	00:56:20:87	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 4	00:00.00:02	9	00:33:11:78	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 5	00:00.00:02	10	00:32:57:97	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 6	00:00.00:02	9	00:25:58:68	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 7	00:00.00:02	9	00:41:37:55	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 8	00:00.00:02	9	00:25:42:72	9	0,00%

Orijinal Veri* (-%30,+%30)	Deneme 9	00:00.00:02	9	00:25:44:89	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 10	00:00.00:02	9	00:25:29:81	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 11	00:00.00:02	10	00:39:56:61	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 12	00:00.00:02	9	00:40:52:67	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 13	00:00.00:02	9	00:42:23:14	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 14	00:00.00:02	11	0:39:57:80	9	22,22%
Orijinal Veri* (-%30,+%30)	Deneme 15	00:00.00:02	9	00:40:50:46	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 16	00:00.00:02	9	00:40:56:99	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 17	00:00.00:02	11	00:38:46:72	9	22,22%
Orijinal Veri* (-%30,+%30)	Deneme 18	00:00.00:02	9	00:39:12:27	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 19	00:00.00:02	9	00:41:26:67	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 20	00:00.00:02	9	00:24:50:79	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 21	00:00.00:02	10	00:35:26:82	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 22	00:00.00:02	9	00:24:53:64	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 23	00:00.00:02	9	00:42:29:95	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 24	00:00.00:02	10	00:25:22:01	9	11,11%

Orijinal Veri* (-%30,+%30)	Deneme 25	00:00.00:02	10	00:36:18:90	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 26	00:00.00:02	9	00:23:49:75	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 27	00:00.00:02	10	00:41:25:63	9	11,11%
Orijinal Veri* (-%30,+%30)	Deneme 28	00:00.00:02	9	00:40:12:08	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 29	00:00.00:02	9	00:38:16:99	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 30	00:00.00:02	11	00:24:52:75	9	22,22%
Ortalama		00:00.00:02	9,46	00:38:02:86	9,03	4,77%

Tüm sonuçlar incelendiğinde sezgisel algoritmanın bulduğu ortalama tur sayısı 9,46 iken optimal sonuçların ortalaması 9,03'tür. Yani optimalden ortalama sapma %4,77. Toplam 35 problemin 23 ünde sezgisel optimal sonucu bulmuştur. Ayrıca sezgisel matematiksel modele göre çok kısa sürede sonuçlar vermektedir.

Hat başı maksimum stok miktarı 5 için yapılan deneme sonuçları aşağıdaki gibidir;

Çizelge 3.6. Hat Başı Stok Miktarı 5 için Deneme Sonuçları

Veri Aralığı	**Deneme Sayısı**	**Algoritma Çözüm Zamanı**	**Algoritma Amaç Fonk. Değeri**	**Matmatiksel Model Çözüm Zamanı**	**Optimal Amaç Fonk. Değeri**	**Sapma**
Orijinal Veri	Deneme 1	00:00.00:02	9	00:39:28:64	9	0,00%
Orijinal Veri*1,2	Deneme 1	00:00.00:02	10	00:46:19:93	10	0,00%
Orijinal Veri*1,4	Deneme 1	00:00.00:02	11	00:42:53:12	10	10,00%
Orijinal Veri*0,8	Deneme 1	00:00.00:02	9	00:44:57:97	9	0,00%
Orijinal Veri*0,6	Deneme 1	00:00.00:02	9	00:53:41:61	8	12,50%

Orijinal Veri* (-%30,+%30)	Deneme 1	00:00.00:02	10	00:35:29:01	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 2	00:00.00:02	9	00:32:28:56	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 3	00:00.00:02	9	00:33:24:56	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 4	00:00.00:02	9	00:53:18:34	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 5	00:00.00:02	10	00:23:59:82	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 6	00:00.00:02	9	00:41:24:83	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 7	00:00.00:02	9	00:41:32:09	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 8	00:00.00:02	9	00:25:40:41	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 9	00:00.00:02	9	00:41:56:74	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 10	00:00.00:02	9	00:41:26:78	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 11	00:00.00:02	10	00:43:56:91	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 12	00:00.00:02	9	00:40:48:26	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 13	00:00.00:02	9	00:26:13:67	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 14	00:00.00:02	11	00:23:54:86	10	10,00%
Orijinal Veri* (-%30,+%30)	Deneme 15	00:00.00:02	9	00:31:54:96	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 16	00:00.00:02	9	00:38:53:68	9	0,00%

Orijinal Veri* (-%30,+%30)	Deneme 17	00:00.00:02	11	00:40:51:08	10	10,00%
Orijinal Veri* (-%30,+%30)	Deneme 18	00:00.00:02	9	00:39:42:99	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 19	00:00.00:02	9	00:25:54:45	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 20	00:00.00:02	9	00:38:15:24	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 21	00:00.00:02	10	00:42:35:72	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 22	00:00.00:02	9	00:24:26:53	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 23	00:00.00:02	9	00:42:15:82	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 24	00:00.00:02	10	00:25:49:82	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 25	00:00.00:02	10	00:40:16:76	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 26	00:00.00:02	9	00:38:35:78	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 27	00:00.00:02	10	00:25:46:97	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 28	00:00.00:02	9	00:34:42:57	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 29	00:00.00:02	9	00:23:19:78	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 30	00:00.00:02	11	00:25:24:39	9	22,22%
Ortalama		00:00.00:02	9,46	00:36:18:57	9,28	1,85%

Bu sonuçlar incelendiğinde sezgisel algoritmanın bulduğu ortalama tur sayısı 9,46 iken optimal sonuçların ortalaması 9,28'dir ve optimalden ortalama sapma %1,85 dir. 34 problemin 30 unda optimal çözüm bulunmuştur.

Hat başı maksimum stok miktarı 4 için yapılan deneme sonuçları aşağıdaki gibidir;

Çizelge 3.7. Hat Başı Stok Miktarı 4 için Deneme Sonuçları

Veri Aralığı	Deneme Sayısı	Algoritma Çözüm Zamanı	Algoritma Amaç Fonk. Değeri	Matematiksel Model Çözüm Zamanı	Optimal Amaç Fonk. Değeri	Sapma
Orijinal Veri	Deneme 1	00:00.00:02	9	00:52:48:49	9	0,00%
Orijinal Veri*1,2	Deneme 1	00:00.00:02	10	00:33:26:33	10	0,00%
Orijinal Veri*1,4	Deneme 1	00:00.00:02	11	00:51:49:31	10	10,00%
Orijinal Veri*0,8	Deneme 1	00:00.00:02	9	00:53:41:23	9	0,00%
Orijinal Veri*0,6	Deneme 1	00:00.00:02	9	00:55:43:56	8	12,50%
Orijinal Veri* (-%30,+%30)	Deneme 1	00:00.00:02	10	00:35:01:00	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 2	00:00.00:02	9	00:49:48:25	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 3	00:00.00:02	10	00:55:35:23	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 4	00:00.00:02	10	00:33:56:51	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 5	00:00.00:02	10	00:39:18:53	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 6	00:00.00:02	9	00:41:54:62	9	0,00%

Orijinal Veri* (-%30,+%30)	Deneme 7	00:00.00:02	9	00:40:24:86	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 8	00:00.00:02	9	00:43:27:38	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 9	00:00.00:02	9	00:42:53:54	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 10	00:00.00:02	9	00:27:55:12	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 11	00:00.00:02	10	00:24:53:49	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 12	00:00.00:02	9	00:31:56:29	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 13	00:00.00:02	9	00:37:25:69	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 14	00:00.00:02	11	00:39:45:76	10	10,00%
Orijinal Veri* (-%30,+%30)	Deneme 15	00:00.00:02	10	00:41:26:75	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 16	00:00.00:02	9	00:31:05:57	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 17	00:00.00:02	11	00:28:48:21	10	10,00%
Orijinal Veri* (-%30,+%30)	Deneme 18	00:00.00:02	9	00:39:32:78	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 19	00:00.00:02	10	00:26:38:49	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 20	00:00.00:02	10	00:25:42:13	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 21	00:00.00:02	10	00:38:19:47	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 22	00:00.00:02	9	00:40:03:17	9	0,00%

Orijinal Veri* (-%30,+%30)	Deneme 23	00:00.00:02	9	00:37:48:63	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 24	00:00.00:02	10	00:25:37:98	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 25	00:00.00:02	10	00:40:28:02	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 26	00:00.00:02	10	00:26:59:17	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 27	00:00.00:02	10	00:29:68:34	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 28	00:00.00:02	9	00:39:45:75	9	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 29	00:00.00:02	10	00:39:46:97	10	0,00%
Orijinal Veri* (-%30,+%30)	Deneme 30	00:00.00:02	11	00:40:59:62	9	22,22%
Ortalama		00:00.00:02	9,66	00:38:51:42	9,49	%1,85

Tüm sonuçlar incelendiğinde sezgisel algoritmanın bulduğu ortalama tur sayısı 9,66 iken optimal sonuçların ortalaması 9,49'dur ve optimalden ortalama sapma yine %1,85 dir. 34 problemin 30 unda optimal çözüm bulunmuştur.

Yukarıdaki deneme sonuçları ve sürelere göre sezgiselin oldukça etkin olduğu değerlendirilmiştir.

Çizelge 3.8'de hat başı maksimum stok miktarı 6, 5 ve 4 alındığında matematiksel model amaç fonksiyonu değerleri verilmiştir. Hat başı maksimum stok miktarı azaltılması ile bazı denemelerde yapılan tur sayısının arttığı gözlemlenmiştir. Buna karşılık, Çizelge 3.9'da hat başı maksimum stok miktarı 6, 5 ve 4 alındığında sezgisel algoritma amaç fonksiyonu değerlerine bakıldığında bu değişim sonucunda tur sayısında bir değişiklik olmadığı görülmektedir.

Çizelge 3.8 Matematiksel Model Karşılaştırma

Veri Aralığı	**Deneme Sayısı**	**G=6**	**G=5**	**G=4**
Orijinal Veri	Deneme 1	9	9	9
Orijinal Veri*1,2	Deneme 1	10	10	10
Orijinal Veri*1,4	Deneme 1	10	10	10
Orijinal Veri*0,8	Deneme 1	9	9	9
Orijinal Veri*0,6	Deneme 1	8	8	8
Orijinal Veri* (-%30,+%30)	Deneme 1	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 2	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 3	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 4	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 5	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 6	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 7	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 8	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 9	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 10	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 11	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 12	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 13	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 14	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 15	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 16	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 17	9	10	10

Orijinal Veri* (-%30,+%30)	Deneme 18	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 19	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 20	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 21	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 22	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 23	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 24	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 25	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 26	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 27	9	10	10
Orijinal Veri* (-%30,+%30)	Deneme 28	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 29	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 30	9	9	9

Çizelge 3.9 Sezgisel Model Karşılaştırma

Veri Aralığı	**Deneme Sayısı**	**G=6**	**G=5**	**G=4**
Orijinal Veri	Deneme 1	9	9	9
Orijinal Veri*1,2	Deneme 1	10	10	10
Orijinal Veri*1,4	Deneme 1	11	11	11
Orijinal Veri*0,8	Deneme 1	9	9	9
Orijinal Veri*0,6	Deneme 1	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 1	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 2	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 3	9	9	10

Orijinal Veri* (-%30,+%30)	Deneme 4	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 5	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 6	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 7	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 8	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 9	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 10	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 11	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 12	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 13	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 14	11	11	11
Orijinal Veri* (-%30,+%30)	Deneme 15	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 16	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 17	11	11	11
Orijinal Veri* (-%30,+%30)	Deneme 18	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 11	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 12	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 13	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 14	11	11	11
Orijinal Veri* (-%30,+%30)	Deneme 15	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 16	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 17	11	11	11
Orijinal Veri* (-%30,+%30)	Deneme 18	9	9	9

Orijinal Veri* (-%30,+%30)	Deneme 19	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 20	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 21	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 22	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 23	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 24	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 25	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 26	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 27	10	10	10
Orijinal Veri* (-%30,+%30)	Deneme 28	9	9	9
Orijinal Veri* (-%30,+%30)	Deneme 29	9	9	10
Orijinal Veri* (-%30,+%30)	Deneme 30	11	11	11

4.DEĞERLENDİRMELER

Sonuçlar incelendiğinde, sisteme ait orijinal verilerle çıkan günlük toplam tur sayısı değerini sezgisel algoritma denemelerin çoğunda optimal ile aynı sonucu bulmuştur. Sezgiselin tüm denemelerde optimalden en fazla bir tur fazla sonuç verdiği görülmüştür. Tüm denemeler üzerinden optimalden yüzde sapma değeri %2,82 ki bu değerler sezgiselin etkinliğini göstermektedir.

Çözüm sürelerinde baktığımızda matematiksel model çözüm süresi 20 dakika ile 56 dakika arasında sürmektedir. Sezgisel ise tüm problemlerde iki saniyeden az sürmüştür. Bu çizelgeleme probleminin her vardiya başında çözüleceği göz önüne alınırsa, matematiksel modelin kullanımının zorluk oluşturacağı bellidir. Ayrıca genelde üretim ortamlarının dinamik yapısı ve gereksinimlerin rassal değişimi göz önüne alındığında çalışmamızda sunduğumuz gibi hızlı ve etkin çalışsan bir yaklaşımın önemi daha da artacaktır.

Sonuç olarak bu çalışma varsayımları itibariyle daha önce çok az çalışılmış montaj hattı beslemesinin çizelgelenmesi problemine hem modelleme hem de çözüm yaklaşımı itibariyle yani yaklaşımlar getirmiş bir çalışmadır. Çalışmamız hem pratik bir problemi ele almış ve hem de literatürdeki önemli bir boşluğu doldurmaya katkı sağlamıştır.

5.KAYNAKLAR

[1]**WILD, Rey**, "Mass Production Management", John Wiley&Sons Ltd., New York, 1972.

[2]**KANSU KARACA,** Murat, "Montaj Hatları", Süleyman Demirel Üniversitesi İktisadi ve İdari Bilimler Fakültesi, 1996

[3] **SCHOLL, A., BECKER, C.,** State-of-The-Art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing, European Journal of Operational Research, 168:666-693, 2004.

[4]**BECKER, C., SCHOLL, A**., A Survey on Problems ve Methods in Generalized Assembly Line Balancing. European Journal of Operations Research, 168:694-715, 2006.

[5]**BOYSEN, Nils, FLIEDNER, Malte, SCHOLL, Armin**, "Sequencing Mixed-Model Assembly Lines: Survey, Classification and Model Critique", 2007

[6]**GOLZ, Jenny, GUJJULA, Rico, GÜNTHER, Hans-Otto, RINDERER, Stefan, ZIEGLER, Marcus**, Department of Production Management, Technical University of Berlin, 2011

[7]**BATTINI, Daria, FACCIO, Maurizio, PERSONA, Alessandro, SGARBOSSA, Fabio,** "Design of the Optimal Feeding Policy in an Aseembly System", Department of Management and Engineering, University of Padova, Stradella San Nicola, Vicenza, Italy, 2009

[8]**ANDERSON, R. L.** (1953). Recent Advances in Finding Best Operating Conditions. J. Amer. Statist. Assoc. 48 789-798.

[9]**KARNOPP, D. C.** (1963). Random Search Techniques for Optimization Problems. Automatica. 1 111-121.

[10]**RASTRIGIN, L. A.** (1963). The Convergence of the Random Search Method in the Extremal Control of a Many-Parameter System. Automat. Remote Control. 24. 1337-1342.

[11]**NEUMANN, F., WEGENER, I.,** Randomized local search, evolutionary algorithms, and the minimum spanning tree problem, Algorithms and Complexity, Max-Planck-Institut fur Informatik, 66123 Saarbr¨ucken, Germany,2006

[12]**COFFMAN EG, GAREY MR, JOHNSON** DS (1997) Approximation algorithms for bin-packing - a survey. In Approximation Algorithms for NP-hard Problems, PWS Publishing Company, Boston: 46-93

EKLER

EK 1: Kullanılan Veri Dosyası Görünümü, Parametreler

EK 2: Kullanılan Veri Dosyası Görünümü, Talepler

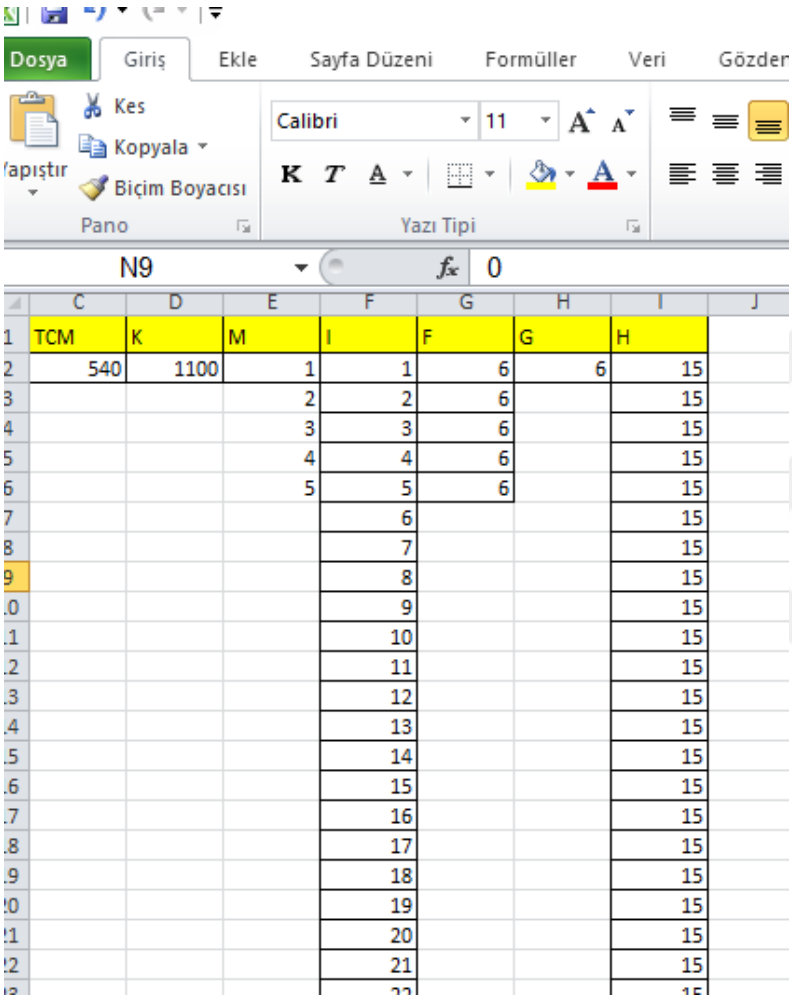
EK 3: İlk Problem Matematiksel Model

EK 4: İlk Problem Sezgisel Algoritma

EK 5: İkinci Problem Matematiksel Model

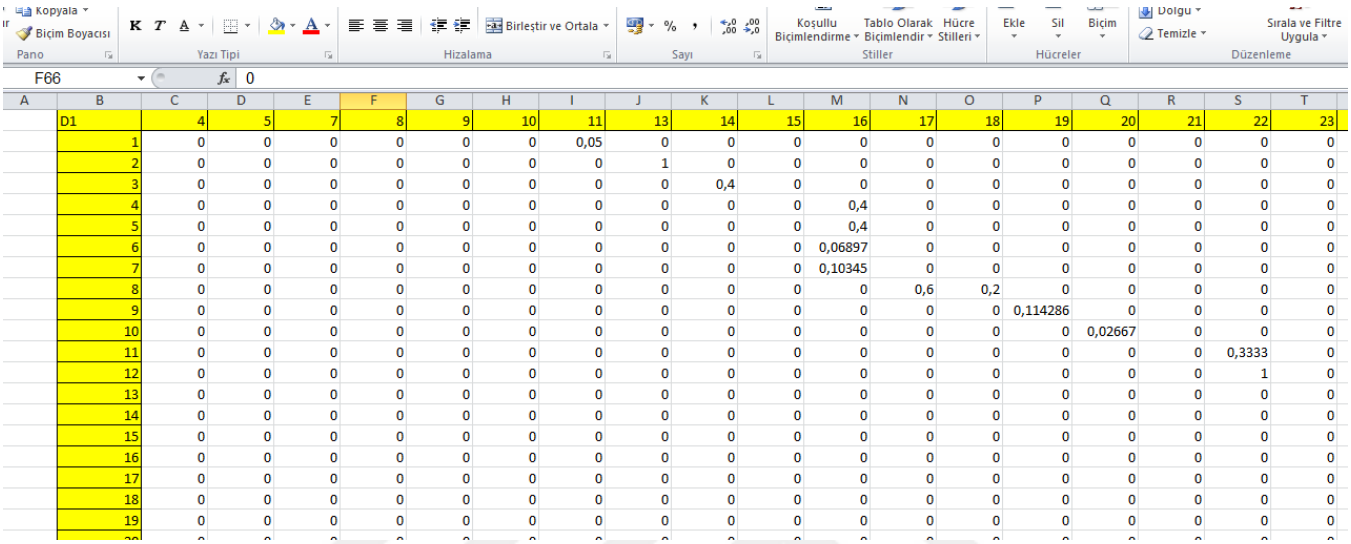
EK 6: İkinci Problem Sezgisel Algoritma

EK 1



	C	D	E	F	G	H	I	J
1	TCM	K	M	I	F	G	H	
2	540	1100	1	1	6	6	15	
3			2	2	6		15	
4			3	3	6		15	
5			4	4	6		15	
6			5	5	6		15	
7				6			15	
8				7			15	
9				8			15	
10				9			15	
11				10			15	
12				11			15	
13				12			15	
14				13			15	
15				14			15	
16				15			15	
17				16			15	
18				17			15	
19				18			15	
20				19			15	
21				20			15	
22				21			15	

EK 2



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
		D1	4	5	7	8	9	10	11	13	14	15	16	17	18	19	20	21	22	23
		1	0	0	0	0	0	0	0,05	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0,4	0	0	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0	0	0	0	0,4	0	0	0	0	0	0	0
		5	0	0	0	0	0	0	0	0	0	0	0,4	0	0	0	0	0	0	0
		6	0	0	0	0	0	0	0	0	0	0	0,06897	0	0	0	0	0	0	0
		7	0	0	0	0	0	0	0	0	0	0	0,10345	0	0	0	0	0	0	0
		8	0	0	0	0	0	0	0	0	0	0	0	0,6	0,2	0	0	0	0	0
		9	0	0	0	0	0	0	0	0	0	0	0	0	0	0,114286	0	0	0	0
		10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,02667	0	0	0
		11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0,3333	0
		12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
		13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

EK 3

```
nt Nbistasyon=...;
 int Nbmalzemeci=...;

 range istasyon=1..Nbistasyon;
 range malzemeci=1..Nbmalzemeci;

 float K[istasyon]=...;
 int a[istasyon][istasyon]=...;

dvar boolean x[istasyon][malzemeci];
 //dvar float+ x[istasyon][malzemeci];
 dvar float+ z;

 minimize z;
 subject to{
   forall(j in malzemeci, l in malzemeci : j!=l){
     sum(i in istasyon)K[i]*x[i][j] - sum(i in istasyon)K[i]*x[i][l]<=z;
     -sum(i in istasyon)K[i]*x[i][j] + sum(i in istasyon)K[i]*x[i][l]<=z;
   }

   forall(i in istasyon)
     sum(j in malzemeci)x[i][j]==1;

   forall(i in istasyon,j in malzemeci, k in istasyon : i!=k)
     x[i][j]*(1-a[i][k])+x[k][j]<=1;



   }

Nbistasyon=32;
 Nbmalzemeci=3;


 K=[38,16,44,39,31,37,40,33,42,15,39,32,18,32,21,22,30,17,22,26,29,23,18,
23,20,37,23,23,18,15,34,27];




 a=[[0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0], //1
    [1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//2
```

```
[1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//3
[1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//4
[1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//5
[1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//6
[1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//7
[1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0],//8
[1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//9
[1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//10
[1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//11
[1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,,1,1,1,,1,1,1],//12
[1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//13
[1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//14
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//15
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//16
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1],//17
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1],//18
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1],//19
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1],//20
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1],//21
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1],//22
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1],//23
[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1],//24
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1],//25
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1],//26
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1],//27
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1],//28
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1],//29
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],//30
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0],//31
[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0]];//32
```

EK 4

```
package DishwasherFactory;
import java.util.ArrayList;
import java.util.Random;
public class AssemblyLine
{

   String LineName;
   Band LeftBand;
   Band RightBand;

   // An Assembly Line inside the system
   public AssemblyLine(String LineName, int StationCountLeft, int
StationCountRight)
   {
      this.LineName = LineName;
      LeftBand = new Band(StationCountLeft);
      RightBand = new Band(StationCountRight);
   }

   // Each Line has 2 bands left and right
   class Band
   {
      Random generator = new Random();
      int rand;
      int total = 0;
      int StationCount;
      int AdjCount;
      ArrayList<Integer> Adjacencies = new ArrayList<Integer>();
      ArrayList<Station> Stations;
      Band(int StationCount)
      {
        this.StationCount = StationCount;
        Stations = new ArrayList<Station>();
        // Create the stations here
        for(int i = 0; i < StationCount; i++)
        {
            // Create small job 1-20
            rand = generator.nextInt(30) + 15;
            Stations.add(new Station(i,rand));
        }
      }

      void CalculateAdjCount()
      {
         AdjCount = Adjacencies.size();
```

```
    }

    void PrintStationLoads()
    {
        int total = 0;
        System.out.println("Band has: " + StationCount + " stations.");
        for(int i = 0; i < Stations.size(); i++)
        {
            System.out.println("Weight: " + Stations.get(i).requiredWeight);
            total += Stations.get(i).requiredWeight;
        }
        this.total = total;
        System.out.println("Total: " + total);
        System.out.println();
    }

    // Each band can have a chosen number of stations on it
    class Station
    {
        int StationNumber;
        int requiredWeight;
        boolean excluded;
        Station(int StationNumber, int requiredWeight)
        {
            this.StationNumber = StationNumber;
            this.requiredWeight = requiredWeight;
            this.excluded = false;
        }

        int getRequiredWeight()
        {
            return this.requiredWeight;
        }

        void FillLoad(int load)
        {
            this.requiredWeight -= load;
            if(this.requiredWeight < 0)
            {
                this.requiredWeight = 0;
            }
        }
    }
}

public void printAssemblyLine()
{
```

```
        System.out.println("Assebmly Line: " + this.LineName);
        System.out.println("Left Band has: " + this.LeftBand.StationCount + "
stations.");
        System.out.print("Adjecent to: ");
        for(int i = 0; i < this.LeftBand.Adjacencies.size();i++)
        {
            if(this.LeftBand.Adjacencies.get(i)%2 == 0)
            {
                System.out.print(this.LeftBand.Adjacencies.get(i)/2 + "L ");
            }
            else
            {
                System.out.print(this.LeftBand.Adjacencies.get(i)/2 + "R ");
            }
            //System.out.print(this.LeftBand.Adjacencies.get(i));
        }
        System.out.println();

        System.out.println("Right Band has: " + this.RightBand.StationCount + "
stations.");
        System.out.print("Adjecent to: ");
        for(int i = 0; i < this.RightBand.Adjacencies.size();i++)
        {
            if(this.RightBand.Adjacencies.get(i)%2 == 0)
            {
                System.out.print(this.RightBand.Adjacencies.get(i)/2 + "L ");
            }
            else
            {
                System.out.print(this.RightBand.Adjacencies.get(i)/2 + "R ");
            }
            //System.out.print(this.RightBand.Adjacencies.get(i));
        }
        System.out.println();
        System.out.println();
    }
}
```

```
package DishwasherFactory;

import java.util.ArrayList;
public class Warehouse
{
    int totalLoad;
    int CarrierCount;
    ArrayList<Carrier> Carriers = new ArrayList<Carrier>();

    public Warehouse(int totalLoad, int CarrierCount)
    {
        this.totalLoad = totalLoad;
        this.CarrierCount = CarrierCount;
        for(int i = 0; i < CarrierCount; i++)
        {
            CreateNewCarrier();
        }
    }

    public void CarryFromWarehouse(Carrier c, AssemblyLine.Band.Station
s)
    {
        this.totalLoad -= c.CarryToStation(s);
    }

    public void CreateNewCarrier()
    {
        this.Carriers.add(new Carrier());
    }

    public void PrintAllCarriers()
    {
        System.out.println("Printing Carriers");
        int totalLoadHere = 0;
        for(int i = 0; i < this.Carriers.size(); i++)
        {
            System.out.println("Carrier " + i + " carried: " +
this.Carriers.get(i).totalLoadCarried + " units of load." );
            totalLoadHere += this.Carriers.get(i).totalLoadCarried;
        }
        System.out.println("Total " + this.Carriers.size() + " carriers created.");
        System.out.println("Total of " + totalLoadHere + " weight carried.");
    }

    class Carrier
    {
        int totalLoadCarried;
```

```
        ArrayList<Integer> FromLine = new ArrayList<Integer>();
        ArrayList<Integer> LoadOfStation = new ArrayList<Integer>();
        Carrier()
        {
            totalLoadCarried = 0;
        }

        int CarryToStation(AssemblyLine.Band.Station s)
        {
            int loadCarried = s.getRequiredWeight();
            s.FillLoad(loadCarried);
            this.totalLoadCarried += loadCarried;
            return loadCarried;
        }
    }
}


package DishwasherFactory;

import java.util.ArrayList;
import javax.print.attribute.standard.Finishings;
public class Factory
{
    int totalWeightNeeded;
    int WeightPerCarrier;
    ArrayList<AssemblyLine> AssemblyLines = new
ArrayList<AssemblyLine>();
    Warehouse war1;
    ArrayList<AssemblyLine.Band.Station> excludedStations = new
ArrayList<AssemblyLine.Band.Station>();

    public Factory(int AssemblyLineCount, int[][] StationMatrix ,int[][]
AdjacencyMatrix, Warehouse war1)
    {
        totalWeightNeeded = 0;
        this.war1 = war1;
        // Factory will have a number of assembly lines
        for(int i = 0; i < AssemblyLineCount; i++)
        {
            // Assembly Lines and Bands created here
            AssemblyLines.add(new AssemblyLine(i + "", StationMatrix[i][0],
StationMatrix[i][1]));
        }

        // Adjacencies of assembly lines is set here (all lines are adjacent to
themselves)
```

```
        for(int i = 0; i < AssemblyLineCount * 2; i++)
        {
            for(int j = 0; j < AssemblyLineCount * 2; j++)
            {
                if(AdjacencyMatrix[i][j] == 1)
                {
                    if(i == j)
                    {
                        if(i%2 == 0)
                        {
                            AssemblyLines.get(i/2).LeftBand.Adjacencies.add(i);
                        }
                        else
                        {
                            AssemblyLines.get(i/2).RightBand.Adjacencies.add(i);
                        }
                    }
                    else
                    {
                        if(i%2 == 0)
                        {
                            AssemblyLines.get(i/2).LeftBand.Adjacencies.add(j);
                        }
                        else
                        {
                            AssemblyLines.get(i/2).RightBand.Adjacencies.add(j);
                        }
                    }
                }
            }
            AssemblyLines.get(i/2).LeftBand.CalculateAdjCount();
            AssemblyLines.get(i/2).RightBand.CalculateAdjCount();
        }
        this.FindExcludedStations();
    }

    public void FindExcludedStations()
    {
        ArrayList<AssemblyLine.Band> highAdj = new
ArrayList<AssemblyLine.Band>();
        int highestAdj = 0;

        // Find highest Adj number
        for(int i = 0; i < this.AssemblyLines.size() * 2; i++)
        {
            if(i % 2 == 0)
            {
```

```
            if(this.AssemblyLines.get(i/2).LeftBand.AdjCount > highestAdj)
            {
                highestAdj = this.AssemblyLines.get(i/2).LeftBand.AdjCount;
            }
        }
        else
        {
            if(this.AssemblyLines.get(i/2).RightBand.AdjCount > highestAdj)
            {
                highestAdj = this.AssemblyLines.get(i/2).RightBand.AdjCount;
            }
        }
    }


    // Add them to the list
    for(int i = 0; i < this.AssemblyLines.size() * 2; i++)
    {
        if(i % 2 == 0)
        {
            if(this.AssemblyLines.get(i/2).LeftBand.AdjCount == highestAdj)
            {
                highAdj.add(this.AssemblyLines.get(i/2).LeftBand);
            }
        }
        else
        {
            if(this.AssemblyLines.get(i/2).RightBand.AdjCount == highestAdj)
            {
                highAdj.add(this.AssemblyLines.get(i/2).RightBand);
            }
        }
    }

    // Find the excluded Stations now
    int smallestWeightStation = Integer.MAX_VALUE;
    int tj = -1,tk = -1;
    for(int i = 0; i < this.war1.CarrierCount + 2; i++)
    {
        for(int j = 0; j < highAdj.size(); j++)
        {
            for(int k = 0; k < highAdj.get(j).Stations.size(); k++)
            {
                if(highAdj.get(j).Stations.get(k).excluded == false &&
highAdj.get(j).Stations.get(k).requiredWeight < smallestWeightStation)
                {
```

```
                    smallestWeightStation =
highAdj.get(j).Stations.get(k).requiredWeight;
                    tj = j;
                    tk = k;
                }
            }
        }
        highAdj.get(tj).Stations.get(tk).excluded = true;
        smallestWeightStation = Integer.MAX_VALUE;
        excludedStations.add(highAdj.get(tj).Stations.get(tk));
        tj = -1;
        tk = -1;
    }
  }

  public void CalculateWeightPerCarrier()
  {
    CalculateTotalWeight();
    int excludedWeight = 0;
    for(int i = 0; i < this.excludedStations.size(); i++)
    {
      excludedWeight += this.excludedStations.get(i).requiredWeight;
    }
    WeightPerCarrier = (totalWeightNeeded - excludedWeight) /
war1.CarrierCount;
  }

  public void CalculateTotalWeight()
  {
    for(int i = 0; i < this.AssemblyLines.size(); i++)
    {
      for(int j = 0; j < this.AssemblyLines.get(i).LeftBand.Stations.size();
j++)
      {
        totalWeightNeeded +=
this.AssemblyLines.get(i).LeftBand.Stations.get(j).requiredWeight;
      }

      for(int j = 0; j < this.AssemblyLines.get(i).RightBand.Stations.size();
j++)
      {
        totalWeightNeeded +=
this.AssemblyLines.get(i).LeftBand.Stations.get(j).requiredWeight;
      }
    }
  }
```

```
    public void FinishAllCarriers()
    {
        CalculateWeightPerCarrier();
        System.out.println("Weight per Carrier: " + WeightPerCarrier /*+ " Total
weight to be carried: " + totalWeightNeeded*/);
        for(int i = 0; i < this.war1.Carriers.size(); i++)
        {
            while(this.war1.Carriers.get(i).totalLoadCarried < WeightPerCarrier)
            {
                AssemblyLine.Band.Station tempSt = FindNonProcessedStation();
                if(tempSt == null)
                {
                    break;
                }
                else
                {

this.war1.Carriers.get(i).LoadOfStation.add(tempSt.requiredWeight);
                    this.war1.CarryFromWarehouse(this.war1.Carriers.get(i),
tempSt);
                }
            }
        }
    }

    public void BalanceCarriers()
    {
        int leastCarriedCarrier = 0;
        for(int i = 0; i < excludedStations.size(); i++)
        {
            for(int j = 0; j < this.war1.Carriers.size(); j++)
            {
                if(this.war1.Carriers.get(j).totalLoadCarried <
this.war1.Carriers.get(leastCarriedCarrier).totalLoadCarried)
                {
                    leastCarriedCarrier = j;
                }
            }

this.war1.CarryFromWarehouse(this.war1.Carriers.get(leastCarriedCarrier),
excludedStations.get(i));
            //excludedStations.remove(i);
            excludedStations.get(i).requiredWeight = 0;
        }
    }

    void FineTuneCarriers(int times)
```

```
    {
        for(int i = 0; i < times; i++)
        {
            FindBestCarrierStationToSwap();
        }
    }

    void FindBestCarrierStationToSwap()
    {
        Warehouse.Carrier givingCarrier = FindBiggestCarrier();
        Warehouse.Carrier takingCarrier = FindSmallestCarrier();

        //Find max load of biggest station
        int maxStation = 0;
        int indexMax = -1;
        for(int i = 0; i < givingCarrier.LoadOfStation.size(); i++)
        {
            if(givingCarrier.LoadOfStation.get(i) > maxStation)
            {
                maxStation = givingCarrier.LoadOfStation.get(i);
                indexMax = i;
            }
        }

        //Find min load of smallest station
        int minStation = Integer.MAX_VALUE;
        int indexMin = -1;
        for(int i = 0; i < takingCarrier.LoadOfStation.size(); i++)
        {
            if(takingCarrier.LoadOfStation.get(i) < minStation)
            {
                minStation = takingCarrier.LoadOfStation.get(i);
                indexMin = i;
            }
        }

        if(givingCarrier.totalLoadCarried > WeightPerCarrier &&
takingCarrier.totalLoadCarried < WeightPerCarrier)
        {
            givingCarrier.totalLoadCarried = givingCarrier.totalLoadCarried -
givingCarrier.LoadOfStation.get(indexMax) +
takingCarrier.LoadOfStation.get(indexMin);
            takingCarrier.totalLoadCarried = takingCarrier.totalLoadCarried +
givingCarrier.LoadOfStation.get(indexMax) -
takingCarrier.LoadOfStation.get(indexMin);

            int temp = givingCarrier.LoadOfStation.get(indexMax);
```

```
            givingCarrier.LoadOfStation.remove(indexMax);

givingCarrier.LoadOfStation.add(takingCarrier.LoadOfStation.get(indexMin));
            takingCarrier.LoadOfStation.remove(indexMin);
            takingCarrier.LoadOfStation.add(temp);
        }
    }

    private Warehouse.Carrier FindBiggestCarrier()
    {
        Warehouse.Carrier temp = null;
        int maxLoad = 0;
        for(int i = 0; i < this.war1.Carriers.size(); i++)
        {
            if(maxLoad < this.war1.Carriers.get(i).totalLoadCarried)
            {
                maxLoad = this.war1.Carriers.get(i).totalLoadCarried;
                temp = this.war1.Carriers.get(i);
            }
        }
        return temp;
    }

    private Warehouse.Carrier FindSmallestCarrier()
    {
        Warehouse.Carrier temp = null;
        int minLoad = Integer.MAX_VALUE;
        for(int i = 0; i < this.war1.Carriers.size(); i++)
        {
            if(minLoad > this.war1.Carriers.get(i).totalLoadCarried)
            {
                minLoad = this.war1.Carriers.get(i).totalLoadCarried;
                temp = this.war1.Carriers.get(i);
            }
        }
        return temp;
    }

    public void PrintExcludedStations()
    {
        int total = 0;
        System.out.println("Printing Excluded Stations.");
        for(int i = 0; i < this.excludedStations.size(); i++)
        {
            System.out.println("Ex Station: " +
this.excludedStations.get(i).requiredWeight);
            total += this.excludedStations.get(i).requiredWeight;
```

```
        }
        System.out.println("Total: " + total);
    }

    public AssemblyLine.Band.Station FindNonProcessedStation()
    {
        for(int i = 0; i < this.AssemblyLines.size() * 2; i++)
        {
            if(i % 2 == 0)
            {
                for(int j = 0; j < this.AssemblyLines.get(i/2).LeftBand.Stations.size();
j++)
                {

if(this.AssemblyLines.get(i/2).LeftBand.Stations.get(j).requiredWeight != 0 &&
this.AssemblyLines.get(i/2).LeftBand.Stations.get(j).excluded == false)
                    {
                        return this.AssemblyLines.get(i/2).LeftBand.Stations.get(j);
                    }
                }
            }
            else
            {
                for(int j = 0; j <
this.AssemblyLines.get(i/2).RightBand.Stations.size(); j++)
                {

if(this.AssemblyLines.get(i/2).RightBand.Stations.get(j).requiredWeight != 0
&& this.AssemblyLines.get(i/2).RightBand.Stations.get(j).excluded == false)
                    {
                        return this.AssemblyLines.get(i/2).RightBand.Stations.get(j);
                    }
                }
            }
        }
        return null;
    }

    public void FinishLine(Warehouse w, int CarrierNo, int
UnfinishedBandIndex)
    {
        if(UnfinishedBandIndex % 2 == 0)
        {
            for(int j = 0; j <
this.AssemblyLines.get(UnfinishedBandIndex/2).LeftBand.Adjacencies.size();
j++)
            {
```

```
if(this.AssemblyLines.get(UnfinishedBandIndex/2).LeftBand.Adjacencies.get(j
)%2 == 0)
            {
               for(int i = 0; i <
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Left
Band.Adjacencies.get(j)/2).LeftBand.Stations.size(); i++)
               {
                  w.CarryFromWarehouse(w.Carriers.get(CarrierNo),
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Left
Band.Adjacencies.get(j)/2).LeftBand.Stations.get(i));
               }
            }
            else
            {
               for(int i = 0; i <
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Left
Band.Adjacencies.get(j)/2).RightBand.Stations.size(); i++)
               {
                  w.CarryFromWarehouse(w.Carriers.get(CarrierNo),
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Left
Band.Adjacencies.get(j)/2).RightBand.Stations.get(i));
               }
            }
         }
      }
      else
      {
         for(int j = 0; j <
this.AssemblyLines.get(UnfinishedBandIndex/2).RightBand.Adjacencies.size(
); j++)
         {


if(this.AssemblyLines.get(UnfinishedBandIndex/2).RightBand.Adjacencies.ge
t(j)%2 == 0)
            {
               for(int i = 0; i <
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Righ
tBand.Adjacencies.get(j)/2).LeftBand.Stations.size(); i++)
               {
                  w.CarryFromWarehouse(w.Carriers.get(CarrierNo),
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Righ
tBand.Adjacencies.get(j)/2).LeftBand.Stations.get(i));
               }
            }
```

```
            else
            {
                for(int i = 0; i <
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Righ
tBand.Adjacencies.get(j)/2).RightBand.Stations.size(); i++)
                {
                    w.CarryFromWarehouse(w.Carriers.get(CarrierNo),
this.AssemblyLines.get(this.AssemblyLines.get(UnfinishedBandIndex/2).Righ
tBand.Adjacencies.get(j)/2).RightBand.Stations.get(i));
                }
            }
        }
    }
    public int FindUnfinishedLineIndex()
    {
        for(int i = 0; i < this.AssemblyLines.size()*2; i++)
        {
            if(i%2 == 0)
            {

if(this.AssemblyLines.get(i/2).LeftBand.Stations.get(0).requiredWeight > 0)
                {
                    return i;
                }
            }
            else
            {

if(this.AssemblyLines.get(i/2).RightBand.Stations.get(0).requiredWeight > 0)
                {
                    return i;
                }
            }
        }
        return -1;
    }

    // Finishes all unfinished lines on the factory
    public void FinishAllLines(Warehouse w)
    {
        while(this.FindUnfinishedLineIndex() != -1)
        {
            w.CreateNewCarrier();
            this.FinishLine(w, w.Carriers.size()-1 ,
this.FindUnfinishedLineIndex());
```

```
    }
}

// Prints the load on each station of each line
public void PrintStationLoads()
{
    int tot = 0;
    for(int i = 0; i < this.AssemblyLines.size(); i++)
    {
        this.AssemblyLines.get(i).LeftBand.PrintStationLoads();
        this.AssemblyLines.get(i).RightBand.PrintStationLoads();
        tot += this.AssemblyLines.get(i).LeftBand.total;
        tot += this.AssemblyLines.get(i).RightBand.total;
    }
    System.out.println("Total load: " + tot);
}

// Prints all lines in the factory and their adjacencies
public void PrintFactory()
{
    for(int i = 0; i < this.AssemblyLines.size(); i++)
    {
        this.AssemblyLines.get(i).printAssemblyLine();
    }
}

public static void main(String[] args)
{
    //15-45 1-20
    System.out.println("Factory Starting");
    int [][] adjMatrix = new int[][]
    {
        {1,1,0,0,1,1,0,0,0,0},
        {1,1,0,0,1,1,1,0,0,0},
        {0,0,1,1,0,0,0,0,0,0},
        {0,0,1,1,0,0,0,0,0,0},
        {1,1,0,0,1,1,0,0,1,0},
        {1,1,0,0,1,1,0,0,1,0},
        {0,1,0,0,0,0,1,1,0,0},
        {0,0,0,0,0,0,1,1,0,0},
        {0,0,0,0,1,1,0,0,1,1},
        {0,0,0,0,0,0,0,0,1,1},
    };

    int [][] stationMatrix = new int[][]
    {
        {8,8},
```

```
            {7,7},
            {8,8},
            {9,9},
        };

        // Create warehouse
        Warehouse war1 = new Warehouse(40000, 5);

        // Create the factory and print it
        Factory fac1 = new Factory(4, stationMatrix, adjMatrix, war1);
        fac1.PrintFactory();

        // Prints the load on each station of each line
        fac1.PrintStationLoads();
        System.out.println();

        // Print the job done by all the carriers
        //fac1.war1.PrintAllCarriers();

        fac1.FinishAllCarriers();
        System.out.println("Finished first iteration");
        fac1.war1.PrintAllCarriers();
        System.out.println();

        fac1.BalanceCarriers();
        System.out.println("Finished balancing");
        fac1.war1.PrintAllCarriers();
        System.out.println();

        fac1.FineTuneCarriers(50);
        System.out.println("Finished fine tuning");
        fac1.war1.PrintAllCarriers();

    }
}
```

EK 5

```
int n=...;/* t turlarının sayısı*/
int h=...;/*bir malzemecinin yapabileceği maksimum tur sayısı*/
range a=1..n;/*t zaman dilimlerinin range*/
range b=1..h;/*maksimum tur range*/
range az=0..n;/
{int} P=...;/*malzeme kümesi*/
{int} M=...;/*malzemeci kümesi*/
{int} I=...;/*istasyon kümesi*/

float D1[P][I]=...;
float D2[P][I]=...;
float D3[P][I]=...;
float D4[P][I]=...;
float D5[P][I]=...;
float D6[P][I]=...;
float D7[P][I]=...;

float F[M]=...;
float V[P]=...;
float S[P][I]=...;
int K=...;
string sepetTipi[P]=...;
dvar int tasinan[M][a][1..h];
dvar int+ x[M][P][I][a];
dvar int+ y[M][P][I][a][1..h];
dvar boolean f[M][P][a][1..h];
dvar boolean beta[M][a][b];
dvar int teta[M][a];
dvar float+ z[P][I][az];


dvar float+ q[P][I][a];
dvar float+ w[P][I][a];

minimize
sum(i in M,t in a) teta[i][t];;
  subject to{

  /*1*/forall(j in P, k in I)
    {
    z[j][k][0] +sum(i in M)(x[i][j][k][1]) -D1[j][k]==z[j][k][1];
    z[j][k][1]>=D2[j][k];

    if(D2[j][k]+D3[j][k]+D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]==0)
    {
```

```
 z[j][k][1]==q[j][k][1];
}
else  if(D2[j][k]+D3[j][k]+D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]!=0)
{
q[j][k][1]==0;
}

}

/*2*/forall(j in P, k in I)
{
z[j][k][1] +sum(i in M)(x[i][j][k][2]) -D2[j][k]==z[j][k][2];
z[j][k][2]>=D3[j][k];

if(D3[j][k]+D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]==0)
{
 z[j][k][2]==q[j][k][2];
}
else  if(D3[j][k]+D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]!=0)
{
q[j][k][2]==0;
}
}

/*3*/forall(j in P, k in I)
{
z[j][k][2] +sum(i in M)(x[i][j][k][3]) -D3[j][k]==z[j][k][3];
z[j][k][3]>=D4[j][k];

if(D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]==0)
{
 z[j][k][3]==q[j][k][3];
}
else  if(D4[j][k]+D5[j][k]+D6[j][k]+D7[j][k]!=0)
{
q[j][k][3]==0;
}
}

/*4*/forall(j in P, k in I)
{
z[j][k][3] +sum(i in M)(x[i][j][k][4]) -D4[j][k]==z[j][k][4];
z[j][k][4]>=D5[j][k];

if(D5[j][k]+D6[j][k]+D7[j][k]==0)
{
 z[j][k][4]==q[j][k][4];
```

```
    }
    else  if(D5[j][k]+D6[j][k]+D7[j][k]!=0)
    {

    q[j][k][4]==0;
    }
    }

/*5*/forall(j in P, k in I)
    {
    z[j][k][4] +sum(i in M)(x[i][j][k][5]) -D5[j][k]==z[j][k][5];
    z[j][k][5]>=D6[j][k];

    if(D6[j][k]+D7[j][k]==0)
    {
     z[j][k][5]==q[j][k][5];
    }
    else  if(D6[j][k]+D7[j][k]!=0)
    {

    q[j][k][5]==0;
    }
    }

/*6*/forall(j in P, k in I)
    {
    z[j][k][5] +sum(i in M)(x[i][j][k][6]) -D6[j][k]==z[j][k][6];
    z[j][k][6]>=D7[j][k];

    if(D7[j][k]==0)
    {
     z[j][k][6]==q[j][k][6];
    }
    else  if(D7[j][k]!=0)
    {
    q[j][k][6]==0;
    }
    }

/*7*/forall(j in P, k in I)
    {
    z[j][k][6] +sum(i in M)(x[i][j][k][7]) -D7[j][k]==z[j][k][7];
    q[j][k][7]==0;
    }
 /*8*/forall(j in P,k in I)
    {
    z[j][k][0]==S[j][k];
```

```
   }
 /*9 */forall(j in P, k in I, t in a)
   {
   w[j][k][t]==z[j][k][t]-q[j][k][t];

   }

   /*10*/forall(k in I, t in a)
   {
  sum(j in P) w[j][k][t]<=G;

   }


/*1*/ forall(i in M, t in a, m in b)
   {
   sum(j in P, k in I)(y[i][j][k][t][m]*V[j])<=F[i];
   }

/*3*/forall(i in M, j in P, t in a,m in b)
   {
   sum(k in I)(y[i][j][k][t][m])<=f[i][j][t][m]*K;
   }


/*4*/forall(i in M, j in P,  t in a,m in b)
   {
   f[i][j][t][m]<=sum(k in I)(y[i][j][k][t][m]*K);
   }


/*5*/forall(i in M, j in P, t in a,m in b, d in P:sepetTipi[j]!=sepetTipi[d])
   {
   f[i][j][t][m]+f[i][d][t][m]<=1;
   }

/*6*/forall(i in M, j in P, t in a,m in b)
   {
   beta[i][t][m]*K>=sum(j in P)f[i][j][t][m];
   }

/*7*/forall(i in M,  t in a)
   {
   teta[i][t]==sum(m in b)beta[i][t][m];
   }
```

```
/*8*/forall(i in M, j in P, k in I, t in a)
    {
    sum(m in b)(y[i][j][k][t][m])==x[i][j][k][t];
    }
/*9*/forall(i in M, t in a,m in b)
    {
    sum(j in P,k in I)(y[i][j][k][t][m])==tasinan[i][t][m];
    }

}

  execute DISPLAY {

  for(var i in thisOplModel.M) {
             for(var j in thisOplModel.P) {
             for(var k in thisOplModel.I) {
               for(var t in thisOplModel.a) {
               for(var m in thisOplModel.b)
               {


              if(y[i][j][k][t][m]>0)
              {



writeln("Malzemecinin,",j,",malzemesinden,",k,",istasyonuna,",t,",zamandilimi,
",m,",turundataşıdığımiktar,",y[i][j][k][t][m]);

          }            }}}}    }       }
```

Veri

```
SheetConnection sheet("deneme.xlsx");

n         from SheetRead(sheet,"EXTRA!A2");
h         from SheetRead(sheet,"EXTRA!B2");

K         from SheetRead(sheet,"EXTRA!D2");

M         from SheetRead(sheet,"EXTRA!E2:E2");
F         from SheetRead(sheet,"EXTRA!G2:G2");

I         from SheetRead(sheet,"EXTRA!F2:F53");

P         from SheetRead(sheet,"MALZEME!B2:B16");
V         from SheetRead(sheet,"MALZEME!G2:G16");
S         from SheetRead(sheet,"S!C2:BB16");
sepetTipi from SheetRead(sheet,"EXTRA!E2:E16");

D1        from SheetRead(sheet,"d1!C2:BB16");
D2        from SheetRead(sheet,"d2!C2:BB16");
D3        from SheetRead(sheet,"d3!C2:BB16");
D4        from SheetRead(sheet,"d4!C2:BB16");
D5        from SheetRead(sheet,"d5!C2:BB16");
D6        from SheetRead(sheet,"d6!C2:BB16");
D7        from SheetRead(sheet,"d7!C2:BB16");
```

EK 6

```
package entity;

public final class OrtamDegiskenleri {

//	public static final int SAAT_DILIMINDEKI_TUR_SAYISI = 6;
//	public static final int MALZEMECI_SAYISI=3;
//	public static final int MALZEMECININ_TASIYABILDIGI_HACIM =6;
//
//	public static final int ZAMAN_DILIMI_SAYISI = 7;
//	public static final int ISTASYON_SAYISI = 52;
//
//	public static final int HATBASI_MAX_MIKTAR = 4;
//
//	public static final String FILE_PATH =
"/home/melis/Desktop/melis_bitirme/bitirme.xlsx";


	public static final int SAAT_DILIMINDEKI_TUR_SAYISI = 4;
	public static final int MALZEMECI_SAYISI=3;
	public static final int MALZEMECININ_TASIYABILDIGI_HACIM =6;

	public static final int ZAMAN_DILIMI_SAYISI = 7;
	public static final int ISTASYON_SAYISI = 52;

	public static final int HATBASI_MAX_MIKTAR = 5;

	public static final String FILE_PATH =
"C:/Users/samsung/Desktop/denemeler son/denemeler/denemeilk.xlsx";

	public static final int DENEME_SAYISI = 150;
}
```

```
package main;

import java.util.List;
import java.util.Random;

import entity.Malzeme;
import entity.OrtamDegiskenleri;
import entity.Talep;
import entity.Tur;
import excel.DataManager;
import solver.Solver2;

public class MelisBitirmeMain {

    public static final int RASGELE_URETILECEK_TALEP_SAYISI = 50;
    public static final String RANDOM_SEED = "melis";
    public static final boolean RASTGELE_URET = false;
    public static final boolean RASTGELE_SIRALA = false;

    public static final int RASTGELE_MAX_TALEP_MIKTARI = 2;
        public static final int MALZEME_SAYISI = 15;

    public static void main(String[] args) {
        System.out.println("started");

        DataManager dm;

        //Rastgele talep üret
        if (RASTGELE_URET) {
            dm = new DataManager(OrtamDegiskenleri.FILE_PATH, false);

            List<Talep> randomTaleps = dm.getTaleps();

            Random random = new Random(RANDOM_SEED.hashCode());
            for (int i = 0; i < RASGELE_URETILECEK_TALEP_SAYISI; i++) {
                Talep talep = new Talep();

talep.setIstasyonNumarasi(random.nextInt(OrtamDegiskenleri.ISTASYON_S
AYISI));
                List<Malzeme> malzemes = dm.getMalzemes();

talep.setMalzeme(malzemes.get(random.nextInt(malzemes.size())));
                talep.setMiktari(random.nextDouble() *
RASTGELE_MAX_TALEP_MIKTARI);

talep.setZamanDilimi(random.nextInt(OrtamDegiskenleri.ZAMAN_DILIMI_SA
YISI));
```

```
            randomTaleps.add(talep);
        }

        dm.setTaleps(randomTaleps);
    }

//      Talepleri dosyadan oku
    else {
        dm = new DataManager(OrtamDegiskenleri.FILE_PATH, true);
    }

    List<Talep> randomTaleps = dm.getTaleps();
//      for (Talep talep : randomTaleps) {
//          System.out.println(talep.getMalzeme().getAdi());
//          System.out.println(talep.getMalzeme().getId());
//          System.out.println("istasyon numarasÄ±: " +
talep.getIstasyonNumarasi());
//          System.out.println("miktarÄ±: " + talep.getMiktari());
//          System.out.println("zaman dilimi: " + talep.getZamanDilimi());
//          System.out.println("----------------------");
//      }

    Solver2 solver2 = new Solver2(dm);
    List<Tur> ilkCozumTurlar = solver2.getTurs();


    printTurs(ilkCozumTurlar);

System.out.println("^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^");
    List<Tur> solve = solver2.solve2(ilkCozumTurlar);

    printTurs(solve);

  }

  private static void printTurs(List<Tur> solve){
    int size = 0;
    for (Tur tur : solve) {
        if (tur.getHacim() == 0)
            continue;
        System.out.println("#########################");
        System.out.println(tur.getZamanDilimi() + ". zaman dilimi");
        List<Talep> taleps = tur.getTaleps();
        for (Talep talep : taleps) {
            System.out.println(talep.getMalzeme().getId() + "  " +
talep.getMiktari() + " " + talep.getIstasyonNumarasi());
```

```
            }
            System.out.println(tur.getHacim() + " hacim");
            System.out.println("sepet tipi: " + tur.getTurSepetTipi());
            size++;
        }

        System.out.println("toplam tur sayı±sı±: " + size);
    }
}

package solver;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import entity.Istasyon;
import entity.OrtamDegiskenleri;
import entity.Talep;
import entity.Tur;
import excel.DataManager;
import main.MelisBitirmeMain;

public class Solver2 {

        private DataManager dm;

        public Solver2(DataManager dm) {
                this.dm = dm;
        }

        public List<Tur> solve3(List<Tur> turs) {

                return turs;
        }

        public List<Tur> solve(List<Tur> turs) {
                for (Tur tur : turs) {
                        Collections.sort(tur.getTaleps(), new
Comparator<Talep>() {

                                @Override
                                public int compare(Talep o1, Talep o2) {
```

```
                                        return (int) (o1.getMalzeme().getHacim() -
o2.getMalzeme().getHacim());
                                }
                        });
                }

                if (MelisBitirmeMain.RASTGELE_SIRALA) {
                        Collections.shuffle(turs);
                } else {

                        Collections.sort(turs, new Comparator<Tur>() {

                                @Override
                                public int compare(Tur o1, Tur o2) {
                                        return o1.getHacim() - o2.getHacim();
                                }
                        });
                }

                boolean ilerleme = true;

                int denemeSayisi = 0;
                while (ilerleme && denemeSayisi <
OrtamDegiskenleri.DENEME_SAYISI) {
                        ilerleme = false;
                        denemeSayisi++;
                        for (Tur tur : turs) {
                                for (Tur tur2 : turs) {

                                        if (tur2.getZamanDilimi() <=
tur.getZamanDilimi())
                                                continue;
                                        if (tur2.getTaleps().size() == 0 ||
tur.getTaleps().size() == 0)
                                                continue;
                                        if
(!tur2.getTurSepetTipi().equals(tur.getTurSepetTipi()))
                                                continue;

                                        Talep talep = tur2.getTaleps().get(0);

                                        try {
                                                tur.addTalep(talep);
                                        } catch (Exception e) {
                                                continue;
                                        }
```

```
                                        tur2.removeTalep(talep);
                                        boolean solveForTurs = solveForTurs(turs);

                                        if (solveForTurs) {
                                                ilerleme = true;
                                        } else {
                                                try {
                                                        tur2.addTalep(talep);
                                                } catch (Exception e) {
                                                        assert (false);
                                                }
                                                tur.removeTalep(talep);
                                        }

                                }
                        }
                }

                return turs;
        }

        public List<Tur> solve2(List<Tur> turs) {
                boolean ilerleme = true;
                while (ilerleme) {
                        ilerleme = false;
                        for (Tur tur : turs) {
                                for (Tur tur2 : turs) {
                                        if (tur2.getZamanDilimi() <=
tur.getZamanDilimi())
                                                continue;
                                        if
(!tur2.getTurSepetTipi().equals(tur.getTurSepetTipi()))
                                                continue;
                                        if (tur2.getTaleps().size() == 0)
                                                continue;
                                        Talep talep = tur2.getTaleps().get(0);

                                        try {
                                                tur.addTalep(talep);
                                        } catch (Exception e) {
                                                continue;
                                        }

                                        tur2.removeTalep(talep);
                                        boolean solveForTurs = solveForTurs(turs);

                                        if (solveForTurs) {
```

```
                                        ilerleme = true;
                                    } else {
                                        try {
                                            tur2.addTalep(talep);
                                        } catch (Exception e) {
                                            assert (false);
                                        }
                                        tur.removeTalep(talep);
                                    }

                                }
                            }
                        }
                        return turs;
                }

                public boolean solveForTurs(List<Tur> turs) {
                        List<Istasyon> istasyons = dm.getIstasyons();
                        for (int i = 0; i < OrtamDegiskenleri.ZAMAN_DILIMI_SAYISI;
i++) {
                            List<Tur> tursByZamanDilimi =
SolveServices.getTursByZamanDilimi(turs, i);
                            List<Talep> talepsByZamanDilimi =
SolveServices.getTalepsByZamanDilimi(dm.getTaleps(),
                                    i);
                            for (Tur tur : tursByZamanDilimi) {
                                List<Talep> taleps = tur.getTaleps();
                                for (Talep talep : taleps) {
                                    int istasyonNumarasi =
talep.getIstasyonNumarasi();
                                    Istasyon istasyon =
istasyons.get(istasyonNumarasi);
                                    try {

                istasyon.hatbasiMalzemeEkle(talep.getMalzeme(), talep.getMiktari());
                                    } catch (Exception e) {
                                        return false;
                                    }
                                }
                            }


                        }

                        return checkTursForMalzemeci(turs);
//                      return true;
                }
```

```
	private static boolean checkTursForMalzemeci(List<Tur> turs) {
		Map<Integer, Integer> turSayilari = new HashMap<>();
		for (Tur tur : turs) {
			if (!turSayilari.containsKey(tur.getZamanDilimi())) {
				turSayilari.put(tur.getZamanDilimi(), 0);
			}
			turSayilari.put(tur.getZamanDilimi(),
turSayilari.get(tur.getZamanDilimi()) + 1);

			if (turSayilari.get(tur.getZamanDilimi()) >
OrtamDegiskenleri.SAAT_DILIMINDEKI_TUR_SAYISI) {
				return false;
			}
		}

		return true;
	}

	public List<Tur> getTurs() {
		List<Tur> turs = new ArrayList<>();
		List<Talep> taleps = dm.getTaleps();
		List<Istasyon> istasyons = dm.getIstasyons();
		for (int i = 0; i < OrtamDegiskenleri.ZAMAN_DILIMI_SAYISI;
i++) {

			List<Talep> talepsByZamanDilimi =
SolveServices.getTalepsByZamanDilimi(taleps, i);
			List<List<Talep>> kategorilendirTalepsBySepetTipi =
SolveServices

	.kategorilendirTalepsBySepetTipi(talepsByZamanDilimi);

			for (List<Talep> list : kategorilendirTalepsBySepetTipi) {
				Tur tur = new Tur(i);
				for (int j = 0; j < list.size(); j++) {
					Talep talep = list.get(j);
					Istasyon istasyon =
istasyons.get(talep.getIstasyonNumarasi());
					Double hatBasiMalzemeMiktari =
istasyon.getMalzemeMiktari()

	.get(talep.getMalzeme().getId());
					if (hatBasiMalzemeMiktari != null
							&& hatBasiMalzemeMiktari >=
talep.getMiktari()) {
						try {
```

```
istasyon.hatbasiMalzemeCikar(talep.getMalzeme(), talep.getMiktari());
                                        } catch (Exception e) {
                                                e.printStackTrace();
                                        }
                                        continue;
                                }
                                try {
                                        tur.addTalep(talep);
                                } catch (Exception e) {
                                        turs.add(tur);
                                        tur = new Tur(i);
                                        j--;
                                }
                        }
                        if (tur.getHacim() > 0)
                                turs.add(tur);
                }
        }

        return turs;
}
```

```
package excel;

import entity.Istasyon;
import entity.Malzeme;
import entity.OrtamDegiskenleri;
import entity.Talep;
import main.MelisBitirmeMain;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.*;

public class DataManager {

    private String filePath;
    private List<Malzeme> malzemes;
    private List<Talep> taleps;
    private Map<Integer, Malzeme> malzemeMap;
    private List<Istasyon> istasyons;
    private List<Istasyon> istasyonsIlkDurum;
    private boolean initFromFile;

    public List<Malzeme> getMalzemes() {
        return malzemes;
    }

    public List<Talep> getTaleps() {
        return taleps;
    }

    public void setTaleps(List<Talep> taleps) {
        this.taleps = taleps;
    }

    public DataManager(String filePath, boolean initFromFile) {
        this.filePath = filePath;
        this.initFromFile = initFromFile;
        malzemes = new ArrayList<>();
        taleps = new ArrayList<>();
        malzemeMap = new HashMap<>();
```

```
        istasyons = new ArrayList<>();

        try {
            initMalzemes();
            mapMalzemes();
            if (initFromFile)
                initTaleps();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void initIstasyons() throws IOException {
        istasyons.clear();
        if (initFromFile) {
            if (istasyonsIlkDurum == null) {
                istasyonsIlkDurum = new ArrayList<>();
                XSSFWorkbook workbook = new XSSFWorkbook(new
FileInputStream(filePath));
                XSSFSheet sheet = workbook.getSheet("S");
                for (int j = 2; j < OrtamDegiskenleri.ISTASYON_SAYISI + 2; j++) {
                    Istasyon istasyon = new Istasyon();

                    for (int i = 1; i <= malzemes.size(); i++) {
                        XSSFRow row = sheet.getRow(i);
                        int malzemeId = (int) row.getCell(1).getNumericCellValue();
                        Malzeme malzemeById = getMalzemeById(malzemeId);
                        double hatBasiSayisi = row.getCell(j).getNumericCellValue();
                        if (hatBasiSayisi != 0)
                            try {
                                istasyon.hatbasiMalzemeEkle(malzemeById,
hatBasiSayisi);
                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                    }

                    istasyonsIlkDurum.add(istasyon);
                }
            }

            for (Istasyon istasyon : istasyonsIlkDurum) {
                try {
                    istasyons.add((Istasyon) istasyon.clone());
                } catch (CloneNotSupportedException e) {
                    e.printStackTrace();
                }
```

```
            }
        } else {
            for (int i = 0; i < OrtamDegiskenleri.ISTASYON_SAYISI; i++) {
                istasyons.add(new Istasyon());
            }
        }

    }

    private void mapMalzemes() {
        for (Malzeme malzeme : malzemes) {
            malzemeMap.put(malzeme.getId(), malzeme);
        }

    }

    private void initTaleps() throws IOException {
        XSSFWorkbook workbook = new XSSFWorkbook(new
FileInputStream(filePath));
        for (int i = 1; i <= OrtamDegiskenleri.ZAMAN_DILIMI_SAYISI; i++) {
            XSSFSheet sheet = workbook.getSheet("d" + i);

            for (int k = 1; k <= malzemes.size(); k++) {
                Row row = sheet.getRow(k);
                Cell cell = row.getCell(1);
                if (cell == null)
                    break;
                int malzemeId = (int) cell.getNumericCellValue();
                Malzeme malzeme = getMalzemeById(malzemeId);
                for (int j = 2; j < OrtamDegiskenleri.ISTASYON_SAYISI + 2; j++) {
                    double talepMiktari = row.getCell(j).getNumericCellValue();
                    if (talepMiktari != 0) {
                        Talep talep = new Talep();
                        talep.setMalzeme(malzeme);
                        talep.setMiktari(talepMiktari);
                        talep.setZamanDilimi(i);
                        talep.setIstasyonNumarasi(j - 2);
                        taleps.add(talep);
                    }
                }
            }
        }
    }

    private Malzeme getMalzemeById(int id) {
        Malzeme malzeme = malzemeMap.get(id);
        if (malzeme == null)
```

```
            System.err.println("Bu id'de malzeme bulunamadÄ±: " + id);
        return malzeme;

    }

    private void initMalzemes() throws FileNotFoundException, IOException {
        XSSFWorkbook workbook = new XSSFWorkbook(new
FileInputStream(filePath));
        XSSFSheet sheet = workbook.getSheet("MALZEME");
        Iterator<Row> rowIterator = sheet.iterator();
        rowIterator.next();

        for (int i = 0; i < MelisBitirmeMain.MALZEME_SAYISI; i++) {
            Malzeme malzeme = new Malzeme();
            Row row = (Row) rowIterator.next();
            Cell cell = row.getCell(1);
            if (cell == null)
                break;
            malzeme.setId((int) cell.getNumericCellValue());
            malzeme.setAdi(row.getCell(2).getStringCellValue());
            malzeme.setSepetTipi(row.getCell(4).getStringCellValue());
            malzeme.setHacim((int) row.getCell(6).getNumericCellValue());
            malzemes.add(malzeme);
        }
    }

    public List<Istasyon> getIstasyons() {
        try {
            initIstasyons();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return istasyons;
    }
}
```

ÖZGEÇMİŞ

Ad-Soyad: : Melis Özcan

Uyruğu: : T.C.

Doğum tarihi ve yeri: : 26.05.1991 Ankara

Telefon: : +90 532 165 8591

e-mail: : melisoydan@gmail.com

Öğrenim Durumu

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lisans	TOBB ÜTÜ/Endüstri Müh.	2013

İş Deneyimi

Yıl	Yer	Görev
Nisan 2014-Mayıs 2015	GATE Elektronik San. Ve Tic A.Ş.	Proje Mühendisi
Mayıs2015-(Devam)	TUSAŞ Türk Havacılık ve Uzay Sanayii	Tedarik Planlama
Ocak –Nisan 2011	FNSS Savunma Sis.	Stajyer Mühendis
Eylül– Aralık 2012	FNSS Savunma Sis.	Stajyer Mühendis
Nisan – Ağustos 2013	FNSS Savunma Sis.	Stajyer Mühendis

Yabancı Dil

İngilizce (Çok İyi)

Almanca (Başlangıç)

Yayınlar

- Özcan, M., Montaj Hattı Besleme İşlemlerinin Çizelgelenmesi İçin Bir Çözüm Yaklaşımı, Yöneylem Araştırması ve Endüstri Mühendisliği Ulusal Kongresi, Eylül 2015,ODTÜ, Ankara, Türkiye.