FARKLI LOKASYONLARDA BULUNAN İKİ FABRİKA ARASINDAKİ YARI MAMÜL VE MALZEME AKIŞININ OPTİMİZASYONU VE STANDARTLAŞTIRILMASI

Elif KURTULUŞ

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FARKLI LOKASYONLARDA BULUNAN İKİ FABRİKA ARASINDAKİ YARI MAMÜL VE MALZEME AKIŞININ OPTİMİZASYONU VE STANDARTLAŞTIRILMASI

Elif KURTULUŞ
0000-0003-1627-9299

Prof. Dr. Hüseyin Cenk ÖZMUTLU
0000-0003-2540-9657

(Danışman)

YÜKSEK LİSANS TEZİ

ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2019

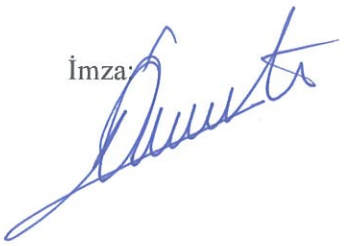
TEZ ONAYI

Elif KURTULUŞ tarafından hazırlanan "Farklı Lokasyonlarda Bulunan İki Fabrika Arasındaki Yarı Mamül ve Malzeme Akışının Optimizasyonu ve Standartlaştırılması " adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Prof. Dr. Cenk ÖZMUTLU
0000-0003-2540-9657

Başkan: Prof. Dr. Hüseyin Cenk ÖZMUTLU
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi
Endüstri Mühendisliği Anabilim Dalı
0000-0003-2540-9657

İmza:



Üye : Doç. Dr. Ali Yurdun ORBAK
Bursa Uludağ Üniversitesi,
Mühendislik Fakültesi
Endüstri Mühendisliği Anabilim Dalı
0000-0002-4921-4275

İmza:

Üye : Doç. Dr. Aytaç YILDIZ
Bursa Teknik Üniversitesi,
Mühendislik ve Doğa Bilimleri Fakültesi
Endüstri Mühendisliği Anabilim Dalı
0000-0002-0729-633X

İmza:

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
../../....

U.Ü. Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

11**/09/2019**

Elif KURTULUŞ

ÖZET

Yüksek Lisans Tezi

FARKLI LOKASYONLARDA BULUNAN İKİ FABRİKA ARASINDAKİ YARI MAMÜL VE MALZEME AKIŞININ OPTİMİZASYONU VE STANDARTLAŞTIRILMASI

Elif KURTULUŞ

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Hüseyin Cenk ÖZMUTLU

Üretim maliyetlerinin yönetimi sabit maliyetlerden çok değişken maliyetler üzerinden yapılmaktadır. Tedarik zincirlerinin uluslararası bir boyut kazandığı günümüzde lojistik maliyetlerin optimizasyonu küresel pazarda rekabetçi üstünlük açısından önemli bir fonksiyon haline gelmiştir. Artan çevre bilinci, kanunlar ve yüksek maliyetler de lojistik alanındaki optimizasyon çalışmalarına olan ilgiyi arttırmaktadır. Lojistik maliyetlerinin aşağı çekilebilmesi için rota optimizasyonu ve yük konsolidasyonunun yanı sıra konteynır/tır hacimlerinden en iyi şekilde faydalanmaya yönelik çalışmalar yapılmaktadır.

Konteynır/tır yükleme problemi, birçok farklı türü olan "kesme ve paketleme" problemleri kümesinde bulunan bir problemdir. Üç boyutlu bir problem olan konteynır/tır yükleme problemlerinde amaç; en, boy ve yüksekliği belirli n adet küçük kutunun, boyutları belirli bir konteynır/tır ın içine yerleştirilebilmesi suretiyle herhangi bir çakışma olmaksızın mevcut konteynır/tır hacminden maksimum derecede faydalanmaktır.

Bu çalışmada ele alınan problemde; Gönen'de bulunan üretim tesisi ve Bursa'da bulunan montaj fabrikası arasında sürekli yarı mamül ve malzeme akışı bulunmaktadır. Bu akış her gün üç vardiya süresince çalışan tırlar ile sağlanmaktadır. Tırlar farklı kutu içi miktarlara sahip yarı mamül ve malzemeleri (kılıf ve kesik kumaş) Gönen üretim tesisinden Bursa montaj fabrikasına taşımaktadır. Bu taşıma optimizasyonu problemi için; amaç fonksiyonu minimum tır sayısı ile akışı sağlamak olan ve tamsayılı karar değişkenlerinin kullanıldığı bir matematiksel model oluşturulmuştur. Ayrıca karınca kolonisi optimizasyonu yaklaşımını temel alan bir algoritma ile aynı problem çözülmüş ve elde edilen sonuçlar karşılaştırılmıştır.

Anahtar Kelimeler: Konteynır/tır yükleme, tamsayılı programlama, karınca kolonisi, optimizasyon

2019, vii + 44 sayfa.

ABSTRACT

MSc Thesis

OPTIMIZATION AND STANDARDIZATION OF RAW MATERIAL & MATERIAL FLOW BETWEEN TWO PLANTS IN DIFFERENT LOCATIONS

Elif KURTULUŞ

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Industrial Engineering

Supervisor: Prof. Dr. Hüseyin Cenk ÖZMUTLU

Production cost management is based on variable costs more than fixed costs. With the internationalization of supply chains, logistic cost optimization becomes a very important function for having competitive advantage in global market. Logistic cost optimization studies are also arousing interest because of the environmental awareness, statutes and high costs. Researchers are working on utilizing container/truck volume better in addition to road optimization and load consolidation for reducing these costs.

Container/truck loading problem is a kind of cutting and packaging problem which has various types. It is a three dimensional problem and its objective function is to utilize container/truck volume efficiently by loading small boxes that have specific dimensions without any intersection to the container/truck which has specific dimensions.

In this study the problem is about raw material and material flow between two plants that locates in Gönen and in Bursa. Trucks provide this flow in three shifts. Raw materials and materials are transported from Gönen production plant to Bursa assembly plant in different boxes and different quantities. An integer programming model was set for this transportation problem that has an objective function of transporting these raw material and material boxes by minimum number of trucks. Also the same problem was solved by generating an ant colony optimization based algorithm and in conclusion the results were compared.

Key words: Container/truck loading, integer programming, ant colony, optimization

2019, vii + 44 pages.

TEŞEKKÜR

"Farklı Lokasyonlarda Bulunan İki Fabrika Arasındaki Yarı Mamül ve Malzeme Akışının Optimizasyonu ve Standartlaştırılması" isimli bu yüksek lisans tez çalışmasında, otomotiv sektöründe faaliyet gösteren bir firmanın, gerçek bir lojistik problemini yüksek lisans eğitimi sürecinde öğrendiğimiz teknikleri kullanarak çözmeye çalıştık.

Çalışma süresince değerli bilgi ve tecrübelerini esirgemeden beni yönlendiren danışmanım Prof.Dr. Hüseyin Cenk ÖZMUTLU' ya teşekkürlerimi sunarım.

Ayrıca tüm eğitim hayatım boyunca bana her türlü desteği sağlayan sevgili anneme, babama ve bu süreçte beni yalnız bırakmayan eşime teşekkürü borç bilirim.

Elif KURTULUŞ
11/09/2019

İÇİNDEKİLER

Sayfa

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
CSCMP	Council of Supply Chain Management Professionals
GACO	Global Ant Colony Optimization
KKO	Karınca Kolonisi Optimizasyonu

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Dünyanın en büyük lojistik organizasyonu olan Tedarik Zinciri Uzmanları Konseyi'nin (Council of Supply Chain Management Professionals, CSCMP) tanımıyla lojistik; "Müşterinin ihtiyaçları doğrultusunda hizmetler de dahil olmak üzere tüm ürünlerin ve ilgili bilgilerin çıkış noktasından varış noktasına kadar etkili ve verimli bir biçimde taşınması ve depolanması için gerekli prosedürleri planlama, uygulama ve denetleme sürecidir." Bu tanım her yöndeki hareketi içermektedir.

Günümüzde teknolojinin gelişmesi ve küreselleşme ile artan ticaret hacmi dolayısıyla lojistik operasyonların üzerinde çalışmak zorunlu hale gelmiştir. Bu operasyonları hızlı, doğru ve düşük maliyetle gerçekleştirmek için matematiksel optimizasyon araçları yoğun bir şekilde kullanılmaktadır. Lojistik sistemlerin matematiksel modeller ve istatistiksel yöntemler kullanılarak tasarlanması, çeşitli amaçlar doğrultusunda çözüm alternatiflerinin değerlendirilmesi bu alandaki çalışmaların bir sonucudur. Bu alanda yapılan çalışmalar lojistik operasyonların verimli yürütülebilmesi için ağ tasarımı problemleri, envanter problemleri gibi ana problemler dışında spesifik lojistik problemlerine de odaklanmaktadır. Konteyner/tır yükleme problemleri de bu spesifik problemler kümesindedir. Optimal kapasite kullanımı ve minimal konteynır kullanımı operasyonların verimliliğini arttırmaktadır. Konteyner/tır yükleme problemleri, kutuların ölçüleri ve konteynır/tırların kapasiteleri çeşitli olduğundan NP-zor problem ailesindendir. Dikkate aldıkları amaçlara, ilgilendikleri boyutlara ve verilerin belirli olup olmamasına bağlı olarak çeşitli gruplara ayrılmaktadırlar.

Bu çalışmada farklı ambalaj, farklı kutu içi miktar ve farklı günlük tüketimi olan yarı mamül ve malzemelerin günlük talep ve taşıma kapasitesi kısıtları altında, tır hacmi maksimum kullanılarak taşınması amaçlanmıştır. Başka bir deyişle; amaç fonksiyonu kullanılan tır sayısını minimize etmek olan matematiksel model ile küçük boyutlu kutuların sadece hacim boyutu göz önüne alınarak tırlara atanması problemi çözülmüştür.

2. KURAMSAL TEMELLER VE KAYNAK ARAŞTIRMASI

2.1. Kuramsal Temeller

2.1.1. Lojistik Yönetimi

Lojistik Yönetimi; müşteri ihtiyaçlarına cevap verebilmek için üretim alanı ve tüketim noktası arasında gerçekleşen ileri ve geri yöndeki mal, hizmet ve bilgi akışı ile depolanmalarının yüksek verimle planlanması, uygulanması ve kontrolünü kapsayan tedarik zinciri sürecidir. Lojistik yönetiminin aşamaları; tedarik lojistiği, üretim lojistiği ve dağıtım lojistiği olarak gruplandırılabilir. Lojistik yönetiminin ilkeleri ise kısaca maliyetlerin düşürülmesi, uygun ambalajların seçilmesi, uygun lojistik modunun seçilmesi, sigortalama işlemleri, yükleme araçlarının optimizasyonu, mesafenin optimizasyonu ve teslimat biçimlerinin belirlenmesi olarak özetlenebilir (Palamutçuoğlu, 2012).

2.1.2. Konteynır/Tır Yükleme Problemi

Lojistik maliyetlerinin düşürülebilmesi için rota optimizasyonu ve yük konsolidasyonu gibi konuların yanında, konteynır/tır hacimlerini en iyi şekilde kullanmaya yönelik çalışmalar yapılmaktadır. Konteynır/tır yükleme problemi; üç boyutlu küçük kutuların, üç boyutlu büyük konteynır/tırlara atanması için geometrik bir atama modeli olarak tasarlanmaktadır. Genellikle tüm kutuların konteynır/tıra atanması ve bu kutuların çakışan lokasyonlara yerleştirilmemesi kısıtları altında ele alınmaktadır. Konteynır/tır yükleme problemleri, kutu ölçülerinin ve konteynır/tır kapasitelerinin çeşitli olmasından ötürü NP-zor problem ailesindendir. Maliyetleri minimize etmek ve konteynır/tır hacminden maksimum faydalanmak için konteynır/tır yükleme problemi araştırmacıların ilgi alanına girmektedir (Tekil ve Özkır 2016).

2.1.3. Tamsayılı Programlama

Doğrusal programlama problemleri modellenirken bazı karar değişkenlerinin tamsayı değeri alması gerektiğini görürüz. Bu durumu içeren problemler tamsayılı program olarak isimlendirilir ve bu tip programların çözümüne tamsayılı programlama denir. Sonlu sayıda seçenek içeren çözüm kümesindeki alternatiflerle ilgili kararların yap/yapma, evet/hayır gibi kesikli olmasından dolayı tamsayılı programlama sıklıkla kullanılmaktadır.

2.1.4. Sezgisel Algoritmalar

Tamsayılı değişkenlerle oluşturulan çoğu doğrusal optimizasyon modeli büyük ölçekli kombinasyonel ve doğrusal olmayan problemlerde yetersiz kalır. Sezgisel optimizasyon algoritmaları bu nedenle geliştirilmiştir. Kolay dönüşüm sağlamaları ve hesaplama yönlerinin güçlü olması sebebiyle sezgisel yöntemler son yıllarda oldukça yaygın kullanılmaktadır. Meta sezgisel yöntemler ise; deterministik yöntemlerin kısa sürede çözemediği büyük boyutlu problemlere yaklaşık bir çözüm bulan, problemden bağımsız algoritmalardır. Sezgisel yöntemlerin daha da geliştirilmesiyle arama uzayının hızlı ve verimli bir şekilde incelenmesi amaçlanır. Bu yöntemler her zaman en iyi çözümü bulmayı garanti etmez ancak büyük boyutlu problemlere etkili çözümler üretebilirler. (Eröz ve Tanyıldızı, 2018)

2.2. Kaynak Araştırması

Önder Karaoğlu 2017' de yaptığı "Akümülatör Geri Kazanım Ağı İçin Bir Tersine Lojistik Optimizasyonu" isimli yüksek lisans tez çalışmasında ülkemizde bulunan akümülatör üretim ve geri dönüşüm tesisleri ile geçici depoları inceleyerek farklı senaryolar altında kazancı hesaplayan bir tersine lojistik optimizasyonu uygulaması geliştirmiştir. Çalışmanın verileri belirsiz olduğundan pek çok varsayımda bulunulmuş ve karışık tamsayılı programlama yöntemiyle çözülen modelde belirtilen varsayımlar altında bulunan

sonuçlar karşılaştırılmıştır. Her senaryo için işleme/bertaraf/üretim ve taşıma maliyetlerindeki değişimin karlılığa olan etkisi gösterilmiştir. Son olarak da tersine lojistikte belirsizliklerin düşmesini sağlayabilecek yöntemlere değinilmiştir.

Çağatay Kalkancı 2009 yılında yaptığı "Coğrafi Bilgi Sistemleri Destekli Üretim ve Lojistik Optimizasyonu ve Asfalt Sektöründe Bir Uygulama" isimli yüksek lisans tezinde GPS uygulamaları ile araç takip sistemleri üzerinde çalışarak asfalt üretimi yapan bir şirketin dağıtım yaptığı lokasyonlar arasında ortaya çıkan problemleri incelemiştir.

Muharrem Bakkal ve Uğur Çelik 2011' de yayınlanan "Lojistik Yönetimi ve e-Lojistik" isimli kitapta lojistik operasyonlar yönetilirken yapılan işlemlere yer vermiştir. Ortaya çıkan akımların, gelişen teknolojinin ve yeni stratejilerin lojistik faaliyetler üzerinde yarattığı etki, değişim ve sağlanan avantajlar çalışmanın bütününü oluşturmaktadır.

Muharrem Bakkal ve Arif Oflaz 2011' de yayınladığı "Lojistik Bilgi Sistemleri" isimli kitapta ise lojistikle ilgili kavramlar, lojistik bilgi sistemleri, lojistik hizmet üreten ve alan işletmelerin bilgi sistemleri üzerinde durmuştur.

Sezgi Tekil ve Vildan Ç. Özkır "Konteynır Yükleme Problemlerinin İncelenmesi ve Lojistik Sektöründe Bir Uygulama" isimli çalışmalarında bir filtre fabrikasının lojistik operasyonlarını inceleyerek konteynırların karmaşık yükleme planlarının optimize etmeye çalışmışlardır. Kullanılan konteynır sayısını azaltmak ve yükleme hızını arttırmak amacıyla geliştirilen kutulama algoritmaları optimum kararların alınmasında kullanılmıştır.

Eliseu Junio Araujo ve arkadaşları 2014 yılında yayınlanan üç boyutlu konteynır yükleme problemine pareto gruplama uygulaması başlıklı çalışmalarında konteynırların gemiye yüklenmesi problemine meta sezgisel ve sezgisel yöntemleri birleştiren hibrit bir çözüm yaklaşımı getirmişlerdir.

Türkay Dereli, Gülesin Sena Daş, 2010 yılında yaptıkları “Konteynır Yükleme Problemleri İçin Karınca Kolonisi Optimizasyonu Yaklaşımı” isimli çalışmada konteynır/tır yükleme probleminin çözümü için karınca kolonisi optimizasyonunu temel alan iki yeni algoritma önermiştir. Parametrelerini deneysel tasarımla belirledikleri bu algoritmaların performansları test edilmiş ve sonuçlar literatürdeki diğer çalışmalarla karşılaştırılmıştır.

Araya ve Rift, 2014, konteynır yükleme problemine ateşböceği optimizasyonu yaklaşımı başlıklı çalışmalarında tek konteynır yükleme problemi için örnek veri kümeleri üzerinde çeşitli çözüm algoritmalarını uygulayarak sonuçları karşılaştırmışlardır.

Wei ve arkadaşları ise 2015 yılına ait çoklu konteynır yükleme maliyet minimizasyonu problemi için hedef odaklı prototip sütun türetme stratejisi isimli çalışmalarında probleme sütun türetme yöntemi ile hedef odaklı arama yöntemlerini birleştiren yeni bir yaklaşım önermişlerdir.

Timur Keskintürk ve Hasan Söylemez 2006 yılında gerçekleştirdikleri “Global Karınca Kolonisi Optimizasyonu” başlıklı çalışmada mevcut birçok karınca kolonisi optimizasyonu algoritmasından farklı olarak, karıncaların tam bir tur yapma veya tüm düğümlere uğrama zorunluluğu bulunmayan ve “Global Karınca Kolonisi Algoritması" adı verilen yeni bir teknik ortaya koymuştur. Bu algoritmanın mevcut algoritmalardan ayrıldığı diğer önemli nokta sistemde bölgesel feromon güncellemesi yapılmamasıdır.

Yüksel Yurtay, Nilüfer Yurtay, Eyüp Akçetin ve Alper Kılıç’ ın 2014 yılına ait “Konteynırda Yük Optimizasyonu: Örnek Uygulama” isimli çalışmalarında ise konteynırların karmaşık yükleme planları için yükleme hızını artırmayı amaçlayan bir karar destek sistemi geliştirilerek literatüre kazandırılmış, uygulamadan elde edilen sonuçlar örnek olarak sunulmuştur.

Lim ve arkadaşları ise 2013 yılında yayınlanan dingil ağırlık kısıtlı tek konteynır yükleme problemi başlıklı çalışmada yasal kısıtlara odaklanan entegre bir sezgisel çözüm yaklaşımı ortaya koymuşlardır.

3. MATERYAL VE YÖNTEM

Konteynr/tır yükleme problemi, üç boyutlu küçük kutuların, üç boyutlu büyük konteynır/tırlara yerleştirilmesini amaçlayan geometrik bir atama modeli olarak tasarlanır. Genel itibariyle bütün kutuların konteynır/tırlara atanması ve atanan kutuların çakışan lokasyonlara yerleştirilmemesi kısıtları altında çalıştırılmaktadır. Konteynır/tır yükleme problemi bir girdi minimizasyonu problemi olarak ele alındığında yedi problem tipi ortaya çıkmaktadır. Bu problem tipleri; tek boyutlu stok kesme problemi, çok boyutlu stok kesme problemi, tek tipli kutulama problemi, çok tipli kutulama problemi, residual kutulama problemi, residual stok kesme problemi ve açık boyutlu problemlerdir. Konteynır/tır yükleme problemleri çıktı maksimizasyonu problemi olarak ele alındığında da yine yedi problem tipi ile karşılaşılır. Bunlar ise; tek büyük kutu yerleştirme problemi, çoklu eş boyutlu büyük kutu yerleştirme problemi, eş kutulu paketleme problemi, tekli sırt çantası problemi, çoklu karma boyutlu büyük kutu yerleştirme problemi, çoklu karma sırt çantası problemi, çoklu eş sırt çantası problemleridir (Tekil ve Özkır 2016).

Optimizasyon kelimesinin anlamı en iyilemektir. Problemin belirli sınırlar içerisinde kalan tüm çözümlerinin arasından en iyi çözümü elde etmek olarak özetlenebilir. Optimizasyon problemi ise mevcut kısıtlar altında parametrelerin alması gereken değerlerin bulunmasını içeren bir problem olarak tanımlanabilir. Optimizasyonun performansını belirleyen değişkenlere karar değişkenleri denir ve karar değişkenlerinin hedeflenen amacı nasıl etkilediklerinin analitik olarak ifade edilmesiyle amaç fonksiyonu oluşturulur (Eröz ve Tanyıldızı 2018). Optimizasyon algoritmalarının pek çoğu, amaç fonksiyonunun belirlenmesi ve sistemin modellenebilmesi için matematiksel modellere ihtiyaç duyar. Ancak tamsayı karar değişkenlerinin kullanıldığı doğrusal modeller büyük boyutlu kombinatorik problemlerin çözümünde ve doğrusal olmayan problemlerin çözümünde yetersiz kalır. Karmaşık sistemlerin matematiksel olarak modellenmesi genellikle zordur ve model oluşturulsa bile çözüm süresi makul olmadığından kullanılmamaktadır.

Bu çalışmada ele alınan problemde; Gönen'de bulunan üretim tesisi ve Bursa'da bulunan montaj fabrikası arasında sürekli yarı mamül ve malzeme akışı bulunmaktadır. Bu akış her gün üç vardiya süresince çalışan tırlar ile sağlanmaktadır. Tırlar farklı kutu içi

miktarlara sahip yarı mamül ve malzemeleri (kılıf ve kesik kumaş) Gönen üretim tesisinden Bursa montaj fabrikasına taşımaktadır. Bu taşıma optimizasyonu problemi için; amaç fonksiyonu minimum tır sayısı ile akışı sağlamak olan ve tamsayılı karar değişkenlerinin kullanıldığı bir matematiksel model oluşturulmuştur. Problem mevcut günlük tüketim miktarları için çözüldüğünde 15 karar değişkeni, 11 kısıtı olan ve 12 iterasyon sonucu ulaşılan en iyi çözüm elde edilmektedir. Problemi firmanın 2021 yılı talep öngörüleri için çözdüğümüzde ise boyutu büyümekte ve 20 karar değişkenli, 14 kısıtlı, 22 iterasyon sonunda elde edilen en iyi çözümle karşılaşılmaktadır. Ancak gelecekte günlük tüketimin 3 katına çıkacağı varsayımı altında matematiksel model üçüncü veri grubu ile çalıştırıldığında problemin boyutu programın kapasitesini aştığından en iyi çözüme ulaşılamamaktadır. Bu durumda problemi çözebilmek için farklı bir yaklaşıma ihtiyaç duyulmaktadır. Bu yaklaşım arama uzayını aktif ve verimli bir şekilde incelemek amacıyla geliştirilen uygun bir sezgisel yöntem kullanmaktır.

Konteynır/tır yükleme probleminde amaç ölçüleri belirli *n* tane küçük kutunun, boyutları belli bir konteyner/tırın içine yerleştirilmesiyle herhangi bir çakışma olmadan mevcut konteyner/tır hacmini en iyi şekilde kullanmak olduğundan bir hacim doruklaştırma problemi olarak ele alınmaktadır. Literatürdeki mevcut çalışmalar incelendiğinde konteynır/tır yükleme problemine yönelik yapılmış çalışmalarda karınca kolonisi optimizasyonu yaklaşımının yaygın olarak kullanılmadığı görülmektedir. Konuyla ilgili Liang ve arkadaşları tarafından 2007 yılında yapılmış bir çalışma ve Tülay Dereli ile Gülesin Sena Daş tarafından 2010 yılında yapılmış bir çalışmaya rastlanmıştır. Liang ve arkadaşları tarafından yapılan çalışmada konteynır yükleme problemini çözmek için karınca kolonisi optimizasyonu ve genetik algoritmayı birlikte kullanan melez bir yöntem kullanılmıştır. Karınca kolonisi optimizasyonu bu çalışmada, konteynıra yerleştirilecek kutulardan kuleler elde etmek için kullanılmıştır. Tülay Dereli ve Gülesin Sena Daş tarafından yapılan çalışmada ise konteynır/tır yükleme probleminin çözümüne yönelik karınca kolonisi algoritması kullanan iki çözüm yöntemi önerilmiş ve bu yaklaşımlar literatürdeki diğer yöntemlerle çeşitli test problemleri üzerinden karşılaştırılmıştır.

Bir gerçek yaşam problemini ele aldığımız bu çalışmada ise Timur Keskintürk ve Hasan Söyler tarafından 2006 yılında geliştirilmiş olan Global Karınca Kolonisi Algoritması tekniği uygulanarak çözüme ulaşılmıştır. Önerilen algoritma Python programlama dilinde kodlanarak mevcut durum için elde edilen sonuçlar karşılaştırılmıştır.

3.1. Matematiksel Model

3.1.1. Veri Kümesinin Oluşturulması

Gönen' de bulunan üretim tesisinde kesik kumaş ve kılıf üretilmektedir. Kesik kumaşlar metal ambalajlar ile kılıflar ise karton ambalajlar ile taşınmaktadır. Tek tip metal ambalaj ve tek tip karton ambalaj kullanılmaktadır. Karton ambalajın, metal ambalajın ve tırın ölçüleri aşağıdaki tabloda belirtilmiştir.

Çizelge 3.1. Tırın iç ölçüleri ve kullanılan ambalaj ölçüleri

Ölçüler	Karton Amb	Metal Amb	Tır
En	1,2 m	1,2 m	2,4 m
Boy	1 m	0,9 m	13,6 m
Yükseklik	0,9 m	1,4 m	2,9 m

Gönen'de üretilen kesik kumaş ve kılıflar Bursa' daki müşteri fabrikaya tır ile taşınmaktadır. Mesafe yaklaşık 135 km olduğundan, dolu bir tırın Gönen' deki üretim tesisinden Bursa' daki montaj fabrikasına gitmesi ve geri dönmesi, yükleme boşaltma süreleri ile brlikte yaklaşık bir vardiya sürmektedir. Müşteri konumundaki firma, üç vardiya süresince bu taşıma işine tahsis edilmek üzere bir tır satın alıp yaklaşık 120 000 € luk yatırım yapmış bulunmaktadır.

Toplam 23 tip kılıf ve 21 tip kesik kumaş taşınmaktadır. Her tip yarı mamülün günlük tüketim adedi ve kutu içi miktarı farklıdır. Ağırlıkları ve taban alanlarının farklı olması sebebiyle ne stok bölgesinde ne de tırın içerisinde karton ambalaj ile metal ambalaj eşleştirilememekte yani üst üste konamamaktadır. Tırın yüksekliği sebebiyle en fazla üç

karton veya metal ambalaj üst üste konabilmektedir. Üst üste konan ambalajlarda en ağır ambalaj en altta, en hafif ambalaj en üstte olacak şekilde istiflenmektedir. Karton ambalaj ile taşınan kılıf tipleri, günlük tüketim adetleri, kutu içi miktarları ve parça ağırlıkları Çizelge 3.2' de belirtilmiştir.

Çizelge 3.2. Karton ambalaj ile taşınan kılıf tipleri, günlük tüketim adetleri, kutu içi miktarları ve parça ağırlıkları

Ürün Kodu	Günlük Tüketim	Kutu İçi Miktar	Parça Ağırlığı (kg)
E12FSCRH	716	120	0,43
E12FSCLH	716	120	0,43
E12FSBRH	716	60	0,66
E12FSBLH	716	60	0,66
E12RER13	716	50	0,45
E12RER23	716	30	0,61
E12RER11	716	30	0,70
E30FSCRH	106	60	0,67
E30FSCLH	106	60	0,67
E30RER13	106	30	0,70
E30RER23	106	15	0,86
E30RER11	106	20	1,22
E3LFSCRH	131	30	0,78
E3LFSCLH	131	30	0,78
E3LRER13	131	30	0,70
E3LRER23	131	15	0,86
E3LRER11	131	15	1,22
E4LFSCRH	28	30	0,78
E4LFSCLH	28	30	0,78
E4LRER13	28	30	0,70
E4LRER23	28	20	0,86
E4LRER11	28	15	1,22

Metal ambalaj içerisinde taşınan kesik kumaş tipleri, günlük tüketim adetleri, kutu içi miktarları ve parça ağırlıkları da Çizelge 3.3' de belirtilmiştir.

Çizelge 3.3. Metal ambalaj içerisinde taşınan kesik kumaş tipleri, günlük tüketim adetleri, kutu içi miktarları ve parça ağırlıkları

Ürün Kodu	Günlük Tüketim	Kutu İçi Miktar	Parça Ağırlığı (kg)
E30FSBRH	106	20	0,67
E30FSBLH	106	20	0,67
E3LFSBRH	131	20	1,10
E3LFSBLH	131	20	1,10
E4LFSBRH	28	20	1,10
E4LFSBLH	28	20	1,10
RSFFSCRH	42	80	0,48
RSFFSCLH	42	80	0,48
RSFFSBRH	42	40	0,72
RSFFSBLH	42	40	0,72
RSFRER13	42	30	0,70
RSFRER23	42	30	0,86
RSFRER11	42	30	1,22
RSLFSCRH	5	30	0,78
RSLFSCLH	5	30	0,78
RSLFSBRH	5	20	1,10
RSLFSBLH	5	20	1,10
RSLRER13	5	20	0,70
RSLRER23	5	30	0,86
RSLRER11	5	15	1,22

Kutu sayısını görece olarak azaltmak ve tır içi yerleşimde özdeş kutuların eşleşmesi kısıtını sağlamak amacıyla karton kutular ve metal kutular kendi içinde "kuleler" şeklinde gruplandırılarak modele dahil edilmiştir. Bu yaklaşım karınca kolonisi optimizasyonu algoritmasını uygularken de düğüm sayısını görece olarak azalttığından oldukça kullanışlıdır. "Kuleler" oluşturmak için ilk aşamada karton ambalajlar ve metal ambalajlar

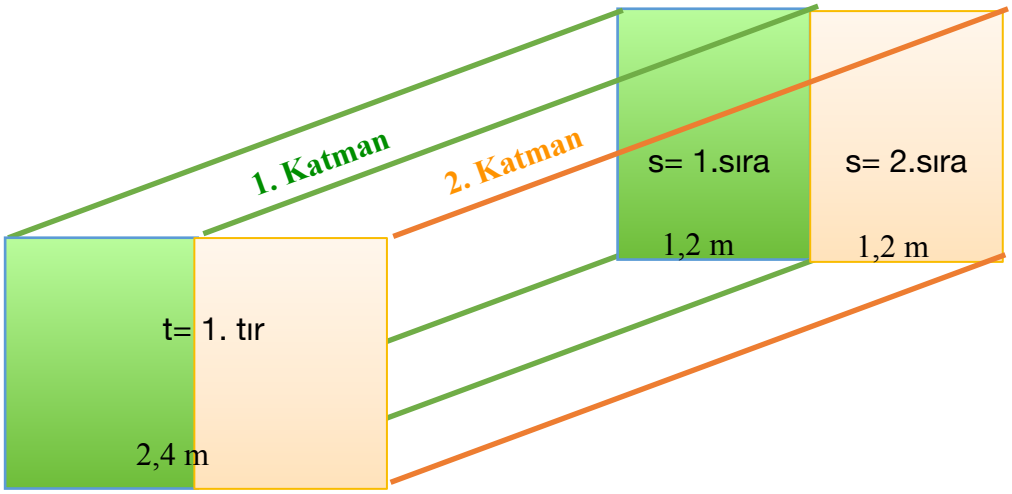
kendi içinde, ayrı ayrı ağırlıklarına göre sıralanır. Yükseklik farkından dolayı üste üste en fazla üç karton ambalaj veya iki metal ambalaj istiflenebildiğinden ağırlıklarına göre sıralanan karton ambalajlar üçerli, metal ambalajlar ise ikişerli gruplanır. Böylece modele dahil edilecek "karton kuleler" ve "metal kuleler" elde edilmiş olur.

3.1.2. Modelde Kullanılan İndisler

i = yerleştirilen kule tipi (karton, metal)
t = tır sayısı (1,2,3,4)
s = kulenin yerleştirildiği sıra numarası (1,2)

i = 1 karton kuleleri, i = 2 ise metal kuleleri temsil etmektedir. Modelde her vardiyada kullanılan tır bir rakamla temsil edilmiştir. Başka bir deyişle; t = 1 birinci vardiyada taşıma yapan tırı, t = 2 ikinci vardiyada taşıma yapan tırı ve t = 3 üçüncü vardiyada taşıma yapan tırı temsil etmektedir. Mevcut tır bir günde üç vardiya taşıma yapabildiğinden (lokasyonlar arası mesafe buna uygun olduğundan) dördüncü tıra ihtiyaç duyulması, yeni bir tır alınması ve tır yatırımı yapılması gerektiği anlamına gelmektedir.

Karton ambalajın ve metal ambalajın eni eşit uzunlukta olup 1,2 metredir. Tırın eni ise 2,4 metredir. Bu durumda tırın içinde yan yana sadece iki kule yerleştirilebilir. Enleri eşit uzunlukta olduğundan yan yana yerleştirilen kulelerin karton veya metal olması farketmemektedir. Bu sebeple; oluşturduğumuz modelde her tır 1,2 metre ene sahip sanal iki katmana ayrılarak, kuleler bu katmanlara yerleştirilmektedir. Modelde kullanılan s indisi her tır için birinci ve ikinci katmanı/sırayı temsil etmektedir.



Şekil 3.1. Tır içi yerleşimde kullanılan özdeş katmanlar

3.1.3. Karar Değişkenleri

Y_t = t tırının kullanılıp kullanılmaması durumu (0, 1)

X_{its} = i kule tipinden t tırının s sırasına yerleştirilecek kule adedi

Tırın uzunluğu 13,6 metre, karton kutu/kulenin uzunluğu 1,0 metre, metal kule/kutunun uzunluğu 0,9 metre olduğundan bir sıraya maksimum 13 adet karton kule veya 15 adet metal kule yerleştirilebilmektedir. Dolayısıyla X karar değişkeninin alabileceği değer aralığı aşağıdaki gibidir:

$$0 \leq X_{1ts} \leq 13 \quad (3.1)$$

$$0 \leq X_{2ts} \leq 15 \quad (3.2)$$

3.1.4 Kısıtlar

Karton kuleler için günlük talebin karşılanması kısıtı :

$$X_{111} + X_{112} + X_{121} + X_{122} + X_{131} + X_{132} = D_1 \quad (3.3)$$

D_1 = Karton kuleler için günlük tüketim adedi

Metal kuleler için günlük talebin karşılanması kısıtı :

$$X_{211} + X_{212} + X_{221} + X_{222} + X_{231} + X_{232} = D_2 \quad (3.4)$$

D_2 = Metal kuleler için günlük tüketim adedi

Yerleştirilen kulelerin toplam uzunluğunun tır uzunluğunu aşmaması kısıtı:

$$(1 * X_{111}) + (0{,}9 * X_{211}) \leq 13.6 \quad (3.5)$$

$$(1 * X_{112}) + (0{,}9 * X_{212}) \leq 13.6 \quad (3.6)$$

$$(1 * X_{121}) + (0{,}9 * X_{221}) \leq 13.6 \quad (3.7)$$

$$(1 * X_{122}) + (0{,}9 * X_{222}) \leq 13.6 \quad (3.8)$$

$$(1 * X_{131}) + (0{,}9 * X_{231}) \leq 13.6 \quad (3.9)$$

$$(1 * X_{132}) + (0{,}9 * X_{232}) \leq 13.6 \quad (3.10)$$

Kullanılmayan tıra atama yapılmaması kısıtı:

$$X_{111} + X_{112} + X_{211} + X_{212} \leq 1000 * Y_1 \quad (3.11)$$

$$X_{121} + X_{122} + X_{221} + X_{222} \leq 1000 * Y_2 \quad (3.12)$$

$$X_{131} + X_{132} + X_{231} + X_{232} \leq 1000 * Y_3 \quad (3.13)$$

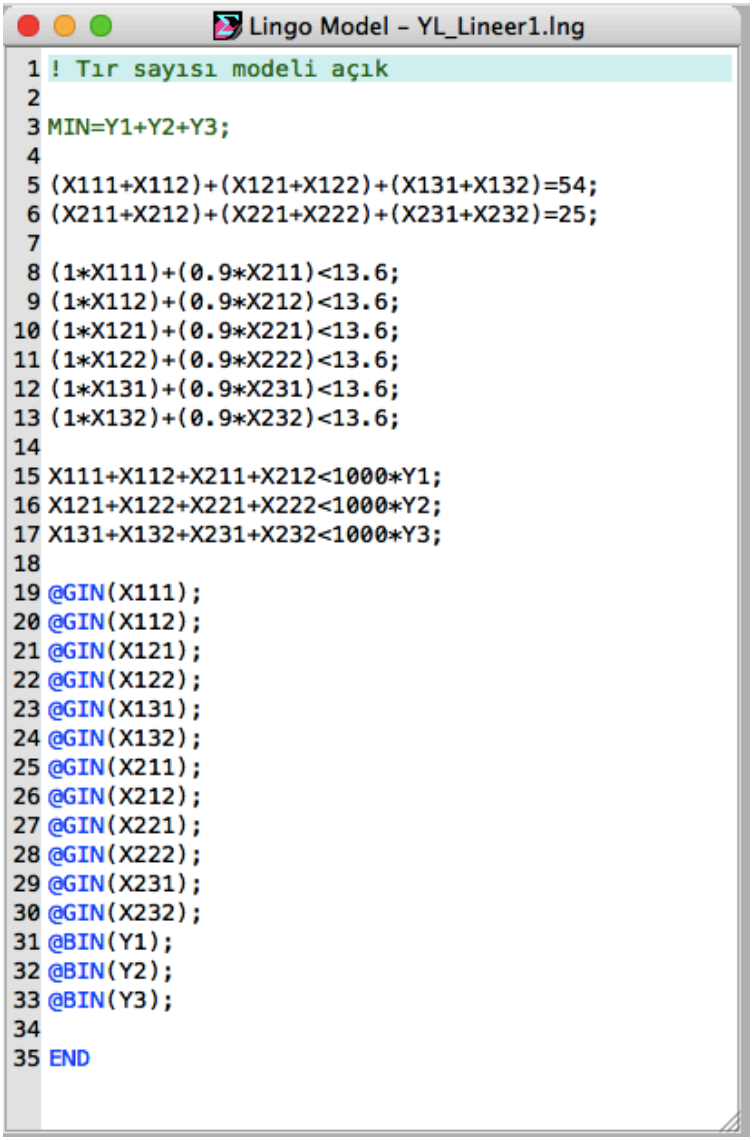
İkili değişken kısıtı:

$$Y_1, Y_2, Y_3 \in (0, 1) \quad (3.14)$$

3.1.5 Amaç Fonksiyonu

Amaç fonksiyonu minimum tır sayısı ile taşımayı gerçekleştirmek olduğundan;

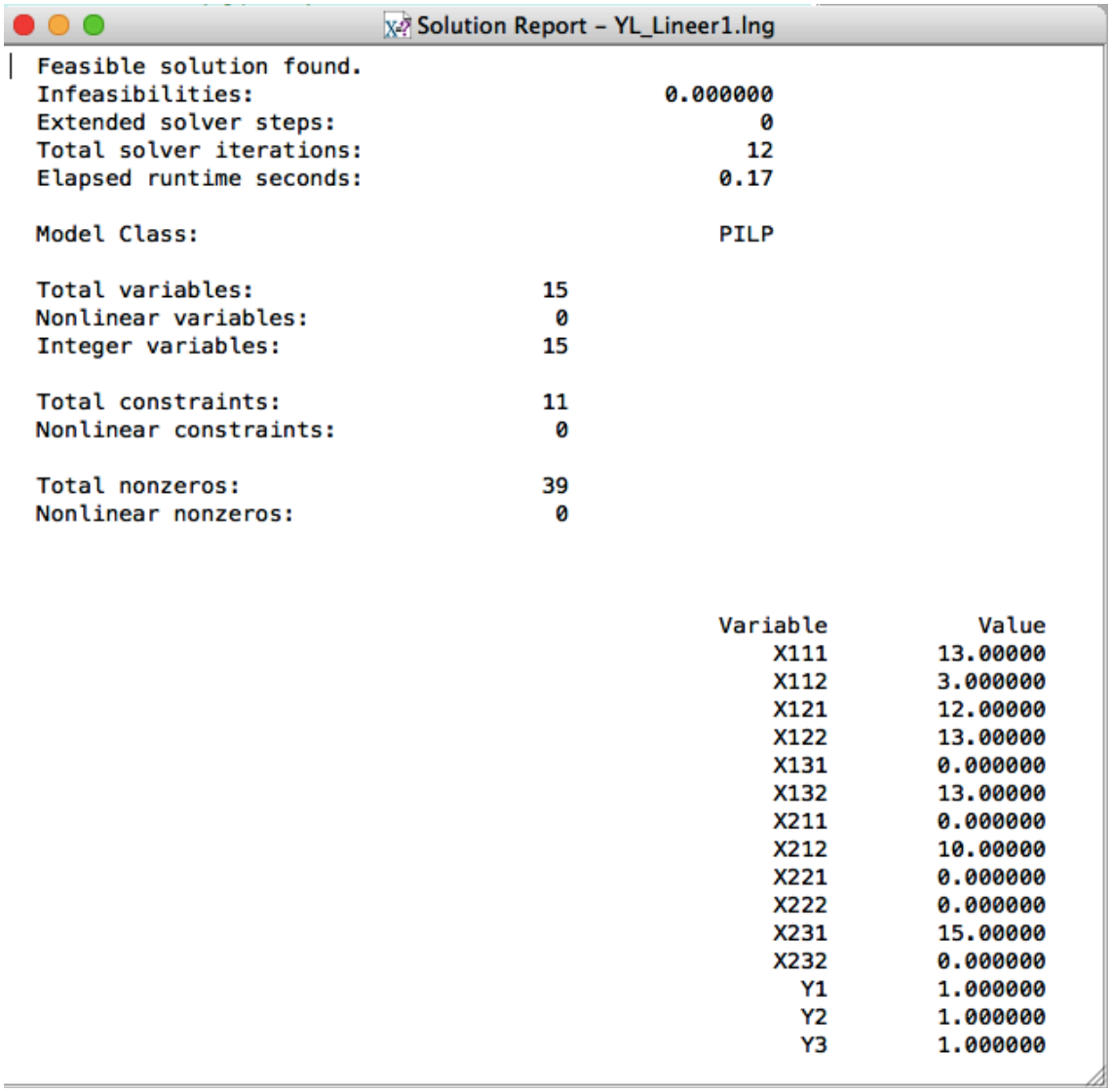
$$Min\ z = Y_1 + Y_2 + Y_3 \quad (3.15)$$



Şekil 3.2. Matematiksel modelin LINGO programı ekran görüntüsü

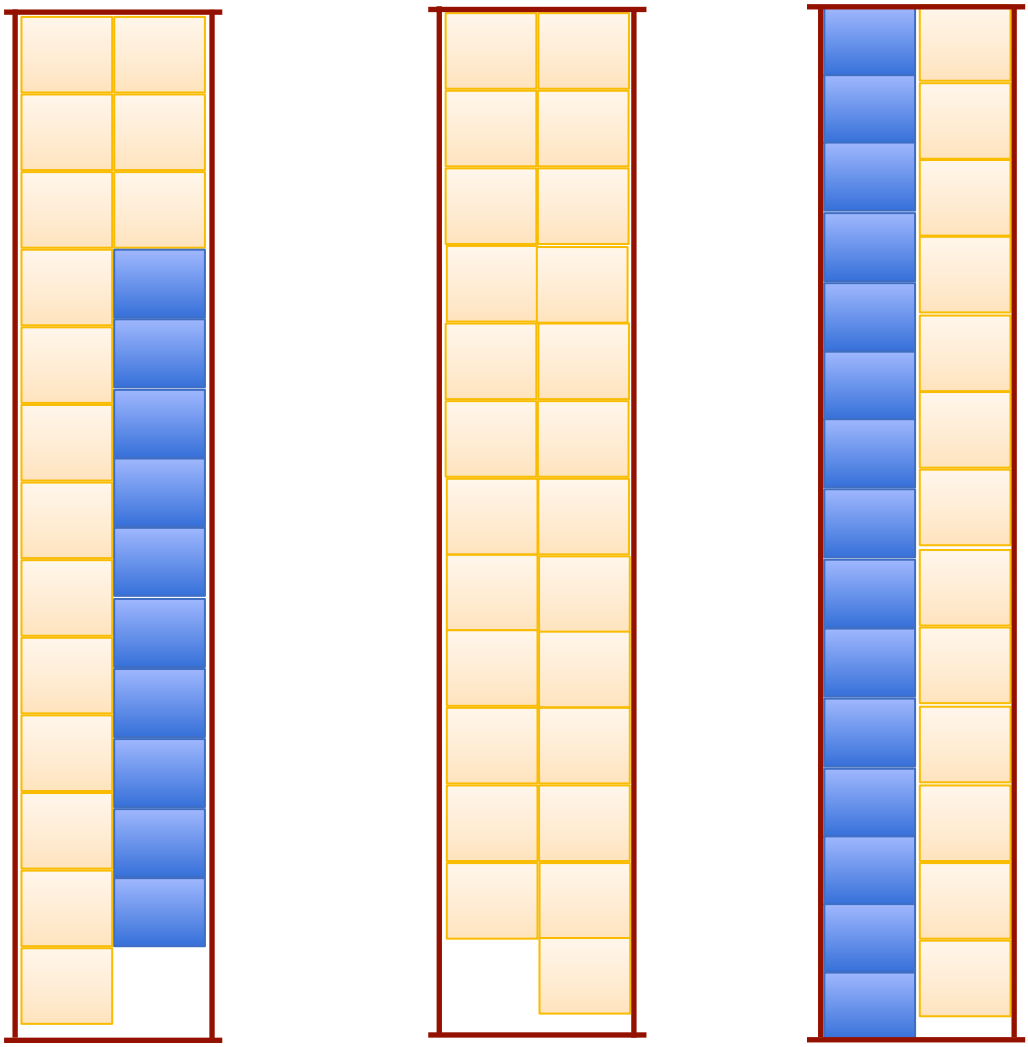
3.1.6 Matematiksel Modelin Çözüm Raporu

Oluşturduğumuz matematiksel model LINGO programı ile çözüldüğünde elde edilen çözüm raporu aşağıdaki gibidir.



Şekil 3.3. Matematiksel modelin çözüm raporu ekran görüntüsü

Bu çözüm raporuna göre üç vardiya çalışacak bir tır taşıma için yeterli olacaktır. Birinci vardiyada taşıma yapan tırın (s = 1) birinci katmanına 13 adet karton kule, (s = 2) ikinci katmanına ise 3 adet karton 10 adet metal kule yerleştirilmeli. İkinci vardiyada taşıma yapacak tırın birinci katmanına 12 adet karton, ikinci katmana ise 13 adet karton kule yerleştirilmeli. Üçüncü vardiyada taşıma yapacak tırın da (s = 1) birinci katmanına 15 adet metal kule; (s = 2) ikinci katmanına ise 13 adet karton kule yerleştirilmeli. Üç vardiya için tır içi yerleşim aşağıdaki gibi olmalıdır:



Şekil 3.4. Tır içi yerleşimin üstten görünüşü

Bu yerleşim planına göre üç vardiyada toplam 54 adet karton kule, 25 adet de metal kule Gönen üretim tesisinden Bursa montaj fabrikasına taşınarak günlük tüketim ihtiyacı karşılanmış oluyor. Mevcut tır bir günde üç tur yapabildiği, yani üç vardiya çalışabildiğinden mevcut talebi karşılamak için yeni bir tır yatırımı yapılması gerekmemektedir.

İlerleyen bölümlerde model farklı veri gruplarıyla çalıştırılarak firmanın talep tahminlerine bağlı elde edilen sonuçlar karşılaştırılmıştır.

3.2. Karınca Kolonisi Optimizasyonu

3.2.1. Karınca Kolonisi Algoritması

Kolektif böcek davranışlarından ilham alan bir yöntem olan Karınca Kolonisi Optimizasyonu (KKO) pratikte en sık kullanılan meta sezgisellerden biridir. Başlangıçta kombinatorik optimizasyon problemlerine odaklanan ve karınca kolonisi optimizasyonu adı altında toplanan bu yaklaşım çoğunlukla zor kombinatorik optimizasyon problemleri için kullanılan çok etmenli bir yöntemdir. Karıncalar, kolektif davranışlar gösteren ve koloniler halinde yaşayan böceklerdir. Tercihleri koloninin bir bileşeninden çok tümünün yaşamına yöneliktir. Gerçek karıncalar yiyecek kaynağından yuvalarına kadar görsel bir izden faydalanmadan feromon bilgisini kullanarak en kısa yolu bulabilirler. Feromon vücutlarının salgıladığı kimyasal bir maddedir ve bununla haberleşirler. Yuvadan yiyecek kaynağına doğru yola çıkan ilk karıncaların bıraktığı feromon koloninin geri kalanının seçiminde etkili olmaktadır. Karıncalar, yiyecek kaynağına giden yolda feromon izi yoksa rassal olarak hareket eder. Aksi takdirde feromon izini fark eder ve tercih ederler. Böylece yol eklenen feromon ile yeniden işaretlenir ve bu izi takip edecek daha fazla sayıda karıncayı çeker (Dereli ve Daş 2010).

Karıncaların davranışından ilham alarak hazırlanan ilk algoritmalar Dorigo tarafından ortaya konmuştur. Karınca Kolonisi Optimizasyonu KKO algoritması gerçek karıncaları taklit eden yapay karıncalar kullanmaktadır. Algoritmanın ilk örnekleri gezgin satıcı problemi ve diğer rotalama problemlerine ilişkin olmasına rağmen günümüzde KKO algoritması en kısa ortak üst dizi, çoklu sırt çantası, genelleştirilmiş atama ve kısıt sağlama problemlerinin de aralarında bulunduğu geniş bir optimizasyon alanına uyarlanmıştır. Son zamanlarda sürekli optimizasyon problemlerinin çözümünde de güncel ve genişletilmiş karınca kolonisi optimizasyonu yaklaşımları kullanılmaya başlamıştır (Dereli ve Daş 2010).

3.2.2. Konteynır Yükleme Problemiyle KKO Algoritmasının Eşleştirilmesi

Konteynır/tır yükleme probleminin KKO algoritmasıyla çözülebilmesi için, problem bir çizge şeklinde tasarlanmaktadır. Başka bir ifadeyle, *n tane* küçük kutu içeren bir konteynır/tır yükleme problemi *n* tane düğümü olan bir çizge olarak tasarlanır. Her düğüme sanal olarak bir küçük kutu yerleştirilir. Aynı yaklaşımla popülasyondaki her karıncanın da bir konteynır/tıra sahip olduğu düşünülür. Her karıncanın, turu süresince uğradığı her düğümde bulunan kutuyu kendi konteynır/tırına yerleştirdiği kabul edilir. Bu algoritmayla küçük kutular boyutları belirli bir konteynır/tıra, üç boyutlu olarak ve konteynır boyutları dahilinde lokasyonlarda çakışma olmadan yerleştirilmekte ve doluluk oranı hesaplanmaktadır.

3.2.3. Önerilen Global Karınca Kolonisi Algoritması

Timur Keskintürk ve Hasan Söyler 2006 yılında gerçekleştirdikleri "Global Karınca Kolonisi Optimizasyonu" isimli çalışmalarında Global Karınca Kolonisi Algoritması (Global Ant Colony Optimization) (GACO) adını verdikleri yeni bir karınca kolonisi optimizasyon tekniği ortaya koymuşlardır.

Bu yeni algoritmada, mevcut pek çok karınca kolonisi optimizasyonu sisteminden farklı olarak sanal karıncaların tam bir tur yapma veya tüm düğümlere uğrama gibi bir zorunlulukları yoktur. Ortaya konan bu yeni algoritmada, herhangi bir düğümden başlanabilir ve herhangi bir düğüme uğrayarak çözüm alternatifi geliştirilebilir. Geliştirilmiş KKO algoritmalarından diğer önemli farkı ise bölgesel feromon güncellemesinin olmayışıdır.

Bir gerçek yaşam problemi üzerinde yaptığımız bu çalışmada, Global Karınca Kolonisi Algoritması konteynır/tır yükleme problemine uygulanarak elde edilen sonuçlar matematiksel modelin çözümü ile karşılaştırılmıştır. Karton kutu ve metal kutuların eşleşmemesi, başka bir deyişle üst üste istiflenememesi kısıtını sağlayabilmek için yine karton kutular ve metal kutular kendi içinde "kuleler" şeklinde gruplandırılmıştır. Bu yaklaşım düğüm sayısını görece olarak azaltacağından oldukça kullanışlıdır. "Kuleler" oluşturmak için ilk aşamada karton ambalajlar ve metal ambalajlar kendi içinde ayrı ayrı, ağırlıklarına göre sıralanır. Yükseklik farkından dolayı üste üste en fazla üç karton

ambalaj veya iki metal ambalaj istiflenebildiğinden karton ambalajlar üçerli, metal ambalajlar ise ikişerli gruplanır. Böylece modele dahil edilecek “karton kuleler” ve “metal kuleler” elde edilmiş olur.

İkinci aşama karınca sayısının belirlenmesidir. Karınca sayısının fazla olması çözümde iyileşme sağlamakta ancak hesaplamaları arttırması nedeniyle işlem sürelerinin uzamasına neden olmaktadır. Gezgin satıcı problemlerinde yapılan birçok deneme sonrası karınca sayısının şehir sayısına eşit olmasının en uygun yaklaşım olduğu sonucuna varılmıştır (Keskintürk ve Söyler 2006). Bu çalışmada da karınca sayısı düğüm sayısına yani yerleştirilecek kule sayısına eşit alınmıştır.

Algoritmanın temel adımlarını şu şekilde özetleyebiliriz:

Çizelge 3.4. Meta sezgisel algoritmanın temel adımları

Adım 1	Problem verisini al ve probleme ait "yerleştirilecek kuleler” kümesini oluştur.
Adım 2	Her düğüme bir kule yerleştir.
Adım 3	Karınca sayısı=düğüm sayısı olacak şekilde her karıncayı bir düğüme yerleştir.
Adım 4	Her bir karınca için sonraki düğümü rassal olarak seç.
Adım 5	Seçilen kuleyi tıra yerleştir ve bu kuleyi “yerleştirilecek kuleler” kümesinden çıkart.
Adım 6	Tır içerisinde kalan boş hacmi hesapla.
Adım 7	Tırın içerisinde yeni bir kule yerleştirmek için yeterli hacim var mı? Varsa Adım 4’e git, yoksa Adım 8’e git.
Adım 8	Tırın doluluğunu hesapla ve dur.

3.2.4. Uygulanan Algoritmanın Kodlanması

Bu çalışmada kullanılan global karınca kolonisi algoritması Python 3.7.3 programa dili ile kodlanmıştır. Bu programlama dili birimsel, etkileşimli, nesne yönelimli ve yüksek seviyeli bir dildir. Girdilere dayanan basit söz dizimi, dilin öğrenilmesini ve hatırlanmasını kolaylaştırmaktadır. Bu özellik söz dizimi ayrıntılarıyla vakit kaybetmeden programlama yapmaya başlanabilen bir dil olma esnekliği kazandırır. Guido Van Rossum tarafından tasarlanmış olup, ilk çıkışı 1991 yılındadır.

Python programları C ve C++ gibi dillerden farklı olarak derlemeye gerek olmadan çalıştırılabilmektedir. Derleme işlemi ortadan kalktığından oldukça hızlı bir şekilde program geliştirilebilir.

Python uygulamaları GNU/Linux, Windows, Mac OS X, AS/400, BeOS, MorphOS, MS-DOS, OS/2, OS/390, z/OS, RiscOS, S60, Solaris, VMS, Windows CE, HP-UX, iOS ve Android gibi pek çok farklı ortamda geliştirilebilir. Ayrıca herhangi bir ortamda hazırlanan bir Python programı, herhangi bir değişikliğe gerek kalmadan ya da ufak değişikliklerle farklı ortamlarda çalıştırılabilir.

Hazırladığımız global karınca kolonisi algoritmasının, Python programlama dilinde bir tır için örnek olarak yazılan kodu ve bu kodun 1,8 GHz Intel Core i5 MacBook Air bilgisayarla çalıştırılmasından elde edilen sonuç ek' te sunulmuştur. Bu çözüme göre birinci tıra toplam 16 metal kule ve 9 karton kule yerleştirilerek toplam 77,54 metreküplük hacim elde edilmiştir.

4. BULGULAR VE TARTIŞMA

Üçüncü bölümde matematiksel modeli 54 karton kule ve 25 metal kuleden oluşan ilk veri grubu ile çalıştırmış, ilerleyen bölümlerde farklı veri gruplarını analiz edeceğimizi belirtmiştik. İlk veri grubunda günlük talepler; birinci vardiya tırına 16 karton kule ve 11 metal kule yerleştirilerek, ikinci vardiya tırına 20 karton kule ve 7 metal kule yerleştirilerek, üçüncü vardiya tırına ise 18 karton kule ve 7 metal kule yerleştirilerek karşılanıyordu. Bu çözüm ile birinci vardiya tırında 85,104 metreküp (%89 doluluk), ikinci vardiya tırında 85,968 metreküp (%90 doluluk) ve üçüncü vardiya tırında 79,488 metreküplük (%83 doluluk) taşıma yapılmaktadır. Tırın su dolu hacminin 96 metreküp, firmanın doluluk oranı hedefinin de %85 olduğu göz önünde bulundurulduğunda, elde ettiğimiz doluluk oranının gayet iyi olduğu rahatlıkla söylenebilir. Uyguladığımız global karınca kolonisi algoritmasını örnek bir tır için çalıştırdığımızda ise 77,54 metreküplük hacim elde edildi. Matematiksel model beklendiği üzere daha iyi bir çözüm sunmaktadır.

İkinci veri grubunda ise firmanın 2021 yılına ait talep öngörüleri çalışılmıştır. Tıpkı birinci veri grubunda olduğu gibi her kesik parça ve kılıf için günlük talebi kutu içi miktara bölerek günlük kutu ihtiyacı bulunmaktadır. Kutu adetlerini karton ve metal olarak ikiye ayırıp kendi içerisinde üçerli ve ikişerli gruplayarak “yerleştirilecek kuleler” veri kümesi oluşturulur. Tüm bu işlemler Excel programı üzerinde yapılmıştır. Sonuçta karşımıza 2021 yılı talep öngörüsü olarak günlük 178 karton kutu ve 53 metal kutu çıkmaktadır. Bu sonuç yerleştirilecek kuleler veri kümemizde toplam 60 karton ve 27 metal kule olduğu anlamına gelir.

İkinci veri grubunu içeren tablo, LINGO programında oluşturduğumuz ikinci modelin ekran görüntüsü ve ikinci modelin çözülmesi ile elde edilen sonuçların ekran görüntüleri aşağıdaki gibidir.

Çizelge 4.1. Firmanın 2021 yılına ait talep öngörüleri

BJA	Ürün Tipi	Günlük Tüketim	Kutu İçi Adet	Ambalaj Tipi
E1-E2	FSC RH	745	120	Karton
	FSC LH	745	120	Karton
	FSB RH	745	60	Karton
	FSB LH	745	60	Karton
	Rear 1/3	745	50	Karton
	Rear 2/3	745	30	Karton
	Rear 1/1	745	30	Karton
E3	FSC RH	127	60	Karton
	FSC LH	127	60	Karton
	FSB RH	127	20	Metal
	FSB LH	127	20	Metal
	Rear 1/3	127	30	Karton
	Rear 2/3	127	15	Karton
	Rear 1/1	127	20	Karton
E3 LEATHER	FSC RH	144	30	Karton
	FSC LH	144	30	Karton
	FSB RH	144	20	Metal
	FSB LH	144	20	Metal
	Rear 1/3	144	30	Karton
	Rear 2/3	144	15	Karton
	Rear 1/1	144	15	Karton
E4 LEATHER IP	FSC RH	31	30	Karton
	FSC LH	31	30	Karton
	FSB RH	31	20	Metal
	FSB LH	31	20	Metal
	Rear 1/3	31	30	Karton
	Rear 2/3	31	20	Karton
	Rear 1/1	31	15	Karton
RS SPORT FABRIC	FSC RH	46	80	Metal
	FSC LH	46	80	Metal
	FSB RH	46	40	Metal
	FSB LH	46	40	Metal
	Rear 1/3	46	30	Metal
	Rear 2/3	46	30	Metal
	Rear 1/1	46	30	Metal
RS SPORT LEATHER	FSC RH	8	30	Metal
	FSC LH	8	30	Metal
	FSB RH	8	20	Metal
	FSB LH	8	20	Metal
	Rear 1/3	8	20	Metal
	Rear 2/3	8	30	Metal
	Rear 1/1	8	15	Metal

```
MIN=Y1+Y2+Y3+Y4;

X111+X112+X121+X122+X131+X132+X141+X142=60;
X211+X212+X221+X222+X231+X232+X241+X242=27;

(1*X111)+(0.9*X211)<13.6;
(1*X112)+(0.9*X212)<13.6;
(1*X121)+(0.9*X221)<13.6;
(1*X122)+(0.9*X222)<13.6;
(1*X131)+(0.9*X231)<13.6;
(1*X132)+(0.9*X232)<13.6;
(1*X141)+(0.9*X241)<13.6;
(1*X142)+(0.9*X242)<13.6;

X111+X112+X211+X212<1000*Y1;
X121+X122+X221+X222<1000*Y2;
X131+X132+X231+X232<1000*Y3;
X141+X142+X241+X242<1000*Y4;

@GIN(X111);
@GIN(X112);
@GIN(X121);
@GIN(X122);
@GIN(X131);
@GIN(X132);
@GIN(X141);
@GIN(X142);
@GIN(X211);
@GIN(X212);
@GIN(X221);
@GIN(X222);
@GIN(X231);
@GIN(X232);
@GIN(X241);
@GIN(X242);

@BIN(Y1);
@BIN(Y2);
@BIN(Y3);
@BIN(Y4);

END
```

Şekil 4.1. İkinci veri grubuna ait matematiksel modelin LINGO ekran görüntüsü

```
Feasible solution found.
Infeasibilities:                           0.000000
Extended solver steps:                            0
Total solver iterations:                         22
Elapsed runtime seconds:                       0.06

Model Class:                                   PILP

Total variables:                 20
Nonlinear variables:              0
Integer variables:               20

Total constraints:               14
Nonlinear constraints:            0

Total nonzeros:                  52
Nonlinear nonzeros:               0


                                     Variable           Value
                                         X111        13.00000
                                         X112        8.000000
                                         X121        0.000000
                                         X122        0.000000
                                         X131        0.000000
                                         X132        13.00000
                                         X141        13.00000
                                         X142        13.00000
                                         X211        0.000000
                                         X212        2.000000
                                         X221        15.00000
                                         X222        0.000000
                                         X231        10.00000
                                         X232        0.000000
                                         X241        0.000000
                                         X242        0.000000
                                           Y1        1.000000
                                           Y2        1.000000
                                           Y3        1.000000
                                           Y4        1.000000
```

Şekil 4.2 İkinci modelin LINGO çözüm raporu

Bu çözüme göre birinci vardiyada çalışacak tıra 13 karton kule, 15 metal kule yerleştirilmelidir. Karton kulelerin tamamı tırın (s=1) birinci katmanına metal kulelerin tamamı da tırın (s=2) ikinci katmanına yerleştirilmelidir. İkinci vardiyada çalışacak tırda ise toplam 20 adet karton kule ve 7 adet metal kule taşınmalı. Karton kulelerin 8 adeti (s=1) birinci katmana, 12 adedi ise (s=2) ikinci katmana yerleştirilmelidir. Metal kulelerin ise 6 adedi (s=1) birinci katmana, 1 adedi ise (s=2) ikinci katmana yerleştirilmelidir. Üçüncü vardiyada çalışacak tıra ise toplam 23 adet karton kule ve toplam 3 adet metal kule yerleştirilmelidir.

Karton kulelerin 12 adedi (s=1) birinci katmanda, 11 adedi ise (s=2) ikinci katmanda taşınmalıdır. Metal kulelerin yerleşimi (s=1) birinci katmana 1 adet, (s=2) ikinci katmana 2 adet olmalıdır.

Bu şekilde üç vardiyada toplam 56 adet karton kule ve 25 adet metal kule taşınmaktadır. Ancak günlük talep 60 adet karton kule ve 27 adet metal kule olduğundan; dördüncü tıra ihtiyaç duyulacaktır. Y_4 karar değişkeninin 1 değeri alması bu durumu ifade etmektedir. Kalan 4 karton kule ve 2 metal kule dördüncü tır ile taşınmalıdır. Bir tır, bir günde üç tur taşıma yaptığından firma günlük talebi karşılayabilmek için yeni bir tıra daha ihtiyaç duyacaktır. Ancak bu yeni tır için yaklaşık 120 000 € gibi bir yatırım yapıp satın almak yerine, gündüz 08:00-16:00 vardiyasında çalışacak bir araç kiralanarak talep çok daha düşük bir maliyetle karşılanabilir.

5. SONUÇ

Bu yüksek lisans tez çalışması kapsamında otomotiv sektöründe faaliyet gösteren bir firmanın yarı mamül ve malzeme akışı güncel veriler için optimize edilmiştir. Yatırım kararlarında referans alınması amacıyla aynı model, farklı veri grupları ile de çalışılmıştır. Örnek teşkil etmesi açısından da problem global karınca kolonisi algoritması adı verilen yeni bir yaklaşımla, bir tır için çözülmüş ve elde edilen sonuçlar karşılaştırılmıştır.

Çalışmanın ilk bölümünde lojistik, lojistik yönetimi, konteynır yükleme problemi, tamsayılı programlama, karınca kolonisi optimizasyonu algoritması gibi temel kavramlara değinilmiştir. İzleyen bölümde optimizasyon probleminin çözümü için önerilen matematiksel model ve meta sezgisel yöntem tanıtılmıştır. Son bölümde ise farklı veri grupları için elde edilen çözümler sunulmuştur.

Bu çalışmada farklı lokasyonlarda bulunan iki nokta arasındaki malzeme akışının optimizasyonu problemi bir “hacim maksimizasyonu” problemi olarak ele alınmıştır. Amaç fonksiyonu; her vardiyada çalışan tırı maksimum hacimle doldurmak, böylece taşıma yapacak tır sayısını en aza indirmektir. Elde edilen çözüm raporları tır içi yerleşim planı niteliğindedir. Üst üste istiflenen kutuların özdeş olması sebebiyle tır içi yerleşimin sadece üstten görünüşü verilmekte yetinilmiş, yandan veya önden görünüşün görselleştirilmesine ihtiyaç duyulmamıştır.

Kutu tiplerinin eşleştirilmesi kısıtını sağlamak ve yerleştirilecek kutu sayısını göreli azaltmak amacıyla “kutu” lardan “kule” lere geçilerek veri kümesi sadeleştirilmiştir. Yine yerleştirilecek kutu tiplerinin en uzunluklarının eşit ve tırın en uzunluğunun yarısı kadar olması sebebiyle tırın içi dikey olarak iki katmana ayrılmıştır. Her bir katmanın boyu ve yüksekliği tırın boyuna ve yüksekliğine eşit ancak eni tırın eninin yarısı kadardır.

Hazırlanan matematiksel model LINGO programının demo versiyonunda, 1,8 GHz Intel Core i5 işlemci ile çözülmüş ve elde edilen çözüm raporlarının ekran görüntüsü metne eklenmiştir. Kullanılan meta sezgisel yöntem içinse Python 3.7.3 programlama dili ve yine 1,8 GHz Intel Core i5 işlemci kullanılmıştır. Beklendiği gibi matematiksel model meta sezgisel yönteme göre tır bazında daha yüksek doluluk oranı sağlamıştır.

Bu çalışma sonucunda günlük tüketim talebini karşılayabilecek, ambalaj istifleme kurallarına uygun, minimum tır sayısı ile taşıma yapılabilecek tır içi yerleşim planları firmaya sunulmuştur. Mevcut durumda, bir tır bu akış için yeterli olmaktadır. Ancak 2021 yılı talep öngörüleri için tek vardiya çalışacak ikinci tıra ihtiyaç duyulacaktır. İkinci tıra taşan hacmin 96 metreküp yani bir tır hacminden oldukça küçük olması sebebiyle, artan talebin yeni bir tır yatırımı yapılmadan, tek vardiya çalışacak bir araç kiralanarak karşılanması tavsiye edilmektedir.

KAYNAKLAR

Alaykıran, K., Engin, O., 2004. Karınca kolonileri meta sezgiseli ve gezgin satıcı problemleri üzerinde bir uygulama. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi,*20(1): 69-76.

Araújo, E.J., Chaves, A.A., Salles Neto, L.L., Azevedo A.T., 2016. Pareto clustering search applied for 3D container ship loading plan problem. *Expert Systems with Applications*, 44, 50-57.

Araya, I., Ri , M.C., 2014. A beam search approach to the container loading problem. *Computers & Operations Research,* 43, 100-107.

Bakır, M.A., Altunkaynak, B., 2003. Tamsayılı Programlama Teori, Modeller ve Algoritmalar. Nobel Yayıncılık, Türkiye, 630s.

Bakkal, M., Çelik, U. 2011. Lojistik Yönetimi ve e-Lojistik. Hiperlink Yayınları, İstanbul, 59s.

Bakkal M., Oflaz, A. 2011. Lojistik Bilgi Sistemleri. Hiperlink Yayınları, İstanbul, 71s.

Demirel, N., Hadi, G., Akçayol, M.A., Demirel, E. 2011. Çok aşamalı bütünleşik lojistik ağı optimizasyonu probleminin melez genetik algoritma ile çözümü. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi,* 26(4): 929-936.

Dereli, T., Daş, G.S., 2010. Konteyner yükleme problemleri için karınca kolonisi optimizasyonu yaklaşımı. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi,* 25(4): 881-894.

Eröz, E., Tanyıldızı, E., 2018. Güncel Metasezgisel Optimizasyon Algoritmalarının Performans karşılaştırılması. http://www.researchgate.net (Erişim tarihi:14.06.2019).

Faina, L., 2000. A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research,* 126, 340-354.

Kalkancı, Ç. 2009. Coğrafi bilgi sistemleri destekli üretim ve lojistik optimizasyonu ve asfalt sektöründe bir uygulama. *Yüksek Lisans Tezi,* İÜ Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, İstanbul.

Karaoğlu, Ö. 2017. Akümülatör geri kazanım ağı için bir tersine lojistik optimizasyonu. *Yüksek Lisans Tezi,* SÜ Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Konya.

Keskintürk, T., Söyler, H., 2006. Global karınca kolonisi optimizasyonu. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi,* 21(4): 689-698.

Liang S.C., Lee, C.Y., Huang S.W.,2007. Hybrid Meta-Heuristic for the Container Loading Problem. *Communications of the IIMA*, 7/4, 73-84.

Lim, A., Ma, H., Qiu, C., Zhu, W., 2013. The single container loading problem with axle weight constraints. *International Journal of Production Economics*, 144, 358-369

Palamutçuoğlu, T. 2012. Lojistik Yönetimi. Celal Bayar Üniversitesi, Kula Meslek Yüksekokulu, Ders Notları, Manisa.

Socha, K. Dorigo, M., 2008. Ant colony optimization for continuos domains. *European Journal of Operational Research,* 185, 1155-1179.

Tekil, S., Özkır, V.Ç., 2016. Konteyner yükleme problemlerinin incelenmesi ve lojistik sektöründe bir uygulama. *Toros Üniversitesi İİSBF* Sosyal Bilimler Dergisi 3(5): 213-224.

Yurtay, Y., Yurtay, N., Akçetin, E., Kılıç, A. 2014. Konteynerde yük optimizasyonu: örnek uygulama. *Yönetim ve Ekonomi Araştırmaları Dergisi,* (23): 228-247.
Wei, L., Zhu, W., Lim, A., 2015. A goal-driven prototype column generation strategy for the multiple container loading cost minimization problems. *European Journal of Operational Research,* 241, 39-49.

EKLER

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu

EK 2 Önerilen Karınca Koloni Algoritması Çözüm Raporu

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu

```
#veri kümelerini tanımlama
düğümler=set(["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","R",
              "S","T","U","V","Y","Z","X","Q"])
karıncalar=set(["N1","N2","N3","N4","N5","N6","N7","N8","N9","N10","N11","N12",
              "N13","N14","N15","N16","N17","N18","N19","N20","N21","N22",
              "N23","N24","N25"])
kartonlar=set(["K1","K2","K3","K4","K5","K6","K7","K8","K9"])
metaller=set(["M1","M2","M3","M4","M5","M6","M7","M8","M9","M10","M11",
            "M12","M13","M14","M15","M16"])
kuleler = kartonlar.union(metaller)

#verileri ekrana yazdırma
print("düğümler   :",(düğümler))
print("karıncalar :",(karıncalar))
print("kuleler    :",(kuleler))

#hacim bilgileri m3 cinsinden
karton_hacim = 3.24
metal_hacim = 3.024
tır_hacim = 94

#başlangıç için her düğüme bir karınca yerleştir
düğüm_karınca = zip(düğümler,karıncalar)
print("düğüm_karınca :",(*zip(düğümler,karıncalar)))

#başlangıç için her düğüme bir kule yerleştir
düğüm_kule = zip(düğümler,kuleler)
print("düğüm_kule :",(*zip(düğümler,kuleler)))
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
#her karınca için bir rota oluştur
rota = []
for k in karıncalar:
   rota += [düğümler]
print("rota :",rota)

#rotadaki düğümlerin kule karşılığı
for r in rota:
   rota[0:len(rota)] = kuleler
print("eşleşme :",rota)

#rotadaki kulelerin hacim karşılığı
atama = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]
for i in atama:
   if rota[0]=="K1" or rota[0]=="K2" or rota[0]=="K3" or rota[0] =="K4" or rota[0]
                  =="K5" or rota[0] =="K6" or rota[0]=="K7" or rota[0]=="K8" or
                  rota[0]=="K9":
      atama[0] = 3.24
   elif rota[0] == "M1" or rota[0] == "M2" or rota[0] == "M3" or rota[0] == "M4" or
                  rota[0] == "M5" or rota[0] == "M6" or rota[0] == "M7" or rota[0] ==
                  "M8" or rota[0] == "M9" or rota[0]=="M10" or rota[0]=="M11" or
                  rota[0]=="M12" or rota[0]=="M13" or rota[0]=="M14" or
                  rota[0]=="M15" or rota[0]=="M16":
      atama[0] = 3.024

   if rota[1]=="K1" or rota[1]=="K2" or rota[1]=="K3" or rota[1] =="K4" or rota[1]
                  =="K5" or rota[1] =="K6" or rota[1]=="K7" or rota[1]=="K8" or
                  rota[1]=="K9":
      atama[1] = 3.24
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
elif rota[1] == "M1" or rota[1] == "M2" or rota[1] == "M3" or rota[1] == "M4" or
        rota[1] == "M5" or rota[1] == "M6" or rota[1] == "M7" or rota[1] ==
        "M8" or rota[1] == "M9" or rota[1]=="M10" or rota[1]=="M11" or
        rota[1]=="M12" or rota[1]=="M13" or rota[1]=="M14" or
        rota[1]=="M15" or rota[1]=="M16":
   atama[1] = 3.024

if rota[2]=="K1" or rota[2]=="K2" or rota[2]=="K3" or rota[2] =="K4" or rota[2]
        =="K5" or rota[2] =="K6" or rota[2]=="K7" or rota[2]=="K8" or
        rota[2]=="K9":
   atama[2] = 3.24
elif rota[2] == "M1" or rota[2] == "M2" or rota[2] == "M3" or rota[2] == "M4" or
        rota[2] == "M5" or rota[2] == "M6" or rota[2] == "M7" or rota[2] ==
        "M8" or rota[2] == "M9" or rota[2]=="M10" or rota[2]=="M11" or
        rota[2]=="M12" or rota[2]=="M13" or rota[2]=="M14" or
        rota[2]=="M15" or rota[2]=="M16":
   atama[2] = 3.024

if rota[3]=="K1" or rota[3]=="K2" or rota[3]=="K3" or rota[3] =="K4" or rota[3]
        =="K5" or rota[3] =="K6" or rota[3]=="K7" or rota[3]=="K8" or
        rota[3]=="K9":
   atama[3] = 3.24
elif rota[3] == "M1" or rota[3] == "M2" or rota[3] == "M3" or rota[3] == "M4" or
        rota[3] == "M5" or rota[3] == "M6" or rota[3] == "M7" or rota[3] ==
        "M8" or rota[3] == "M9" or rota[3]=="M10" or rota[3]=="M11" or
        rota[3]=="M12" or rota[3]=="M13" or rota[3]=="M14" or
        rota[3]=="M15" or rota[3]=="M16":
   atama[3] = 3.024
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
if rota[4]=="K1" or rota[4]=="K2" or rota[4]=="K3" or rota[4] =="K4" or rota[4]
        =="K5" or rota[4] =="K6" or rota[4]=="K7" or rota[4]=="K8" or
        rota[4]=="K9":
   atama[4] = 3.24
elif rota[4] == "M1" or rota[4] == "M2" or rota[4] == "M3" or rota[4] == "M4" or
        rota[4] == "M5" or rota[4] == "M6" or rota[4] == "M7" or rota[4] ==
        "M8" or rota[4] == "M9" or rota[4]=="M10" or rota[4]=="M11" or
        rota[4]=="M12" or rota[4]=="M13" or rota[4]=="M14" or
        rota[4]=="M15" or rota[4]=="M16":
   atama[4] = 3.024

if rota[5]=="K1" or rota[5]=="K2" or rota[5]=="K3" or rota[5] =="K4" or rota[5]
        =="K5" or rota[5] =="K6" or rota[5]=="K7" or rota[5]=="K8" or
        rota[5]=="K9":
   atama[5] = 3.24
elif rota[5] == "M1" or rota[5] == "M2" or rota[5] == "M3" or rota[5] == "M4" or
        rota[5] == "M5" or rota[5] == "M6" or rota[5] == "M7" or rota[5] ==
        "M8" or rota[5] == "M9" or rota[5]=="M10" or rota[5]=="M11" or
        rota[5]=="M12" or rota[5]=="M13" or rota[5]=="M14" or
        rota[5]=="M15" or rota[5]=="M16":
   atama[5] = 3.024

if rota[6]=="K1" or rota[6]=="K2" or rota[6]=="K3" or rota[6] =="K4" or rota[6]
        =="K5" or rota[6] =="K6" or rota[6]=="K7" or rota[6]=="K8" or
        rota[6]=="K9":
   atama[6] = 3.24
elif rota[6] == "M1" or rota[6] == "M2" or rota[6] == "M3" or rota[6] == "M4" or
        rota[6] == "M5" or rota[6] == "M6" or rota[6] == "M7" or rota[6] ==
        "M8" or rota[6] == "M9" or rota[6]=="M10" or rota[6]=="M11" or
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
            rota[6]=="M12" or rota[6]=="M13" or rota[6]=="M14" or
            rota[6]=="M15" or rota[6]=="M16":
    atama[6] = 3.024

if rota[7]=="K1" or rota[7]=="K2" or rota[7]=="K3" or rota[7] =="K4" or rota[7]
            =="K5" or rota[7] =="K6" or rota[7]=="K7" or rota[7]=="K8" or
            rota[7]=="K9":
    atama[7] = 3.24
elif rota[7] == "M1" or rota[7] == "M2" or rota[7] == "M3" or rota[7] == "M4" or
            rota[7] == "M5" or rota[7] == "M6" or rota[7] == "M7" or rota[7] ==
            "M8" or rota[7] == "M9" or rota[7]=="M10" or rota[7]=="M11" or
            rota[7]=="M12" or rota[7]=="M13" or rota[7]=="M14" or
            rota[7]=="M15" or rota[7]=="M16":
    atama[7] = 3.024

if rota[8]=="K1" or rota[8]=="K2" or rota[8]=="K3" or rota[8] =="K4" or rota[8]
            =="K5" or rota[8] =="K6" or rota[8]=="K7" or rota[8]=="K8" or
            rota[8]=="K9":
    atama[8] = 3.24
elif rota[8] == "M1" or rota[8] == "M2" or rota[8] == "M3" or rota[8] == "M4" or
            rota[8] == "M5" or rota[8] == "M6" or rota[8] == "M7" or rota[8] ==
            "M8" or rota[8] == "M9" or rota[8]=="M10" or rota[8]=="M11" or
            rota[8]=="M12" or rota[8]=="M13" or rota[8]=="M14" or
            rota[8]=="M15" or rota[8]=="M16":
    atama[8] = 3.024

if rota[9]=="K1" or rota[9]=="K2" or rota[9]=="K3" or rota[9] =="K4" or rota[9]
            =="K5" or rota[9] =="K6" or rota[9]=="K7" or rota[9]=="K8" or
            rota[9]=="K9":
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
    atama[9] = 3.24
elif rota[9] == "M1" or rota[9] == "M2" or rota[9] == "M3" or rota[9] == "M4" or
            rota[9] == "M5" or rota[9] == "M6" or rota[9] == "M7" or rota[9] ==
            "M8" or rota[9] == "M9" or rota[9]=="M10" or rota[9]=="M11" or
            rota[9]=="M12" or rota[9]=="M13" or rota[9]=="M14" or
            rota[9]=="M15" or rota[9]=="M16":
    atama[9] = 3.024

if rota[10]=="K1" or rota[10]=="K2" or rota[10]=="K3" or rota[10] =="K4" or
            rota[10] =="K5" or rota[10] =="K6" or rota[10]=="K7" or
            rota[10]=="K8" or rota[10]=="K9":
    atama[10] = 3.24
elif rota[10] == "M1" or rota[10] == "M2" or rota[10] == "M3" or rota[10] == "M4"
            or rota[10] == "M5" or rota[10] == "M6" or rota[10] == "M7" or rota[10]
            == "M8" or rota[10] == "M9" or rota[10]=="M10" or rota[10]=="M11"
            or rota[10]=="M12" or rota[10]=="M13" or rota[10]=="M14" or
            rota[10]=="M15" or rota[10]=="M16":
    atama[10] = 3.024

if rota[11]=="K1" or rota[11]=="K2" or rota[11]=="K3" or rota[11] =="K4" or
            rota[11] =="K5" or rota[11] =="K6" or rota[11]=="K7" or
            rota[11]=="K8" or rota[11]=="K9":
    atama[11] = 3.24
elif rota[11] == "M1" or rota[11] == "M2" or rota[11] == "M3" or rota[11] == "M4"
            or rota[11] == "M5" or rota[11] == "M6" or rota[11] == "M7" or rota[11]
            == "M8" or rota[11] == "M9" or rota[11]=="M10" or rota[11]=="M11"
            or rota[11]=="M12" or rota[11]=="M13" or rota[11]=="M14" or
            rota[11]=="M15" or rota[11]=="M16":
    atama[11] = 3.024
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
if rota[12]=="K1" or rota[12]=="K2" or rota[12]=="K3" or rota[12] =="K4" or
        rota[12] =="K5" or rota[12] =="K6" or rota[12]=="K7" or
        rota[12]=="K8" or rota[12]=="K9":
   atama[12] = 3.24
elif rota[12] == "M1" or rota[12] == "M2" or rota[12] == "M3" or rota[12] == "M4"
        or rota[12] == "M5" or rota[12] == "M6" or rota[12] == "M7" or rota[12]
        == "M8" or rota[12] == "M9" or rota[12]=="M10" or rota[12]=="M11"
        or rota[12]=="M12" or rota[12]=="M13" or rota[12]=="M14" or
        rota[12]=="M15" or rota[12]=="M16":
   atama[12] = 3.024

if rota[13]=="K1" or rota[13]=="K2" or rota[13]=="K3" or rota[13] =="K4" or
        rota[13] =="K5" or rota[13] =="K6" or rota[13]=="K7" or
        rota[13]=="K8" or rota[13]=="K9":
   atama[13] = 3.24
elif rota[13] == "M1" or rota[13] == "M2" or rota[13] == "M3" or rota[13] == "M4"
        or rota[13] == "M5" or rota[13] == "M6" or rota[13] == "M7" or rota[13]
        == "M8" or rota[13] == "M9" or rota[13]=="M10" or rota[13]=="M11"
        or rota[13]=="M12" or rota[13]=="M13" or rota[13]=="M14" or
        rota[13]=="M15" or rota[13]=="M16":
   atama[13] = 3.024

if rota[14]=="K1" or rota[14]=="K2" or rota[14]=="K3" or rota[14] =="K4" or
        rota[14] =="K5" or rota[14] =="K6" or rota[14]=="K7" or
        rota[14]=="K8" or rota[14]=="K9":
   atama[14] = 3.24
elif rota[14] == "M1" or rota[14] == "M2" or rota[14] == "M3" or rota[14] == "M4"
        or rota[14] == "M5" or rota[14] == "M6" or rota[14] == "M7" or rota[14]
        == "M8" or rota[14] == "M9" or rota[14]=="M10" or rota[14]=="M11"
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
                or rota[14]=="M12" or rota[14]=="M13" or rota[14]=="M14" or
                rota[14]=="M15" or rota[14]=="M16":
    atama[14] = 3.024

if rota[15]=="K1" or rota[15]=="K2" or rota[15]=="K3" or rota[15] =="K4" or
                rota[15] =="K5" or rota[15] =="K6" or rota[15]=="K7" or
                rota[15]=="K8" or rota[15]=="K9":
    atama[15] = 3.24
elif rota[15] == "M1" or rota[15] == "M2" or rota[15] == "M3" or rota[15] == "M4"
                or rota[15] == "M5" or rota[15] == "M6" or rota[15] == "M7" or rota[15]
                == "M8" or rota[15] == "M9" or rota[15]=="M10" or rota[15]=="M11"
                or rota[15]=="M12" or rota[15]=="M13" or rota[15]=="M14" or
                rota[15]=="M15" or rota[15]=="M16":
    atama[15] = 3.024

if rota[16]=="K1" or rota[16]=="K2" or rota[16]=="K3" or rota[16] =="K4" or
                rota[16] =="K5" or rota[16] =="K6" or rota[16]=="K7" or
                rota[16]=="K8" or rota[16]=="K9":
    atama[16] = 3.24
elif rota[16] == "M1" or rota[16] == "M2" or rota[16] == "M3" or rota[16] == "M4"
                or rota[16] == "M5" or rota[16] == "M6" or rota[16] == "M7" or rota[16]
                == "M8" or rota[16] == "M9" or rota[16]=="M10" or rota[16]=="M11"
                or rota[16]=="M12" or rota[16]=="M13" or rota[16]=="M14" or
                rota[16]=="M15" or rota[16]=="M16":
    atama[16] = 3.024

if rota[17]=="K1" or rota[17]=="K2" or rota[17]=="K3" or rota[17] =="K4" or
                rota[17] =="K5" or rota[17] =="K6" or rota[17]=="K7" or
                rota[17]=="K8" or rota[17]=="K9":
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
    atama[17] = 3.24
elif rota[17] == "M1" or rota[17] == "M2" or rota[17] == "M3" or rota[17] == "M4"
                or rota[17] == "M5" or rota[17] == "M6" or rota[17] == "M7" or rota[17]
                == "M8" or rota[17] == "M9" or rota[17]=="M10" or rota[17]=="M11"
                or rota[17]=="M12" or rota[17]=="M13" or rota[17]=="M14" or
                rota[17]=="M15" or rota[17]=="M16":
    atama[17] = 3.024

if rota[18]=="K1" or rota[18]=="K2" or rota[18]=="K3" or rota[18] =="K4" or
                rota[18] =="K5" or rota[18] =="K6" or rota[18]=="K7" or
                rota[18]=="K8" or rota[18]=="K9":
    atama[18] = 3.24
elif rota[18] == "M1" or rota[18] == "M2" or rota[18] == "M3" or rota[18] == "M4"
                or rota[18] == "M5" or rota[18] == "M6" or rota[18] == "M7" or rota[18]
                == "M8" or rota[18] == "M9" or rota[18]=="M10" or rota[18]=="M11"
                or rota[18]=="M12" or rota[18]=="M13" or rota[18]=="M14" or
                rota[18]=="M15" or rota[18]=="M16":
    atama[18] = 3.024

if rota[19]=="K1" or rota[19]=="K2" or rota[19]=="K3" or rota[19] =="K4" or
                rota[19] =="K5" or rota[19] =="K6" or rota[19]=="K7" or
                rota[19]=="K8" or rota[19]=="K9":
    atama[19] = 3.24
elif rota[19] == "M1" or rota[19] == "M2" or rota[19] == "M3" or rota[19] == "M4"
                or rota[19] == "M5" or rota[19] == "M6" or rota[19] == "M7" or rota[19]
                == "M8" or rota[19] == "M9" or rota[19]=="M10" or rota[19]=="M11"
                or rota[19]=="M12" or rota[19]=="M13" or rota[19]=="M14" or
                rota[19]=="M15" or rota[19]=="M16":
    atama[19] = 3.024
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
if rota[20]=="K1" or rota[20]=="K2" or rota[20]=="K3" or rota[20] =="K4" or
        rota[20] =="K5" or rota[20] =="K6" or rota[20]=="K7" or
        rota[20]=="K8" or rota[20]=="K9":
   atama[20] = 3.24
elif rota[20] == "M1" or rota[20] == "M2" or rota[20] == "M3" or rota[20] == "M4"
        or rota[20] == "M5" or rota[20] == "M6" or rota[20] == "M7" or rota[20]
        == "M8" or rota[20] == "M9" or rota[20]=="M10" or rota[20]=="M11"
        or rota[20]=="M12" or rota[20]=="M13" or rota[20]=="M14" or
        rota[20]=="M15" or rota[20]=="M16":
   atama[20] = 3.024

if rota[21]=="K1" or rota[21]=="K2" or rota[21]=="K3" or rota[21] =="K4" or
        rota[21] =="K5" or rota[21] =="K6" or rota[21]=="K7" or
        rota[21]=="K8" or rota[21]=="K9":
   atama[21] = 3.24
elif rota[21] == "M1" or rota[21] == "M2" or rota[21] == "M3" or rota[21] == "M4"
        or rota[21] == "M5" or rota[21] == "M6" or rota[21] == "M7" or rota[21]
        == "M8" or rota[21] == "M9" or rota[21]=="M10" or rota[21]=="M11"
        or rota[21]=="M12" or rota[21]=="M13" or rota[21]=="M14" or
        rota[21]=="M15" or rota[21]=="M16":
   atama[21] = 3.024

if rota[22]=="K1" or rota[22]=="K2" or rota[22]=="K3" or rota[22] =="K4" or
        rota[22] =="K5" or rota[22] =="K6" or rota[22]=="K7" or
        rota[22]=="K8" or rota[22]=="K9":
   atama[22] = 3.24
elif rota[22] == "M1" or rota[22] == "M2" or rota[22] == "M3" or rota[22] == "M4"
        or rota[22] == "M5" or rota[22] == "M6" or rota[22] == "M7" or rota[22]
        == "M8" or rota[22] == "M9" or rota[22]=="M10" or rota[22]=="M11"
```

EK 1 Önerilen Karınca Koloni Algoritması Python Kodu (devam)

```
                or rota[22]=="M12" or rota[22]=="M13" or rota[22]=="M14" or
                rota[22]=="M15" or rota[22]=="M16":
        atama[22] = 3.024

    if rota[23]=="K1" or rota[23]=="K2" or rota[23]=="K3" or rota[23] =="K4" or
                rota[23] =="K5" or rota[23] =="K6" or rota[23]=="K7" or
                rota[23]=="K8" or rota[23]=="K9":
        atama[23] = 3.24
    elif rota[23] == "M1" or rota[23] == "M2" or rota[23] == "M3" or rota[23] == "M4"
                or rota[23] == "M5" or rota[23] == "M6" or rota[23] == "M7" or rota[23]
                == "M8" or rota[23] == "M9" or rota[23]=="M10" or rota[23]=="M11"
                or rota[23]=="M12" or rota[23]=="M13" or rota[23]=="M14" or
                rota[23]=="M15" or rota[23]=="M16":
        atama[23] = 3.024

    if rota[24]=="K1" or rota[24]=="K2" or rota[24]=="K3" or rota[24] =="K4" or
                rota[24] =="K5" or rota[24] =="K6" or rota[24]=="K7" or
                rota[24]=="K8" or rota[24]=="K9":
        atama[24] = 3.24
    elif rota[24] == "M1" or rota[24] == "M2" or rota[24] == "M3" or rota[24] == "M4"
                or rota[24] == "M5" or rota[24] == "M6" or rota[24] == "M7" or rota[24]
                == "M8" or rota[24] == "M9" or rota[24]=="M10" or rota[24]=="M11"
                or rota[24]=="M12" or rota[24]=="M13" or rota[24]=="M14" or
                rota[24]=="M15" or rota[24]=="M16":
        atama[24] = 3.024

print("atama :",atama)

print("toplam hacim :",sum(atama))
```

EK 2 Önerilen Karınca Koloni Algoritması Çözüm Raporu

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 16:52:21)

[Clang 6.0 (clang-600.0.57)] on darwin

Type "help", "copyright", "credits" or "license()" for more information.

>>>

================ RESTART: /Users/elifkurtulus/Desktop/tez2.py
================

düğümler : {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}

karıncalar : {'N11', 'N13', 'N6', 'N7', 'N23', 'N12', 'N15', 'N3', 'N14', 'N22', 'N16', 'N5', 'N9', 'N19', 'N2', 'N1', 'N8', 'N17', 'N21', 'N25', 'N18', 'N24', 'N10', 'N20', 'N4'}

kuleler : {'K9', 'M7', 'M15', 'K3', 'M1', 'M11', 'M2', 'M13', 'K4', 'K7', 'K6', 'M10', 'K5', 'M4', 'K8', 'M5', 'M9', 'K1', 'M3', 'M16', 'M6', 'K2', 'M8', 'M14', 'M12'}

düğüm_karınca : ('I', 'N11') ('E', 'N13') ('L', 'N6') ('T', 'N7') ('Q', 'N23') ('C', 'N12') ('X', 'N15') ('D', 'N3') ('M', 'N14') ('G', 'N22') ('B', 'N16') ('P', 'N5') ('V', 'N9') ('F', 'N19') ('Y', 'N2') ('A', 'N1') ('U', 'N8') ('Z', 'N17') ('H', 'N21') ('R', 'N25') ('J', 'N18') ('N', 'N24') ('O', 'N10') ('S', 'N20') ('K', 'N4')

düğüm_kule : ('I', 'K9') ('E', 'M7') ('L', 'M15') ('T', 'K3') ('Q', 'M1') ('C', 'M11') ('X', 'M2') ('D', 'M13') ('M', 'K4') ('G', 'K7') ('B', 'K6') ('P', 'M10') ('V', 'K5') ('F', 'M4') ('Y', 'K8') ('A', 'M5') ('U', 'M9') ('Z', 'K1') ('H', 'M3') ('R', 'M16') ('J', 'M6') ('N', 'K2') ('O', 'M8') ('S', 'M14') ('K', 'M12')

rota : [{'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I',

EK 2 Önerilen Karınca Koloni Algoritması Çözüm Raporu (devam)

'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}, {'I', 'E', 'L', 'T', 'Q', 'C', 'X', 'D', 'M', 'G', 'B', 'P', 'V', 'F', 'Y', 'A', 'U', 'Z', 'H', 'R', 'J', 'N', 'O', 'S', 'K'}]

eşleşme : ['K9', 'M7', 'M15', 'K3', 'M1', 'M11', 'M2', 'M13', 'K4', 'K7', 'K6', 'M10', 'K5', 'M4', 'K8', 'M5', 'M9', 'K1', 'M3', 'M16', 'M6', 'K2', 'M8', 'M14', 'M12']

atama : [3.24, 3.024, 3.024, 3.24, 3.024, 3.024, 3.024, 3.024, 3.24, 3.24, 3.24, 3.024, 3.24, 3.024, 3.24, 3.024, 3.024, 3.24, 3.024, 3.024, 3.024, 3.24, 3.024, 3.024, 3.024]

toplam hacim : 77.54400000000001

>>>

ÖZGEÇMİŞ

Adı Soyadı	: Elif KURTULUŞ
Doğum Yeri ve Tarihi	: Bursa 25.06.1985
Yabancı Dil	: İngilizce, Fransızca
Eğitim Durumu	
Lise	: Bursa Gazi Anadolu Lisesi - 2003
Lisans	: Eskişehir Osmangazi Üniversitesi Endüstri Mühendisliği Bölümü - 2007 Eskişehir Osmangazi Üniversitesi Makine Mühendisliği Bölümü - 2007 (ÇAP)
Yüksek Lisans	: Uludağ Üniversitesi Endüstri Mühendisliği Bölümü
Çalıştığı Kurum/Kurumlar	: Yazaki Otomotiv Bursa - 2007 - 2009 Uludağ Üniversitesi - 2009 - 2011 Oyak Renault Otomobil Fabrikaları - 2011 - 2016
İletişim (e-posta)	: elif.kurtulus@outlook.com
Yayınları	: -