T.C.
İSTANBUL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

DEPO YERLEŞİM DÜZENLEMESİ PROBLEMİNİN METASEZGİSELLERLE ÇÖZÜMÜ

Emre ÇAKMAK

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Danışman

Doç. Dr. Ş. Alp BARAY

Haziran, 2014

İSTANBUL

Bu çalışma 23 / 06 / 2014 tarihinde aşağıdaki jüri tarafından Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği programında Doktora olarak kabul edilmiştir.

Tez Jürisi:

İmza
Doç. Dr. Ş. Alp BARAY (Danışman)
İstanbul Üniversitesi
Mühendislik Fakültesi

İmza
Prof. Dr. Şakir ESNAF
İstanbul Üniversitesi
Mühendislik Fakültesi

İmza
Prof. Dr. Yaman ÖZTEK
Galatasaray Üniversitesi
İktisadi ve İdari Bilimler Fakültesi

İmza
Prof. Dr. Necdet ÖZÇAKAR
İstanbul Üniversitesi
İşletme Fakültesi

İmza
Doç. Dr. Mehmet Mutlu YENİSEY
İstanbul Teknik Üniversitesi
İşletme Fakültesi

ÖNSÖZ

Doktora tez çalışmamın hazırlanması sırasında yardım ve önerilerini esirgemeyen ve her konuda yol gösterici olan değerli tez danışmanım Doç. Dr. Ş. Alp BARAY'a sonsuz teşekkürlerimi sunarım. Tez çalışmamı yönlendirmemde yapıcı eleştirileri ile değerli katkılar sağlayan Prof. Dr. Şakir ESNAF'a teşekkürü bir borç bilirim. Tez çalışmam boyunca değerli önerileri ve katkıları için Prof. Dr. Yaman ÖZTEK'e çok teşekkür ederim. Ayrıca, değerli önerileri için Prof. Dr. Necdet ÖZÇAKAR ve Doç. Dr. M. Mutlu YENİSEY'e teşekkür ederim.

İlgi ve destekleri ile her zaman yanımda olan aileme teşekkürlerimi sunarım.

Varlığından güç aldığım, bu zorlu süreçte beni yorulmadan destekleyen sevgili eşim Pınar ÇAKMAK'a çok teşekkür ederim.

Haziran, 2014 Emre ÇAKMAK

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Simgeler	Açıklama
K	: Toplam gerekli olan stok pozisyonu sayısı
PS	: Yıllık depodan çıkan palet sayısı
C_h	: Elleçleme maliyeti
d	: Deponun malzeme girişini/çıkışını sağlayan kapının sayısı
gs	: Ürün grupları sayısı
N_i	: i malzeme grubu için ayrılan stok pozisyonu sayısı
P_i	: i ürün grubu için talepte bulunma olasılığı
L_{Xmax}	: x-ekseni boyunca maksimum uzunluk
L_{Ymax}	: y-ekseni boyunca maksimum uzunluk
L_{Zmax}	: z-ekseni boyunca maksimum uzunluk
T_x	: x-ekseni boyunca katedilen ortalama toplayıcı mesafesi
T_y	: y-ekseni boyunca katedilen ortalama toplayıcı mesafesi
T_z	: z-ekseni boyunca katedilen ortalama toplayıcı mesafesi
a_x	: x-ekseni boyunca palet için ayrılan mesafe
a_y	: y-ekseni boyunca palet için ayrılan mesafe
a_z	: z-ekseni boyunca palet için ayrılan mesafe
a_{zi}	: i ürün grubu için z-ekseni boyunca palet için ayrılan mesafe
w_x	: Ara koridor genişliği
w_y	: Ana koridor genişliği
n_X	: x-ekseni boyunca raf sayısı
n_Y	: y-ekseni boyunca raf sayısı
n_Z	: z-ekseni boyunca raf sayısı
n_{Xmax}	: x-ekseni boyunca maksimum raf sayısı
$n_{Yi\text{-}max}$	: i ürün grubu için y-ekseni boyunca maksimum raf sayısı
$n_{Zi\text{-}max}$	: i ürün grubu için z-ekseni boyunca maksimum raf sayısı
n_{Xmin}	: x-ekseni boyunca minimum raf sayısı
$n_{Yi\text{-}min}$	: i ürün grubu için y-ekseni boyunca minimum raf sayısı
$n_{Zi\text{-}min}$	: i ürün grubu için z-ekseni boyunca minimum raf sayısı
n_{Xi}	: i ürün grubu için x-ekseni boyunca ayrılan stok pozisyonu sayısı
n_{Yi}	: i ürün grubu için y-ekseni boyunca ayrılan stok pozisyonu sayısı
n_{Zi}	: i ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı
D	: Parçacıkların boyut sayısı
v_{min}	: Hız vektörünün minimum değeri
v_{max}	: Hız vektörünün maksimum değeri
$v_{i,d}^{k}$	: i. bireyin d. boyutu k. iterasyon hızı
$x_{i,d}^{k}$	: i. bireyin d. boyutunun k. iterasyondaki konumu
$p_{i,d}^{k}$	: i. bireyin d. boyutunun k. iterasyondaki bireysel en iyi değeri
g_{d}^{k}	: d. boyutun k. iterasyondaki global en iyi değeri
$c_1 ve\ c_2$	: Bilişsel ve sosyal parametreleri

r_1 ve r_2 : [0,1] aralığında uniform dağılmış rastgele sayılar
w_k : k. iterasyondaki atalet ağırlığı
$iter_{max}$: maksimum iterasyon sayısı

Kısaltmalar	Açıklama
GA	: Genetik Algoritma
KK	: Karınca Koloni
PSO	: Parçacık Sürü Optimizasyonu
TA	: Tabu Arama
TB	: Tavlama Benzetimi

ÖZET

DOKTORA TEZİ

DEPO YERLEŞİM DÜZENLEMESİ PROBLEMİNİN METASEZGİSELLERLE ÇÖZÜMÜ

Emre ÇAKMAK

İstanbul Üniversitesi

Fen Bilimleri Enstitüsü

Endüstri Mühendisliği Anabilim Dalı

Danışman : Doç. Dr. Ş. Alp BARAY

Teknolojinin artan bir hızla gelişmesi ve rekabetin artması ile ortaya çıkan baskılar tedarik zincirinde yer alan tüm şirketler üzerinde hissedilmektedir. Söz konusu baskılar ile birlikte müşteri taleplerinin de hızlı değişimlere uğraması tedarik zincirinde yer alan lojistik operasyonların birbirleriyle entegre biçimde gerçekleştirilmesini gerektirmektedir. Bu bağlamda üzerinde durulması gereken en önemli unsurlardan birisi de depolardır. Müşteriler ile tedarikçiler arasında köprü rolü üstlenen depoların, görevlerini amacına uygun bir şekilde gerçekleştirebilmeleri depoların ne kadar iyi düzenlendiğine bağlıdır. Bu noktada depo yerleşim düzenlemesi önem kazanmaktadır.

Bu çalışmada, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemi modeli geliştirilmiştir. Geliştirilen modelin NP-zor yapısından dolayı, metasezgisel yöntemlerden biri olan parçacık sürü optimizasyon (PSO) algoritması kullanılmıştır. PSO algoritması için geliştirilmiş mevcut sınırlandırma yöntemlerine ek olarak iki yeni sınırlandırma yöntemi önerilmiştir. Ayrıca, çözüm uzayını daraltarak parçacıkların en uygun ve en iyi sonucu bulma olasılığını artırmayı sağlayan parçacıkların başlangıç minimum değerleri için yeni bir atama algoritması önerilmiştir. Geliştirilen algoritmalı PSO algoritması ve standart PSO algoritması farklı örnek problemler üzerinde uygulanarak, elde edilen sonuçlar karşılaştırılmıştır. Karşılaştırma sonucunda, en iyi sonucu veren yöntemlerin depo raf sistemi konfigürasyonu ve depo yerleşim boyutlarına ilişkin verdiği sonuçlar sunulmuştur.

Haziran 2014, 156 Sayfa.

Anahtar kelimeler: Depo yerleşim düzenlemesi, PSO, sınırlandırma yöntemleri.

SUMMARY

Ph.D. THESIS

SOLVING WAREHOUSE LAYOUT PROBLEM WITH METAHEURISTICS

Emre ÇAKMAK

İstanbul University

Graduate School of Science and Engineering

Department of Industrial Engineering

Supervisor : Assoc. Prof. Dr. Ş. Alp BARAY

Emerging pressures caused by the rapid development of technology and increase of competition, are perceived by all companies in the supply chain. Due to the rapid changes of customer demands along with these pressures, logistics operations within the supply chain need to be integrally performed. In this context, warehouses are one of the important elements that need to be focused on. Warehouses, taking the role of bridge between customers and suppliers, can only perform their duties accordingly their purposes, if only they organized well. At this point, warehouse layout becomes an important issue.

In this study, a warehouse layout problem model for pallets with variable height is developed. Due to the NP-hard nature of the developed model, one of the metaheuristics methods particle swarm optimization (PSO) algorithm is used. Two new boundary conditions are proposed in addition to the current boundary conditions developed for PSO algorithms. Besides, a new assignment algorithm for particles' initial minimum values is proposed which provides the particles increase the probability of finding the most appropriate and best results by narrowing the solution space. The PSO algorithm with proposed algorithm and standard PSO algorithm are applied on different problems, and the obtained results are compared. At the end of comparison, results obtained from the methods that give the best solutions in terms of warehouse rack configuration and warehouse layout dimensions are presented.

June 2014, 156 Pages.

Keywords: Warehouse layout, PSO, boundary conditions.

1. GİRİŞ

Teknolojinin, rekabetin ve taleplerin değişkenliğinin arttığı günümüz iş koşullarında, gelen siparişlerin olabildiğince hızlı ve hatasız olarak karşılanabilmesi için; söz konusu siparişlerin karşılanması sırasında önemli roller üstlenen lojistik operasyonların birbirleriyle harmoni içinde ve birbirlerini destekleyecek şekilde gerçekleştirilmesi gerekmektedir. Diğer taraftan, artan rekabet ve değişen talepler gelen siparişlerin karşılanması sırasında tedarik zincirindeki tüm elemanlar üzerinde de büyük baskılar yaratmaktadır. Günümüzde müşteri siparişleri içerisinde talep edilen malzemelerin miktarları azalmış, malzeme çeşitliliği ise yükselmiştir. Bunlara ek olarak, daha sık sipariş verilmekte ve siparişlerin istenilen zamanda teslim edilmesi beklenmektedir. Dolayısıyla sık aralıklı verilen, olabildiğince hızlı teslim edilmesi istenen ve çok çeşitlilik içeren siparişlerin karşılanması için; malzeme akışındaki tüm şirketlerin tedarik zincirini ve lojistik sistemini, bu akışı sekteye uğratmayacak bir düzende kurgulaması gerekmektedir. Tedarik zincirinin ve lojistik sisteminin kurgulanması aşamasında üzerinde durulması gereken en önemli unsurlardan birisi de depo ve depolama sürecidir.

Depolama sürecinin gerçekleştiği tesisler olan depolar, tedarik zincirinin ve lojistik sisteminin vazgeçilmez unsurlarından birisidir. Depolar, üreticiler ve tüketiciler arasında bir bağ oluştururlar. Bu bağın ne kadar kuvvetli olduğu ise, deponun ve depolama sürecinin başarısına bağlıdır. Başarının kalıcı olabilmesi için deponun, depolama sürecinde gerçekleştirilen depo operasyonlarına uygun şekilde düzenlenmesi gerekmektedir. Bu düzenlemenin ise depo tasarımı aşamasında ve tasarım aşamasının içerisinde yer alan depo yerleşim düzenlemesi aşamasında düşünülmesi gerekmektedir.

Depo yerleşim düzenlemesi, tasarlanacak olan deponun kullanım amacını ve bu amaca ulaşmak için gerekli olan depo fonksiyonlarını dikkate alarak; depo boyutlarını belirleme, raf sistemlerinin seçimi, rafların uzunluklarının, yüksekliklerinin ve derinliklerinin belirlenmesi, depolama politikasının seçimi, yerleşim planının hazırlanması, yükleme/boşaltma için kullanılacak yerlerin belirlenmesi ve sipariş

toplama yöntemlerinin tespiti gibi birçok kararı içeren bir tasarım sürecidir. Depo yerleşim düzenlemesi çok zor ve karmaşık bir süreçtir. Bu sürecin zor ve karmaşık olmasının temel nedeni ise yukarıda sayılmakta olan kararların süreç içerisinde birbiri ile çelişiyor olması ve sürecin birçok alternatife sahip olmasıdır. Yerleşim düzenlemesi sırasında verilmesi gereken her bir karar, diğer kararlar da göz önünde bulundurularak verilmelidir.

Literatürde depolama ile ilgili yer almakta olan çalışmalar genel olarak sipariş toplama operasyonu ve bu operasyon ile ilgili sipariş toplama süreci, sipariş toplama stratejisi ve sipariş birleştirme yöntemi gibi konulara odaklanmıştır. Depolamada gerçekleştirilen tüm operasyonlara büyük etkisi bulunan depo yerleşim düzenlemesi konusu üzerine ise fazla çalışma bulunmamaktadır. Bu bağlamda doktora tezi kapsamında yapılacak olan çalışma, farklı palet yüksekliğine sahip olan malzemeler için uygun bir depo yerleşim düzenlemesi üzerinedir. Çalışmaya konu olan depo yerleşim düzenlemesinde, stokta tutulacak malzemeler farklı palet yüksekliğine sahiptir. Bunun sonucu olarak aynı kolonda, aynı rafta veya raf bloğunda üst üste yerleştirilecek paletler farklı yükseklikte olduğundan, raflar arasında mesafe farklı olacak; rafların arasındaki mesafe değişecek ve doğal olarak aynı kolonda, aynı rafta veya raf bloğunda stok kapasitesi değişecektir.

"Depo Yerleşim Düzenlemesi Probleminin Metasezgisellerle Çözümü" başlıklı doktora tezinin amacı, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemine metasezgisel bir yöntem ile çözüm bulmaktır. NP-zor niteliğe sahip bir problem olan depo yerleşim düzenlemesi problemi, doktora tezi kapsamında geliştirilmiş ve değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemi haline getirilmiştir. Doktora tezinin amacına ulaşmada, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi probleminin çözümünde kullanılan metasezgisel yöntem, sürü davranışı sergileyen parçacık sürü optimizasyon (PSO) algoritmasıdır. Ancak, PSO algoritması ile çözülecek söz konusu problemin yapısında kısıtlar bulunduğundan; PSO algoritması tek başına yetersiz kalmaktadır. Bu yetersizliğin üstesinden gelebilmek amacıyla PSO algoritması için geliştirilmiş mevcut sınırlandırma yöntemleri kullanılacaktır. Ayrıca, bu çalışmada var olan sınırlandırma yöntemlerine ek olarak iki yeni sınırlandırma yöntemi olan *"Görünür Yansıtma"* ve *"Görünür Sönümleme"* sınırlandırma yöntemleri de önerilecektir. PSO algoritmasının

bir diğer yetersizliği ise, parçacıklar çözüm uzayı büyüdüğünde çözüme ulaşmak için uygun olmayan çözümlerde çok zaman harcamakta ve bazı durumlarda ise en iyi çözüme ulaşmadan yerel uygunluklara takılmaktadır. Bunun üstesinden gelebilmek amacıyla ise, PSO algoritmasında parçacıkların çözüm uzayında daha dar bir alana dağıtılmasını sağlayan, *"parçacıkların başlangıç minimum değerleri için atama algoritması"* önerilecektir.

Doktora tezinin ilk bölümünü giriş bölümü oluşturmaktadır. Giriş bölümünde, konunun önemi vurgulanarak; tez çalışmasının amacı ve tez çalışmasında kullanılan yöntem ortaya konmuştur.

Doktora tezinin ikinci bölümünde depo ve depolama ile ilgili kavramlar, depolamanın amacı, depolama fonksiyonları, depo çeşitleri ve depolama ile ilgili süreçler, depo yerleşim düzenlemesinin amacı, depo yerleşim düzenlemesinin aşamaları ve depo yerleşim düzenlemesi tasarımları açıklanmıştır. Aynı bölümde, depo yerleşim düzenlemesi ile ilgili literatürde yer alan çalışmaların kapsamlı bir incelemesi yapılmıştır.

Üçüncü bölümde ise, depo yerleşim düzenlemesi problemi matematiksel olarak ifade edilmiştir. Matematiksel olarak ifade edilen problem kısıtlı çözüm uzayına sahip olduğu için, bu tür problemlerin çözümünde literatürde var olan PSO algoritması ile uyumlu sınırlandırma yöntemleri anlatılacak ve iki yeni sınırlandırma yöntemi önerilmiştir. Yine aynı bölümde, PSO algoritmasında parçacıkların daha dar uzaya dağılmasını sağlayan parçacıkların başlangıç minimum değerleri için atama algoritması önerilmiştir.

Doktora tezinin dördüncü bölümünde, öncelikle değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemine ilişkin, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri için ve standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış bilgisayar programları sunularak, 10 farklı örnek problem üzerinde çalıştırılmıştır. Ardından 10 örnek problemin her iki yöntem ile çözümünden elde edilen sonuçlar ayrı ayrı tablolaştırılmış ve bu sonuç tabloları tek bir tablo haline getirilerek karşılaştırılmıştır. Karşılaştırma sonucunda, en iyi sonucu veren yöntemlerin 10 farklı problem üzerinde depo raf sistemi konfigürasyonu (raf sayısı, raf

uzunluğu, raf yüksekliği) ve depo yerleşim boyutlarına (depo eni, depo boyu) ilişkin verdiği sonuçlar sunulmuştur.

Doktora tezinin beşinci bölümü sonuç bölümüdür. Bu bölümde, doktora tezinin belirlenen amaçları doğrultusunda ortaya çıkan genel sonuçlar değerlendirilerek; tezin depo yerleşim düzenleme konusu ile ilgili olarak literatüre katkılarına, ileride yapılacak çalışmalara ilişkin önerilere yer verilmiştir.

2. GENEL KISIMLAR

Bu bölümde, depo ve depolama ile ilgili kavramlar, depolamanın amacı, depolama fonksiyonları, depo çeşitleri ve depolama ile ilgili süreçler anlatılacaktır. Daha sonra, depo yerleşim düzenlemesinin amacı, depo yerleşim düzenlemesinin aşamaları ve depo yerleşim düzenlemesi tasarımları açıklanacaktır. En son olarak ise, depo yerleşim düzenlemesi ile ilgili literatürde yer alan çalışmaların kapsamlı bir incelemesi sunulacaktır.

2.1. DEPO VE DEPOLAMA

Depo ve depolama ile ilgili tanımlamalar ve temel kavramlar açıklanmadan önce bu kavramlara konu olan stok kavramının açıklanmasına gerek duyulmaktadır.

Her tedarik zincirinde sürekli olarak malzeme akışı olmaktadır. Bu zincir üzerinde herhangi bir noktada herhangi bir nedenden dolayı malzeme akışının kesintiye uğraması ile stoklar meydana gelmektedir. “Stok” tedarik zinciri boyunca yayılmış olan; tedarik zincirindeki üreticiler, distribütörler ve perakendeciler tarafından tutulan hammaddeden süreçteki stoğa ve bitmiş ürüne kadar her türlü malzemeyi içerir (Hugos, 2003). Stoklar, hem üretim hattını desteklemek için kullanılan hammaddeler ve alt montaj parçaları, hem de müşterilere veya tedarik zincirindeki diğer elemanlara satılmayı bekleyen tamamlanmış ürünler olarak tanımlanmaktadır. Literatürde stok ile envanter aynı anlamda kullanılsa da, envanter stokta tutulan malzemelerin listesidir (Walter, 2003).

Stok tutulmasının şirketlere sağladığı birçok avantaj bulunmaktadır. Stok tutulmasının avantajları aşağıda sıralanmaktadır.

- Tedarik zincirinin tüm aşamalarında zincirler arasında tampon görevini görmektedir (Scott ve diğ., 2011).
- Şirketleri talep varyasyonlarının veya arz kısıtlamalarının neden olduğu belirsizliklere karşı korurlar (Scott ve diğ., 2011).

- Kalite hatalarından kaynaklanan malzeme yetersizliğinden korunmayı sağlar. Böylece, hatalı malzemeler yenileri ile hızlı bir şekilde değiştirilebilir (Scott ve diğ., 2011).
- Üretim hattının dengeli üretim yapması sağlanır (Scott ve diğ., 2011).
- Satın almada, üretimde veya taşımada ölçek ekonomisinden faydalanabilmeyi sağlar (Lambert ve diğ., 1998).
- Şirketleri gelecek fiyat dalgalanmalarından veya gelecek üretim azalmalarından kaynaklanan stok yetersizliğine karşı korur (Walter, 2003).
- Beklenmeyen taleplerden veya tedarik süresindeki değişkenliklerden kaynaklanan stoksuzluk olasılığı azaltılırken; müşteri hizmet düzeyi ve/veya müşteri memnuniyeti artırılır (Abdullah ve diğ., 2012).

Endüstrinin herhangi bir sektöründe veya bu sektördeki herhangi bir kuruluşunda tedarikçiden son kullanıcıya kadar olan malzeme akışının kontrolü hayati öneme sahiptir. Malzeme akışının kontrolü düzgün ve tam bir biçimde yapılmadığında, stoklar çoğalmakta ve stoğa yapılan yatırım maliyetleri artmaktadır. Bu bağlamda, tepe yöneticileri malzeme akışının stratejik öneme sahip olduğunun farkındadır. Bilimsel yöntemlerle yapılan stok kontrolü şirketlere fark yaratacak şekilde rekabet avantajı sağlamaktadır (Axsater, 2006).

2.1.1. Depo ve Depolama Kavramı

Günümüz lojistik ve tedarik zinciri yönetiminde, sipariş edilen malzemelerin olabildiğince hızlı şekilde ve en kısa sürede müşteriye ulaştırılabilmesi ve bu teslim zamanın en az sayıda stokla sağlanması gerekmektedir. Siparişlerde istenilen malzemelerin miktarları azalmakta, sipariş içerisindeki malzeme çeşitliliği ise artmaktadır. Bu durum küçük miktarlardaki çok çeşitli malzemenin, daha sık ve daha hızlı teslim zamanlarıyla dağıtılması gerekliliğini ortaya çıkarmıştır (van den Berg ve Zijm, 1999). Bu zaman baskısının en fazla hissedildiği süreç ise tedarik zinciri elemanları içerisindeki başlıca süreçlerden birisi olan depolamadır (Lambert ve diğ., 1998).

Depolama süreci sayesinde üretici ile tüketici arasındaki bağlantı sağlanmış olur. Şirketlerin üretim sürecinde kullanacakları hammaddelerin, yarı mamullerin, yardımcı malzemelerin, bakım işlemlerinde kullandıkları bakım onarım sarf malzemelerinin ve

yedek parçaların, üretim sürecinin çıktıları olan bitmiş ürünlerin stoklanmasında ve stokların yönetilmesinde yapılması gereken işlemlerin bütününe depolama denilmektedir. Tüm bu işlemlerin yapıldığı tesisler ise “depo” olarak adlandırılmaktadır.

Depo, en genel tanımlama ile ticari değere sahip her türlü malzemenin korunup saklandığı tesistir (Shiau ve Lee, 2010). Ancak bu açıklama günümüzde depoları tanımlamakta yetersiz kalmaktadır. Depo, malzemelerin tedarik kaynaklarından belirli bir sistem içerisinde teslim alındığı, ayrımının yapıldığı, kayıtlarının tutulduğu, muhafazasının ve bakımının yapıldığı, siparişe uygun olarak dağıtıldığı ve bu işlemlerin yanı sıra malzemelere katma değer sağlayan diğer operasyonların yer aldığı tesis olarak tanımlanmaktadır. Depo, malzemelerin hammadden başlayarak son müşteriye kadar uzanan tüm faaliyetler dizisinin gerçekleştirilmesini sağlayan stratejik öneme sahip tesistir.

Deponun modern tedarik zincirlerinde önemli bir yeri vardır ve şirketlerin başarılarında veya başarısızlıklarında hayati bir rol oynamaktadır (Frazelle, 2002). Depoların değişik şekilde adlandırılmalarına ve değişik tiplerinin bulunmasına rağmen depolar, genel olarak açık veya kapılı şekilde olabilen ve malzemelerin uygun koşullarda saklanmasını sağlayan tesislerdir.

2.1.2. Depolamanın Amacı ve Önemi

E-ticaretin gelişmesi, tedarik zinciri entegrasyonu, müşteriye etkili ve hızlı yanıt verebilme ve tam zamanında üretim gibi birçok yenilik sayesinde, tedarik zincirindeki üretici ve tüketiciler arasındaki bağın hiç olmadığı kadar kuvvetli olması sağlanmıştır. Bu yeniliklere tedarik zincirinden çıkarılmaya çalışılan depolama sürecinin zincir içerisinde rolü ve görevi değişmiş; ancak, zincirin önemli parçası olmasına engel olunamamıştır (Frazelle, 2002).

Depolama lojistik sistemi içerisinde maliyete neden olmasına rağmen, birçok yönden şirketlere katkıda da bulunmaktadır. Depolamanın şirketlere sağladığı katkılar aşağıda özetlenmektedir (Lambert ve diğ., 1998).

- Üretici ile tüketici arasında tampon görevi üstlenir.
- Malzemelerin daha fazla üretilip saklanması ile birim üretim maliyetlerinin azaltılması sağlanır.

- Malzemelerin daha fazla taşınıp saklanması ile birim taşıma maliyetlerinin azaltılması sağlanır.
- Gelecek alımlar ile miktar indirimlerinden faydalanılır.
- Kesintisiz tedarik kaynağı sağlanır.
- Müşteri servis politikası desteklenir.
- Değişen pazar koşullarına cevap verebilme yeteneği kazanılır.
- İmha edilecek veya tekrar kullanılacak malzemelerin geçici olarak tutulması sağlanır.

Depolamanın lojistik sistemi içerisinde yukarıda saydığımız gibi birçok katkısı olduğu sürece sistem içerisindeki önemi ve yeri kesinlikle göz ardı edilemez. Şirketler depolama için katlanacakları maliyetler ile kıyaslandığında, bu maliyetlere göre daha fazla fayda elde etmektedir. Bunun yanı sıra depolama, malzeme akışında ortaya çıkabilecek kesintinin etkisinin azaltılmasında büyük rol oynamaktadır. Depolama ile daha düzenli ve kesintisiz malzeme akışı sağlanmış olur.

Depolamanın amacı genel olarak yüksek müşteri hizmetini ve düşük maliyeti bir arada sağlayarak, diğer tüm lojistik fonksiyonları desteklemektir. Depolamanın diğer amaçları ise aşağıda sıralanmaktadır (Walter, 2003).

- Tedarik zincirinde kilit noktalarda gerekli depolamanın yapılması
- Malzeme özelliğine bağlı olarak güvenli depolama ortamı sağlanması
- En az hasar ile en uygun ortamın sağlanarak malzemelerin saklanması
- Yüksek müşteri hizmet düzeyinin sağlanması
- Faaliyetlerin yüksek verimlilik ve düşük maliyet ile gerçekleştirilmesi
- Yüksek verimlilik ve yüksek kaynak kullanımının elde edilmesi
- Düzenli ve hatasız malzeme hareketinin kontrol edilmesi
- Hızlı bir şekilde teslim alınması ve depo içerisine hızlı transfer edilmesi
- Teslimat için hızlı sipariş toplanması, sevkiyat bölümüne taşınması ve sevkiyatların konsolide edilmesi
- Stok seviyeleri değişimleri ile verimli bir şekilde başa çıkılması için esnek olunması
- Malzeme rotasyonlarına veya buna benzer özel durumlara cevap verebilmesi

2.1.3. Depolama Fonksiyonları

Depolama sürecinin dört temel fonksiyonu bulunmaktadır. Bu fonksiyonlar; malzeme stoklama fonksiyonu, hareket fonksiyonu, ürün işleme fonksiyonu ve bilgi transfer fonksiyonudur (Acar, 2010).

2.1.3.1. Stoklama Fonksiyonu

Stoklama fonksiyonu depoların ilk akla gelebilecek görevidir. Tedarik zincirindeki müşteriden veya son kullanıcıdan talep gelinceye kadar malzemenin (hammadde, montaj parçası, süreçteki stok veya nihai ürün) saklanmasıdır. Bilindiği gibi arz ile talep her zaman eşit olmayabilir. Bu durumda, talep gelinceye kadar üretilen tüm malzemelerin stoklanması gerekir (Sople, 2007).

Stoklama fonksiyonu üretime destek olması açısından değerlendirildiğinde, sürekli ve düzenli malzeme akışını sağlamak ve malzeme yetersizliğinden ortaya çıkabilecek kesintileri engellemek için kullanılmaktadır. Bu fonksiyon dağıtıma destek olması açısından irdelendiğinde ise şirketlerin rekabet edilebilmesi için pazarda sürekli aktif olarak bulunulması gerekir. Bunun sonucu olarak şirketlerin dağıtım ağlarında stok bulundurması gerekmektedir.

Stoklama, kısa süreli yapılabileceği gibi yarı-kalıcı süreli ve kalıcı şekilde de yapılmaktadır (Voortman, 2004). Kısa süreli stoklama, tersine lojistikte imha edilecek veya tekrar kullanılacak malzemelerin depoların belirli bölgesinde stoklanabilmesi veya gelen malzemelerin talep gelinceye kadar saklanmasıdır. Yarı-kalıcı süreli stoklamaya depolarda tutulan tampon veya güvenlik stokları örnek gösterilebilir. Kalıcı stoklama için ise, devletlerin olağanüstü durumlar için bazı temel insani ihtiyaçları kalıcı şekilde stoklaması örnek olarak verilebilir. Ancak bu tür malzemelerin son kullanma tarihi çok uzun süreli olması gerekmektedir. Bu tür malzemelere madeni yağlar, petrol ve petrol türevi yakıtlar, konserve gıdalar ve sıvı yağlar örnek gösterilebilir.

Stoklama fonksiyonunda yalnızca depoya gelen malzemelerin stoklanması yeterli değildir; ayrıca malzemelere uygun ortam koşulları da sağlanmalıdır. Malzemelerin özelliğine ve karakteristiklerine bağlı olarak uygun aydınlatma, iklimlendirme, havalandırma ve buna benzer dış ortam koşulları sağlanmalıdır. Uygun koşulların sağlanması ile malzemelerde oluşabilecek bozulmaların, çürümelerin ve buna benzer

istenmeyen hasarların oluşması engellenir. Ayrıca, söz konusu hasarlardan oluşan malzeme kayıpları da azaltılır.

2.1.3.2. Hareket Fonksiyonu

Depoların hareket fonksiyonu kapsamında olarak ele alınan beş ayrı fonksiyon bulunmaktadır. Bunlar birleştirme (konsolidasyon), dağıtım/ayrıştırma, sınıflandırma/karıştırma, çapraz sevkiyat ve tersine lojistik fonksiyonlarıdır.

Birleştirme (Konsolidasyon), depo yönetiminde aynı müşteri ve/veya aynı güzergaha ait olan malzemelerin gruplandırılması olarak tanımlanmaktadır (Stroh, 2006). Bir şirketin farklı üretim merkezlerinden gelen çeşitli malzemeleri, merkezi bir depoya yerleştirmesi ve daha sonrasındaki dağıtım için birleştirmesidir (Murphy ve Wood, 2011). Konsolidasyonun ekonomik faydası, yüklemede ölçek ekonomisinden faydalanarak, taşımada ise verimlilik sağlanarak depolama imkânlarından yararlanmak yoluyla taşıma maliyetlerinin düşürülmesidir. Küçük parti sevkiyatların dağıtım merkezlerinde birleştirilerek büyük parti sevkiyatlara dönüştürülmesi ile maliyet avantajı sağlanır (Bowersox ve diğ., 2002).

Dağıtım (Ayrıştırma), birleştirme fonksiyonuna benzemesine rağmen malzemelerin hareket yönü farklıdır. Bu fonksiyonda, üretim merkezinden gelen tam kamyon yükü ve tam vagon yükü malzemeler, müşterilere yakın bir noktadaki dağıtım deposuna taşınır. Bu depodan malzemeler ayrıştırılır ve müşterinin isteğine bağlı olarak az kamyon yükü halinde müşterilere gönderilir (Gourdin, 2006). Üretim merkezinden çıkan malzemeler tam kamyon yükü veya tam vagon yükü şeklinde uzun mesafelere taşınır. Dağıtım deposundan çıkan malzemeler ise az kamyon yükü halinde sevkiyatlara dönüştürülerek kısa mesafe taşınır. Bu sayede, tam kamyon yükü veya tam vagon yükü olmayan sevkiyatların uzun mesafe taşınması azaltılarak taşıma ekonomisinden faydalanılır. Buna ek olarak, tam kamyon yükü veya tam vagon yükü malzeme satın alındığı için satın alma ekonomisinden de yararlanılmış olunur (Wisner ve diğ., 2008).

Sınıflandırma (Karıştırma), Sınıflandırma fonksiyonunda birçok farklı tedarikçiden gelen malzemeler ayrıştırılır ve farklı malzeme grupları birleştirilerek birçok farklı müşteriye sevk edilir. Bu fonksiyon, başlangıç noktası ile varış noktası arasındaki ara lokasyonlarda (depolarda) daha çok gerçekleştirilir (Bowersox ve diğ., 2002).

Tedarikçilerden gelen yüksek hacimli malzemeler depoya gelir, ayrıştırılır ve müşterilerin isteği doğrultusunda küçük hacimli malzeme grupları birleştirilerek yüksek hacimli sevkiyatlar halinde müşterilere gönderilir. Bu fonksiyonda, hem tedarik kısmında hem de sevkiyat kısmında taşıma ekonomisinden yararlanılmaktadır.

Çapraz Sevkiyat fonksiyonu, depo ve dağıtım merkezlerinde malzeme elleçleme sayısının azaltılması, boşaltma ve yükleme yapılırken malzeme üzerinde oluşabilecek hasarların en aza indirilmesi ve dağıtım hızının artırılması için yapılan bir faaliyettir (Coyle ve diğ., 2003). Bu fonksiyon sayesinde depolar malzeme stoklama noktası olma özelliğinin yerine malzeme akışı koordinasyonu noktası olma özelliğini elde eder (Simchi-Levi ve diğ., 2003). Çapraz sevkiyat fonksiyonunda tedarikçilerden gelen malzemeler, kartonlar ve paletler nakliye kamyonlarından alınarak doğrudan sevkiyat araçlarına yüklenir. Gelen malzemeler genellikle çok az depolanarak veya hiç depolanmadan müşterilere gönderilir (Choy ve diğ., 2012). Çapraz sevkiyat ile hızlı malzeme akışı sağlanır ve stok tutma ile ilgili maliyetler ortadan kaldırır. Gelen malzemeler stoklanmadığı için stoklama alanı ihtiyacında da azalma sağlanır. Bunun yanı sıra, gelen malzemelerde yapılması gereken niteliksel ve niceliksel kontrollerde de azalma olur. Çapraz sevkiyat fonksiyonun etkin şekilde kullanılması ile daha düşük hacimli siparişler verilerek daha sık sevkiyat yapılır. Daha sık aralıklarla yapılan sevkiyatlara bağlı olarak, eksik olan siparişlerin daha hızlı bir şekilde tamamlanması sağlanır. Napolitano (2000) çapraz sevkiyat işleminin planlanması, tasarlanması ve uygulanması için pratik bir kılavuz önermiştir.

Tersine Lojistik, kullanım ömürlerinin sonuna gelmiş malzemelere tekrar değer kazandırılması veya bu malzemelerin imha edilmesi için tüketicilerden veya müşteri hizmet merkezlerinden dönen malzemelere yapılan süreçtir (Dowlatshahi, 2012). Bir diğer deyişle, malzeme akışı müşteriden veya tedarik zinciri elemanından üreticiye doğru ters bir şekilde yapılmaktadır. Tersine lojistik malzemeleri; müşteri iadelerini, kusurlu malzemeleri, taşıma sırasında kullanılan taşıma paketleri veya kutularını ve taşıma paketlerinde hasar oluşmasını engelleyici malzemeleri içerir. Kullanım ömürlerinin sonuna gelmiş malzemelerin tekrar kazanılması ile hem değer kazandırılarak tasarruf elde edilir; hem de çevreye verilebilecek zarar azaltılır. Ancak bu durum "yeşil lojistik" ile karıştırılmamalıdır. Yeşil lojistiğin birincil motivasyonu

çevresel kararlar iken tersine lojistikte birincil öncelik ticarettir (Scott ve diğ., 2011). Malzemenin tekrar kazanılması ile hammadde ihtiyacı da azaltılmış olur. Bu fonksiyonun gerçekleştirilmesinde depolar kritik bir rol oynarlar. Geri dönen malzemelerin toplanması, sınıflandırılması, ayrılması ve birleştirilerek gönderilmesi gibi birçok operasyon depolarda gerçekleştirilir.

2.1.3.3. Ürün İşleme Fonksiyonu

Ürün işleme fonksiyonu, belirli seviye üretimi gerçekleştirilmiş ancak tamamen sonlandırılmamış malzemelerin son işlemlerinin yapıldığı operasyonlar bütünüdür. Bu fonksiyonda bazı basit üretim işlemleri veya paketleme işlemleri yapılarak malzemelerin son şekli verilir. Bu işlemler arasında depoya gelen malzemelerin tekrar paketlenmesi, etiketlenmesi, işlenmesi, promosyon veya diğer amaçlar için farklı malzemelerin birleştirilmesi veya barkodlanması sayılabilir. Söz konusu işlemlere katma değerli işlemler de denilmektedir. Depolar, ürün işleme fonksiyonu sayesinde maliyet yükleyen birimler olmaktan çıkarak, kazanç sağlayan tesislere dönüşürler. Diğer bir ekonomik faydası ise, müşterilerin özel isteklerini karşılamak amacıyla malzemelere yapılması gereken son işlemlerin bu fonksiyonda yapılması ile siparişlerin gecikme riskinin ve stok miktarının azaltılmasıdır. Bu fonksiyon yapılmadığı durumlarda ise, iki farklı çözüm yöntemi ortaya çıkmaktadır. İlk çözüm yöntemi, müşteri isteklerini karşılamak için çok sayıda farklı türde ve tipte malzemelerin stokta tutulması gerekir. Bunun sonucu olarak, elde tutma maliyeti ve stokta tutmak için katlanılan diğer maliyetler de yükselmektedir. İkinci çözüm yöntemi ise, müşterilerin malzemeler üzerinde özel istekleri olduğunda, malzeme bu özel isteklere göre üretilip müşteriye gönderilir. Ancak bu çözümde malzemenin tedarik süresinin artmasından kaynaklanan maliyetler ortaya çıkmaktadır.

2.1.3.4. Bilgi Transfer Fonksiyonu

Bilgi transferi, tedarik zinciri içerisinde kaynaklardan veya bu zincir ile etkileşim içerisinde olan diğer kaynaklardan elde edilen ham bilgilerin, kullanıcıların isteklerine ve ihtiyaçlarına göre düzenlenip istenilen bir biçime geldikten sonra şirket içerisinde veya şirketler arası en hızlı yoldan ve en kolay ulaşılabilen şekilde paylaşılmasıdır. Hammaddelerin, ara montaj parçalarının ve bitmiş ürünlerin takibi, depolardan giriş-çıkış miktarları, malzemelerin lokasyon bilgileri, kalan raf ömürleri ve buna benzer

depolar için hayati olan bilgilerin hızlı ve kolay bir şekilde işlenip transfer edilmesi gerekmektedir.

Bilgi transfer fonksiyonunun depolamanın diğer temel fonksiyonları ile eş zamanlı olarak gerçekleşmesi gerekir. Depolama sürecinde sadece malzeme stoklanmasının, malzeme hareketinin ve ürünlerin işlenmesinin yönetilmesi değil, bu fonksiyonları destekleyen bilginin de yönetilmesi zorunludur (Voortman, 2004).

Bilgi transferinin sadece departmanlar arasında değil, şirketler arasında da gerçekleştirilmesi gerekmektedir. Tedarik zinciri içerisinde elde edilen bilgiler tüm zincir boyunca çift yönlü aktarılmalıdır. Bilginin yetersiz transferi ile aşırı stok maliyeti, düşük müşteri hizmet seviyesi, kayıp kazançlar, hatalı kapasite planları gibi istenmeyen durumlar ortaya çıkabilmektedir (Lee ve diğ., 1997). İyi bir bilgi transfer fonksiyonu ile bu maliyetlerde de azalmalar sağlanabilmektedir.

Bilgi transfer fonksiyonunun geliştirilebilmesi için gelişen bilgi teknolojilerinden yararlanılması ve ihtiyaca en uygun teknolojinin seçilerek yatırım yapılması gerekmektedir. Ancak bilginin transferinin sorunsuz olması için bu teknolojinin sadece departmanlar arası veya şirket içi için uygun olması yetmez; şirketin etkileşim içerisinde olduğu diğer şirketler için de uygun olması gerekir. Bu teknolojiler maliyetli olmasına rağmen depolar için vazgeçilmeyen sistemlerdir.

2.1.4. Depo Çeşitleri

Tedarik zincirindeki konumlarına göre dört farklı depo çeşidi bulunmaktadır. Bunlar, üretim/fabrika depoları, merkezi depolar, dağıtım depoları ve perakende depolarıdır. Tedarik zincirindeki konumlarına göre depolarda önem verilecek operasyonlar ve depoların sahip olması gereken özellikler değişiklik göstermektedir.

2.1.4.1. Üretim/Fabrika Depoları

Üretim/fabrika depoları, üretim veya montaj ile ilgili tüm hammaddeler, ara montaj parçaları, süreçteki stokların ve bitmiş malzemelerin saklanması için kullanılır (van den Berg, 2007). Üretim sürecinden çıkan bitmiş malzemeler sevk edilinceye kadar stoklanır. Daha sonra bitmiş malzemeler merkezi depoya, dağıtım deposuna ya da müşteriye sevk edilir. Müşteriye direkt olarak sevkiyat çok sık karşılaşılan bir durum değildir.

Üretim/fabrika depolarında tüm malzemeler uzun süreli stoklanır. Malzemelerin uzun süreli stoklanmasında uygun maliyetle stoklama ön plana çıkmaktadır. Diğer bir anlatımla, yüksek miktarda satın alınan malzemelerin uzun süreli stoklanması sırasında düşük maliyetli olacak şekilde saklanması öncelikli bir durumdur. Bu tür depoların hedefleri ise düşük yatırım maliyeti ve düşük operasyon maliyetidir.

Üretim/fabrika depolarının tasarım kriteri stoklama kapasitesidir. (Rouwenhorst ve diğ., 2000). Bu tür depolarda stoklama yoğunluğu yüksek olan raf sistemleri tercih edilmektedir. Yüksek yoğunluğa sahip raf sistemlerine uygun malzeme elleçleme ekipmanları da belirlenmelidir. Ancak bu sistemler seçilirken maliyetler göz önünde bulundurulmalıdır.

Üretim/fabrika depoları çoğunlukla lokasyon olarak üretim tesislerinin içerisinde veya çok yakınında bulunur. Bu sayede üretim tesisi ile depo arasındaki malzeme akışı kolaylaştırılır. Üretim hattına kesintisiz ve sürekli olarak malzeme beslenir. Üretime girecek olan malzemelerin hazırlık aşamaları bu depolarda yapılır. Bu tür depoların tasarımı sırasında malzemenin kesintisiz akışını sağlayacak depo iç düzenlemeleri yapılmalıdır.

2.1.4.2. Merkezi Depolar

Merkezi depolar, tedarik zincirine gelen taleplerin çoğunun karşılandığı tesislerdir. Bu depolardan belirli bir bölge içerisindeki belirli büyüklükteki dağıtım merkezlerine ve müşterilere hizmet verilmektedir (Farahani ve diğ., 2011). Merkezi depolar, stoklanan malzemelerin miktarları bakımından fabrika depolarına benzemektedir. Ancak üretim/fabrika depolarında üretime girecek veya bitmiş malzemeler saklanırken; merkezi depolarda sadece bitmiş malzemeler saklanmaktadır.

Merkezi depolar stoklama kriteri bakımından üretim/fabrika depolarına benzemektedir. Bu tür depolarda stok yoğunluğu yüksek depo raf sistemleri tercih edilmektedir. Ancak buradaki temel fark, merkezi depoların üretim/fabrika depolarına göre daha hızlı cevap verebilme yeteneğine sahip olmaları beklenmektedir. Diğer bir deyişle, üretim/fabrika depolarında stoklama operasyonu önemliyken merkezi depolarda yerleştirme ve stoklama operasyonları daha önemlidir. Ancak bu süreçlere önem verilirken tıpkı

üretim/fabrika depolarında olduğu gibi düşük yatırım maliyeti ve düşük operasyon maliyeti sağlanmalıdır.

Merkezi depolarda, dağıtım ve operasyon süreçlerini tek bir merkez tarafından kontrol ve koordine edilmesi ile yönetim faaliyetleri daha etkin şekilde gerçekleştirilir. Merkezi depolarda yüksek miktarlarda tutulan stoklar sayesinde müşterilere daha hızlı sürede ve istenilen seviyede cevap verilmiş olur (Görçün, 2013).

2.1.4.3. Dağıtım Depoları

Malzemelerin saklandığı, siparişlerin daha kısa sürede sevk edildiği ve daha sık sevkiyat yapılabilen yüksek hacimli depolardır. Bu tür depolarda, müşteri siparişlerin karşılanması için çok çeşitli malzemeler stoklanır. Depodaki malzeme çeşitliliği yüksek olmasına rağmen, malzeme miktarı küçük olabilir. Saklanan malzemelerin hacmi ve miktarları düşüktür.

Dağıtım depolarını diğer depo çeşitlerinden ayıran en temel özelliklerden birisi, üretim/fabrika depolarında veya merkezi depolarda kapasite sayısı önemliyken; dağıtım merkezlerinde kapasitenin kullanım oranının önemli olmasıdır. Dağıtım depoları değerlendirilirken kullanım oranlarına bakılması gerekir. Diğer bir ayrım ise, stoklanan malzemelerin paketleme türlerindedir. Üretim/fabrika depolarında ve merkezi depolarda çoğunlukla depoya gelen malzemeler paletler halinde saklanmaktadır. Dağıtım depolarında ise, genellikle malzemeler paletler ve kutular halinde saklanmaktadır. Bu durum, dağıtım deposu tasarımı sırasında dikkat edilmesi gereken bir konudur.

Dağıtım depolarının, diğer depolardan ayrılması sağlayan diğer bir özellik ise, katma değerli operasyonların yapılmasıdır. Katma değerli operasyonların varlığı sayesinde şirketlerin gelir elde etmesi ile depolama maliyetleri azaltılarak kazanç sağlanır. Ancak katma değerli işlemlerin yapılabilmesi için özel ekipmanların ve işgücünün olması gerekir.

Dağıtım depolarında yapılan en temel operasyon, depolamaya ek olarak, sipariş toplama sürecidir (Parikh ve Meller, 2008). Dağıtım depolarına gelen müşteri siparişleri çok fazla malzeme çeşitliliğine sahiptir. Malzeme çeşitliliğinin artması ile farklı malzemeler bir araya getirilerek birleştirilmektedir. Bu durum karmaşık ve dolayısıyla maliyetli sipariş sürecine neden olacaktır.

Dağıtım depolarında dikkat edilmesi gereken diğer bir kriter ise, sipariş operasyonunun sonucu olan sipariş işlem hacmidir. Bu tür depolarda istenilen sipariş işlem hacmi elde edilirken minimum yatırım maliyeti ve minimum operasyon maliyeti ile sağlanmalıdır (Rouwenhorst ve diğ., 2000). Karmaşıklığı azaltmanın ve yüksek işlem hacminin sağlanmasının yolu ise, bilgi teknolojilerine yapılacak yatırımlar ile gerçekleşir. Yeni teknolojilerin kullanılması ile maliyetli ve karmaşık yapı azaltılarak, daha iyi yönetilir ve kontrol edilir.

Dağıtım depolarının kapasitelerinin çok yüksek olmaması ve malzeme çeşitliliğinin fazla olması, stoklanan malzemelerin stok miktarlarının düşük olmasına neden olur. Müşteri hizmet seviyesinin yüksek tutulabilmesi için stok devir hızlarının yüksek olması gerekmektedir. Bu sayede az sayıda stokla müşterilerin talepleri karşılanır. Stok devir hızı yükseldikçe daha az kapasite ile istenilen müşteri hizmet düzeyi elde edilir.

2.1.4.4. Perakende Depoları

Tedarik zincirinin en sonunda yer alan perakende depoları, alıcı tarafından doğrudan tüketim için küçük partiler halinde fiziksel malzeme satan sabit yerlerdir (Abdullah ve diğ., 2012). Perakende depolarında malzeme çeşitliği çok yüksek olmasına rağmen, stoklanan malzeme miktarları ise çok düşüktür.

Perakende depoları, son kullanıcılar ile tedarik zincirindeki tüm elemanların aracısı konumundadır. Perakende depolarının müşteri isteklerini çok hızlı şekilde karşılaması gereken tesisler olması, bu noktada her malzemeden stok tutulmasını gerektirir. Malzemeler yüksek miktarlarda stoklanmadığı için biten malzemelerin yerine yeni malzemelerin siparişleri verilmesi gerekmektedir. Yenileme siparişlerinin iyi işlememesi durumunda kısa dönemde kayıp satışlar gerçekleşirken, uzun sürede müşteri memnuniyetsizliği ortaya çıkmaktadır.

Perakende depolarında temel noktalar malzeme özellikleri ve bu özelliklere bağlı olan depolamadır. Kozmetik, gıda, dayanıklı tüketim malzemeleri, hazır giyim ve buna benzer birçok farklı sektörden malzeme bulunduran perakende depolarında her malzeme grubuna uygun malzeme akışının ve depolama sürecinin belirlenmesi gerekmektedir. Depo yerleşim düzenlemesi bu süreçler dikkate alınarak yapılmalıdır. Yerleşim düzenlemesi sırasında diğer bir dikkat edilmesi gereken husus ise, her bir malzeme

grubunun farklı depolama koşullarına ve iklimlendirmeye ihtiyaç duymasıdır. Bu bağlamda, malzeme özelliğine ve karakteristiğine bağlı olarak ısıtma, soğutma, havalandırma, aydınlatma ve bunlara benzer koşulları sağlayan depo lokasyonları belirlenmelidir. Tabii ki bu durum depo lokasyonları ile sınırlı kalmamalı, bu lokasyonların amaca uygun olacak şekilde entegrasyonuna da dikkat edilmelidir.

Çok farklı sektörden çok farklı sayıda malzeme içeren perakende depolarında malzeme takibinin ve izlenmesinin hatasız, hızlı ve eş zamanlı yapılabilmesi için yüksek bilgi teknolojilerine sahip depolara ihtiyaç duyulmaktadır. Bilgi teknolojilerinden faydalanılması ile istenilen hıza ve malzeme takibe ulaşılabilinir.

2.1.5. Depolamadaki Temel Süreçler

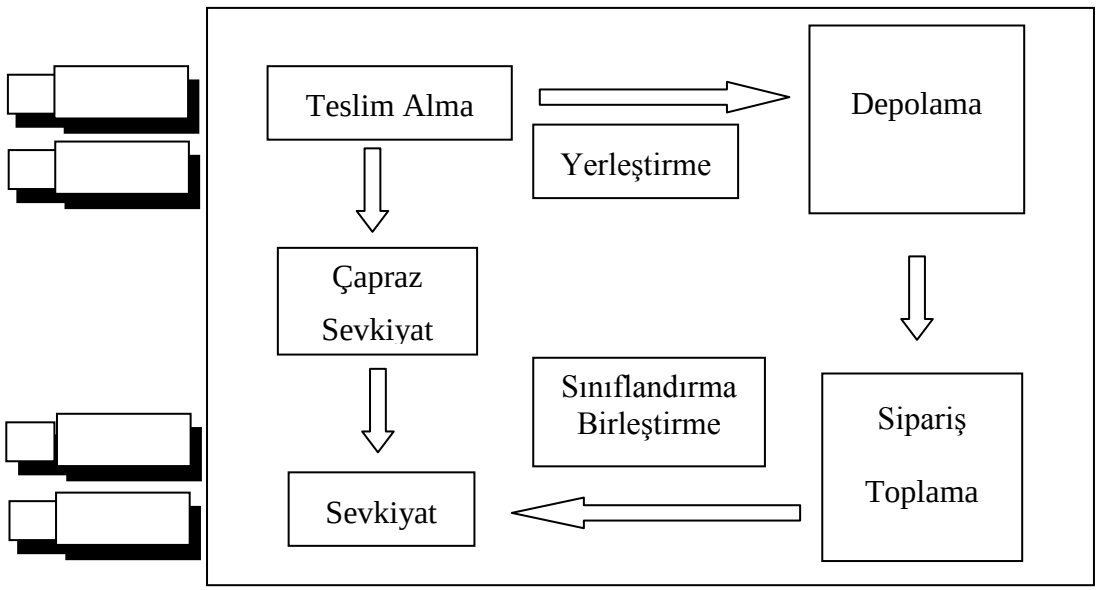
Depolarda daha önceki bölümlerde ele alınan malzeme stoklama, hareket, ürün işleme ve bilgi transfer fonksiyonları bulunmaktadır. Bu fonksiyonların yerine getirilmesi için depolarda birçok operasyonun yapılması gerekmektedir. Depolarda kullanılan bilgi sistemlerinin, teknolojilerinin veya ekipmanlarının sürekli olarak gelişmesine rağmen; depolamadaki operasyonlar çok fazla değişime uğramamıştır. Depolama operasyonlarında büyük değişimler yaşanmamasına rağmen; kullanılan son teknoloji sistemler ve ekipmanlar sayesinde depolamadaki operasyonlar daha rahat kontrol edilebilir, daha rahat yönetilebilir ve daha iyi ölçülebilir hale gelmiştir.

Depolamada yer alan süreçlerin belirli bir düzen içermesi gerekmektedir. Depoda stoklanması istenen malzemelerin müşteri talebi gelinceye kadar saklanması ve müşteri talebi geldiğinde hızlı toplanması ve hatasız şekilde gönderilmesi için depolama sürecinin belirli bir sıra izlemesi gerekmektedir. Depolardan beklenilen müşteri hizmet düzeyi ve çıkış miktarları izlenmesi gereken yolun başarısına bağlı olarak değişmektedir. Stoklanacak malzemenin özelliğine ve tedarik zincirindeki konumuna göre depolama süreçleri ve söz konusu süreçlerde izlenen sıralama değişmektedir. Sektörel bazda depolama süreçlerinin önemlilikleri ve sıralamaları değişmesine rağmen, çoğu depo çeşidinde aşağıdaki süreçler bulunmaktadır. Teslim alma süreci

- Yerleştirme süreci
- Depolama süreci
- Sipariş toplama süreci

- Sınıflandırma ve birleştirme süreci
- Sevkiyat süreci
- Aktarma (çapraz sevkiyat) süreci

Bu süreçler daha iyi anlaşılabilmesi için Şekil 2.1.'de görselleştirilmiştir. Şekil 2.1.'de verilen depolamadaki temel süreçler şekli, Tompkins ve diğ. (2010)'den uyarlanmıştır.



Şekil 2.1: Depolamadaki temel süreçleri (Tompkins ve diğ., 2010).

Depolamada yer alan temel süreçler aşağıda ayrıntılı bir biçimde açıklanmaktadır.

2.1.5.1. Teslim Alma

Teslim alma süreci, depoya gelen her türlü malzemenin karşılaşacağı ilk süreçtir. Söz konusu süreç, depoda saklanacak tüm malzemelerin sistemli bir şekilde kabulü, sipariş edilen malzemenin niteliksel ve niceliksel bakımdan güvence altına alınması ve kullanılan depo bilgi sisteminin güncellenmesi ile ilgili tüm faaliyetlerin toplamıdır. Ayrıca bu süreç, taşıma paketlerinin açılması ve tekrar paketlemeyi de kapsayabilir (Bidgoli, 2010).

Teslim alma süreci, taşıma aracının teslim alma için kullanılan mal kabul kapısına yanaşması ile başlar. Taşıma aracı emniyete alındıktan sonra fiziksel malzeme akışı başlar. Taşıma aracından malzeme boşaltılmasında depoya gelen malzemelerin taşıma paketlerine bağlı olarak kullanılacak ekipmanlar değişiklik gösterebilir.

Teslim alma sürecinin devamında ise, araçtan indirilen malzemeler ile verilen siparişler nicelik ve nitelik olarak kontrol edilir. Bu kontrol süreci, malzeme karakteristiğine göre değişiklik gösterebilir. Kontrol süreci bazı malzeme gruplarında basit sayma veya ağırlık ölçme olabileceği gibi, bazı malzemeler için daha karmaşık kimyasal yöntemlerle kontrolü içerebilir. Kontrol süresi malzemelere yapılan kontrol prosedürüne bağlı olarak uzun veya kısa olabilmektedir.

Teslim alma sürecinde yapılan diğer aktivite tekrar paketlemedir. Depoya gelen malzemeler depoda saklanabilecek bir paket yapısında olmayabilir. Bazı durumlarda ise diğer tedarikçilerden malzeme karması şeklinde sevkiyat gelebilmektedir. Bu şekilde teslim alınan veya paket yapısı uygun olmayan malzemeler, tekrar paketlenerek stoklanmaya hazır hale getirilir.

Teslim alma sürecinde faydalı olabilecek birtakım prensipler bulunmaktadır. Söz konusu prensiplerin kullanılması ile malzeme akışı kolaylaştırılmakta ve hızlandırılmaktadır (Tompkins ve diğ., 2010). Bu prensipler aşağıda yer almaktadır.

- **Teslim almamak:** Bazı malzemeler tedarikçilerden direkt olarak müşterilere sevk edilir. Bu sayede zamandan ve işgücünden kazanımlar elde edilir.
- **Ön teslim:** Teslim alma sonrası yapılması gereken operasyonların malzemeler depoya gelmeden yapılması ile teslim alma süreci hızlandırılır.
- **Çapraz sevkiyat:** Çapraz sevkiyat olabilecek malzemeler belirlenerek sürecin çapraz sevkiyat sürecine uygun olacak şekilde değiştirilmesi ile teslim süreleri kısalır.
- **İleri veya rezerve alanına yerleştirme:** Çapraz sevkiyat olmayan malzemelerin direkt olarak ileri veya rezerve alanına kaldırılması ile malzeme elleçleme sayısı azaltılır. Bunun yapılabilmesi için malzemelerin yerleştirilmesi gereken yerlerinin önceden belirlenmesi ve sürece uygun malzeme elleçleme ekipmanına sahip olunması gerekmektedir.
- **Etkin yerleşim için gelen malzemenin ayrıştırılması:** Depoya gelen malzemeler teslim alma sürecinde sipariş toplama süreci dikkate alınarak gruplara ayrılır ve depoya yerleştirilir.
- **Yerleştirme ile toplama sürecinin birleştirilmesi:** Depoya gelen malzemeler ve depodan giden malzemeler için gerçekleştirilen süreçlerin birleştirilmesi ile

elleçleme sayısı azaltılır. Ekipmanların boş taşıma miktarları azaltılarak, etkin malzeme yerleştirme ve toplama elde edilir.

- **Kaynak dengeleme:** Malzeme getiren taşıyıcıların çizelgelenmesi ve zaman kaybettiren siparişlerin yoğun olmayan zamanlarda yapılması ile kaynak dengelemesi sağlanır. Kaynak dengelemesinin yapılması için etkin kullanılan bilgi teknolojilerine ihtiyaç duyulmaktadır.
- **İstasyonlar arası malzeme akışındaki taşımaların azaltılması:** Toplama sırasında yapılması gereken etiketleme, tekrar paketleme gibi operasyonların tek bir noktada yapılması ile malzeme elleçleme ve taşıma azaltılır.

2.1.5.2. Yerleştirme Süreci

Depoya gelen malzemelerin teslim alındıktan sonra depoya yerleştirilmesi sürecidir. Yerleştirme süreci malzeme elleçlemeyi, lokasyon doğrulamayı ve malzeme yerleştirmeyi içerir (Frazelle, 2002). Depolama politikasına bağlı olarak bu süreç farklılık gösterebilir.

Rassal depolama politikası belirlenmiş depolarda, malzemeler uygun durumda olan herhangi bir stok pozisyonuna yerleştirilir. Rassal depolama politikasının tam tersi olan sabit depolama politikasında, malzemeler karar vericiler tarafından belirlenmiş stok pozisyonlarından en uygun olan stok pozisyonuna yerleştirilir. ABC sınıflandırma politikası ve ürün ailesi gruplama politikası, sabit depolama politikasına benzemektedir. Rezerve/ileri depolama politikasında ise, malzemeler rezerve alanına taşınabilir. Bazı durumlarda gelen malzemeler direkt olarak ileri depolama alanına da yerleştirilebilir. Yerleştirme sürecinde kullanılan tüm bu depolama politikaları bir sonraki bölümde detaylı şekilde anlatılacaktır.

2.1.5.3. Depolama Süreci

Depolama süreci, malzemelere talep gelinceye kadar fiziksel olarak saklanmasıdır. Depolama süreci stokta bulunan malzemelerin sayısına, özelliğine, miktarına ve taşıma paketlerinin özelliğine bağlı olarak değişir (Frazelle, 2002). Günümüzde çok farklı tipte ve özellikte depolama sistemleri vardır. Zeminde ve raflarda stoklanan manuel depolama sistemleri olduğu gibi, tamamen bilgisayar ile yönetilen otomatik depolama sistemleri de bulunmaktadır.

Depolama sürecinde kullanılan depolama politikaları; rassal depolama politikası, sabit depolama politikası, ABC sınıflandırma politikası, ürün ailesi gruplama politikası ve rezerve/ileri depolama politikasıdır.

- **Rassal depolama politikası:** Depolama alanında malzemelerin yerleşimi için herhangi bir sınırlama bulunmamaktadır. Malzemeler depolardaki ilk uygun lokasyona yerleştirilir (Abbasi, 2011). Bu politikanın uygulanması ve yönetilmesi karmaşık ve zor olmasına rağmen, stok pozisyonlarının etkin kullanımı sayesinde depolama kapasitesi ihtiyacı azaltılır.
- **Sabit depolama politikası:** Bu politikada, malzemeler önceden belirlenmiş stok pozisyonlarına yerleştirilir. Politikanın uygulanması ve yönetilmesi rassal depolamaya göre daha kolaydır. Ancak malzeme olsa da olmasa da, stok posizyonu o malzeme için ayrıldığından, başka bir malzeme o stok pozisyonuna yerleştirilemez. Bu durumda depolama kapasitesi ihtiyacı artacak ve stok pozisyonu ihtiyacı yükselecektir.
- **ABC sınıflandırma politikası:** Bu depolama politikası rassal depolama politikası ile sabit depolama politikası arasında bir politikadır. Tüm malzemeler sipariş miktarına bağlı olarak sınıflandırılır. Malzemeler bu sınıflandırmaya göre önceden belirlenmiş yerlere rastgele olarak dağıtılır (Petersen ve Aase, 2004). Bu politikanın yönetilmesi ve uygulanması rassal depolama politikasına göre nispeten daha kolaydır. Stok pozisyonu sayısı ise sabit depolama politikasına göre nispeten daha düşüktür.
- **Ürün ailesi gruplama politikası:** Şimdiye kadar anlatılan tüm depolama politikalarında malzemeler arasındaki olası ilişkiler göz ardı edilmiştir. Benzer malzemelerin depolama alanının aynı bölgesinde stoklanmasına ürün ailesi gruplama politikası denir. Ürün ailesi gruplama politikasında, depolanan malzemelere eş zamanlı olarak ihtiyaç duyuluyorsa, bu malzemelerin birbirlerine yakın yerleştirilmesi amaçlanmaktadır (Rouwenhorst ve diğ., 2000).
- **Rezerve/ileri depolama politikası:** Malzemeler sipariş toplama sürecini hızlandırmak için yığın stoklar ve sipariş toplama stokları halinde ayrıştırılır. Malzemelerin paletler (yığınlar) halinde ve en ekonomik şekilde saklandığı alanlar rezerve alanlarıdır. Malzemelerin paletler (yığınlar) ve kutular (bozulmuş yığınlar) halinde saklandığı alanlar ise ileri depolama alanlarıdır. Siparişler

sadece daha küçük alan olan ileri depolama alanından karşılanır. Rezerve alanları, ileri toplama alanlarındaki malzemelerin stok sayıları tekrar sipariş verme noktasına kadar düştüğünde dahili malzeme besleme alanı olarak kullanılır (Strack ve Pochet, 2010).

2.1.5.4. Sipariş Toplama Süreci

Şirketler depolama maliyetlerini düşürmek, depo ve dağıtım merkezlerinin içinde verimliliği artırmak için sipariş toplama sürecine dikkat etmektedirler. Sipariş toplama belirli bir müşteri isteğine yanıt olarak depolardan (veya tampon bölgelerden) ürünlerin alınması işlemidir. Sipariş toplama süreci manuel sistemli depolarda emek-yoğun, otomatik sistemli depolarda ise sermaye-yoğun bir süreçtir (Kosner ve diğ., 2007). Bu bağlamda, sipariş toplama süreci verimlik artışı istendiğinde düşünülmesi gereken ilk süreçtir.

Sipariş toplama süreci; müşteri taleplerini sınıflandırmayı ve kümelemeyi, stoktaki malzemeleri siparişlere atamayı, depoya siparişleri serbest bırakmayı, stok lokasyonlarından malzemelerin toplanmasını ve siparişin sonlandırılmasını içerir. Her müşteri siparişi, bir veya daha fazla sipariş satırı içermekte ve her satır ise belirli bir malzeme ve istenilen miktarı göstermektedir (Kosner ve diğ., 2007).

Sipariş, belirli bir hedefe veya müşteriye gönderilecek malzeme ve malzeme çeşitlerinin listesidir. Sipariş toplama sürecinde, siparişlerin hazırlanması için bu listelerin toplama listelerine dönüştürülmesi gerekmektedir. Toplama listelerinin hazırlanması ise sipariş toplama yöntemlerine göre değişiklik göstermektedir.

Literatürde sipariş toplama yöntemleri için ortak bir karar olmamasına rağmen, genel olarak dört farklı yöntem bulunmaktadır. Bunlar; tek sipariş toplama, parti toplama, bölgesel toplama ve dalgasal toplamadır. Bu sipariş toplama yöntemlerinin altında farklı alt toplama yöntemleri de bulunmaktadır (Kosner ve diğ., 2007; Gagliardi ve diğ.,2008).

Tek (kesikli) sipariş toplama: Tek (kesikli) sipariş toplamada sipariş toplayıcı bir toplama turunda sadece bir müşteri talebini karşılayacak malzemeleri toplar. Bu sipariş toplama yönteminde toplayıcı tek bir sipariş üzerinde çalıştığı için hızlı toplama sağlanır. Acil sipariş karşılanmasında ideal bir toplama yöntemidir. Toplama süreci

daha basit ve toplayıcı için tek bir toplama listesi hazırlanır. Tek bir toplama liste olduğu için siparişteki tüm malzemelerin toplanmama riski azaltılır.

Parti toplama: Parti toplama, birçok siparişin birleştirilerek toplayıcı tarafından bir turda toplanmasıdır. Bu toplama yöntemi ile toplama mesafeleri azalır ve toplama süresi kısalır (Lin ve Lu, 1999). Tek (kesikli) sipariş toplama düşük talepli büyük siparişler ve acil siparişler için uygunken; parti toplama yüksek talepli küçük siparişler için uygundur. "Toplarken ayırma" ve "Topla ve ayırma" olmak üzere iki şekilde gerçekleştirilir (Park, 2012).

Toplayıcı, müşteri siparişlerindeki malzemeleri toplarken ayırma işlemini de yapıyorsa "toplarken ayırma" denir. Bu toplamada, toplayıcı birden fazla talebi aynı anda karşılarken, her bir siparişi farklı toplama kaplarına yerleştirerek hazırlar. Müşteri siparişleri farklı kaplara yerleştirildiği için toplama bittikten sonra ayırma yapılmasına gerek yoktur. Ancak toplama sırasında ayırma işlemi de yapıldığı için toplama sayısı daha düşüktür. "Topla ve ayırma" şeklindeki sipariş karşılamada ise, siparişlerdeki tüm malzemeler birlikte toplanır. Tüm malzemeler toplandıktan sonra müşteri siparişlerine göre ayırma işlemi yapılır. Toplama sayısı yüksektir ancak manuel veya otomatik ayırma işlemi yapılması gerekmektedir (Parikh ve Meller, 2008).

Bölgesel toplama: Bölgesel toplamadan önceki toplamalarda, sipariş toplayıcı deponun koridorları arasında dolaşarak toplamayı gerçekleştirirdi. Bölgesel toplamada ise, deponun stoklama alanları belirli bölgelere ayrılır ve her bölgeye bir toplayıcı atanır. Bu yöntemde, müşteri siparişleri bir veya daha fazla toplama listesine dağıtılır (Gagliardi ve diğ., 2008). Toplayıcı, bölgesine gelen toplama listesine göre malzemeleri toplar. Bölgesel toplamanın uygulanması için üç farklı yöntem bulunmaktadır. Bunlar, sıralı (ilerici veya topla ve geçir) toplama, senkronize toplama ve insan zinciri olmak üzere üç şekilde uygulanır (Tompkins ve diğ., 2010).

Sıralı (ilerici veya topla ve geçir) toplamada, müşteri siparişi bölgelere göre toplama listelerine ayrılır. İlk toplayıcıya toplama listesi verilir ve ilk bölgedeki toplayıcı toplama listesi göre kendi bölgesindeki tüm malzemeleri toplar. Toplayıcı bölgesindeki tüm malzemeleri topladıktan sonra toplama listesini bir sonraki bölgedeki sipariş toplayıcısına iletir. Toplama listesindeki tüm malzemeler toplandıktan sonra sipariş

hazırlanmış olur. Bu yöntem çoğunlukla parti toplama yöntemi ile birlikte kullanılır; ancak, tek sipariş toplama yöntemine de uygulanabilir. Bu yöntemin faydası ise, malzemeler toplandıktan sonra ayırma ve birleştirme işleminin yapılmasına ihtiyaç yoktur.

Senkronize toplamada, siparişler aynı sıralı toplamada olduğu gibi bölgelere ayrılarak toplama listeleri oluşturulur. Ancak, sıralı toplamada bir bölgenin toplaması bitmeden diğer toplama başlamadığı için toplayıcı atıl durumda kalabilmektedir. Senkronize toplama ise, toplama listeleri bölgelerdeki sipariş toplayıcılarına aynı anda gönderilir. Toplayıcı kendi bölgesini bağımsız şekilde toplar. Tüm malzemeler toplandıktan sonra ayırma ve birleştirme işlemleri yapılmalıdır. Sıralı toplamaya göre daha fazla sipariş toplanmasına rağmen ayırma ve birleştirme operasyonunun yapılması gerekmektedir.

Parti toplamın son yöntemi ise, insan zinciri toplama yöntemidir. Bu yöntemde, toplama listesi ilk sipariş toplayıcıya gönderilir. İlk bölgedeki sipariş toplayıcı kendisinden sonraki bölgedeki sipariş toplayıcı boşalıncaya kadar malzeme toplamaya devam eder. Kendisinden sonraki toplayıcı boşalınca toplama listesini ona teslim eder. Kendisi ise kaynağa dönerek, yeni toplama listesini alır. Sipariş toplayıcılar kendisinden sonraki toplayıcının işi bitinceye kadar toplama yapmaya devam eder. Bu sayede, sipariş toplayıcıların atıl durumda kalma durumu ortadan kalkmıştır (Bartholdi ve Hackman, 2011).

Dalgasal toplama: Ortak bir hedef için (örneğin, belirli bir seferi olan belli bir taşıyıcının kalkış zamanı için) toplama listelerinin bir veya daha fazla depo bölgesine aynı anda serbest bırakılarak sipariş hazırlama yöntemine dalgasal toplama denir. Genellikle (ancak zorunlu değil) parti toplama ile birleştirilir. Toplam sipariş toplama süresi genellikle 30 dakika ile 2 saat arasında olacak şekilde belirlenir. Tüm malzemeler toplanıncaya kadar yeni bir dalga gönderilmez (Kosner ve diğ., 2007).

2.1.5.5. Sınıflandırma ve Birleştirme Süreci

Sınıflandırma ve birleştirme sürecinde, toplanan tüm malzemeler siparişlere göre sınıflandırılarak ayrılır ve birleştirilir. Sınıflandırma ve birleştirme süreci, sipariş toplama sürecinin devamı niteliğindedir. Söz konusu süreç, sipariş toplama sürecinden etkilenmekte ve sipariş toplama sürecine bağlı olarak değişiklik göstermektedir. Sipariş

toplama yöntemlerinden olan parti toplama, dalgasal toplama ve bazı bölgesel toplama yöntemlerin söz konusu süreçten geçmesi gerekmektedir.

Sınıflandırma sürecinde biriktirme noktasında toplanan malzemeler müşteri siparişlerine göre ayrı ayrı hazırlanır. Toplanan malzemelerin müşteri siparişlerine göre ayırma işlemleri personel tarafından manuel olarak yapılabildiği gibi bilgisayar kontrollü makineler tarafından da otomatik olarak yapılabilir. Toplanan malzemeler, manuel ayırma sürecinde depo personeli yardımıyla müşteri siparişlerine göre ayrılır. Depoda hazırlanacak sipariş miktarları düşük olduğunda manuel ayırma tercih edilir. Bilgisayar kontrollü ayırma işlemlerinde konveyör sistemleri ve otomatik tanımlama sistemleri kullanılarak siparişler ayrılır. Ayrılacak sipariş sayısı yükseldikçe bilgisayar kontrollü ayırma işlemleri tercih edilir.

Birleştirme süreci, siparişlerin taşıma paketleri üzerine yerleştirilmesi sürecidir. Malzemelerin yerleştirileceği taşıma paketlerine bağlı olarak birleştirme süreci değişmektedir. Taşıma paketinin seçimi içine yerleştirilecek malzemelerin boyutlarına ve çeşitliliğine göre değişmektedir. Bunun yanı sıra, taşıma paketinin sayısına ve boyutlarına bağlı olarak da taşıma paketinin hazırlanmasında kullanılan yöntem değişmektedir. Taşıma paketlerine malzeme yerleştirilmesi manuel, mekanik ve otomatik olmak üzere üç farklı şekilde yapılmaktadır.

2.1.5.6. Sevkiyat Süreci

Sınıflandırma ve birleştirme sürecinden çıkan hazırlanmış siparişler sevkiyat alanında toplanır. Sevkiyat süreci, hazırlanan siparişlerdeki malzemelerin son kez kontrol edildiği, gerekli evrakların hazırlandığı ve paketlenmiş malzemelerin taşıma araçlarına yerleştirildiği süreçtir.

Sevkiyat sürecinde üç operasyon yapılmaktadır. Bu operasyonlar; hazırlanan siparişlerin doğrulanması, sevkiyat evraklarının hazırlanması ve siparişlerin taşıma araçlarına elleçleme ekipmanları ile yerleştirilmesi operasyonlarıdır. Sipariş doğrulaması müşteri siparişlerinin tam ve eksiksiz olduğunun kontrolüdür. Diğer bir deyişle, sipariş doğrulamasında siparişlerin nitelik ve nicelik kontrolleri yapılır. Sevkiyat evraklarının hazırlanması ise, malzemelerin taşınması sırasında yasal olarak taşıyıcı araçlarda bulundurulması zorunlu tüm evrakların ve dokümanların hazırlanması operasyonudur.

Son operasyon olan siparişlerin taşıma araçlarına elleçleme ekipmanları ile yerleştirilmesinde, tüm hazırlanmış siparişlerin personel veya elleçleme ekipmanları yardımıyla taşıma araçlarına yüklenmesi sağlanır.

Sevkiyat sürecinde malzeme akışının kolaylaştırılmasında ve hızlandırılmasında birtakım prensipler bulunmaktadır. Bu prensipler aşağıda açıklanmaktadır (Tompkins ve diğ., 2010).

- **Maliyete ve hacme uygun elleçleme birimlerinin seçilmesi:** Ambalajlama için geri dönüştürülebilen veya tekrar kullanılabilen taşıma paketleri (birimleri) tercih edilmesiyle hem maliyet azalması sağlanır hem de çevreye karşı duyarlı olunur.
- **Malzeme hasarlarının azaltılması:** Malzemelerin elleçlenmesi ve taşınması sırasında malzeme hasarlarının azaltılması için koruyucu malzemelerin kullanılması faydalı olacaktır. Taşıma paketlerinin titreşimden ve çarpmadan etkilenmemesi için, bu paketler paket içerisine ve çevresine koruyucu malzemeler kullanılarak taşınmalıdır.
- **Sevkiyat aşamasının atlanması:** Sevkiyatın iyi planlamasıyla depodan çıkan malzemeler direkt olarak taşıma araçlarına yüklenir. Bu sayede, malzemeler daha az elleçlendiği için iş gücüne olan ihtiyaç azalır. Bunlara ek olarak, malzemenin taşınması ve elleçlenmesi sırasında oluşabilecek hasarlar azaltılır.
- **Sevkiyat alanı ihtiyacının azaltılması:** Depodaki bazı rafların sevkiyat için kullanılmasıyla sevkiyat alanı ihtiyacı azaltılır. Sevk edilecek malzemelerin raflarda saklanmasıyla hem sevkiyat için ayrılan alan küçülür, hem de sevkiyat alanındaki malzeme yoğunluğu azaltıldığı için malzeme elleçleme kolaylaşır.

2.1.5.7. Çapraz Sevkiyat Süreci

Çapraz sevkiyat, depoya gelen malzemelerin az stoklanarak veya hiç stoklanmadan paketleme, ambalajlama, etiketleme gibi gerekli olan operasyonlardan geçirilerek taşıma araçlarına yüklendiği sevkiyat sürecidir (Choy ve diğ., 2012). Bu fonksiyon sayesinde depolar malzeme stoklama noktası olma özelliğinin yerine, malzeme akışı koordinasyonu noktası olma özelliği elde ederler (Simchi-Levi ve diğ., 2003).

Çapraz sevkiyat operasyonunda sadece teslim alma ve sevkiyat fonksiyonları bulunmaktadır. Çapraz sevkiyat yerleştirme, depolama, malzeme besleme, sipariş toplama, sınıflandırma ve birleştirme, sevkiyata hazırlama ve paketleme süreçlerini ortadan kaldırmaktadır. Bu süreçlerin ortadan kaldırılması ile önemli tasarruflar sağlanmaktadır (Ertek, 2010). Depoya gelen malzemelerin direkt olarak sevkiyata hazır halde getirildiği ve neredeyse malzemeler hiç stoklanmadığı için stok miktarlarında azalma sağlanır. Stok miktarlarının azalması ile depolama ve elleçleme ile ilgili olan maliyetler de azaltılmış olur. Ayrıca, daha az stok tutulması ile depo kapasitesi ve bununla ilgili diğer maliyetler de azaltılır. Depolama veya malzemelerin aktarılması sırasında ortaya çıkabilecek hasar ve kayıp da azaltılmış olur.

Çapraz sevkiyatın sağladığı faydalardan yararlanılabilinmesi için çapraz sevkiyat ile ilgili aşağıda yer almakta olan koşullara dikkat edilmesi gerekmektedir. (Napolitano, 2000).

- **Ortaklık gereklilikleri:** Çapraz sevkiyat, tedarik zincirinde bulunan tüm elemanlar tarafından desteklenmeli ve sürekli takip edilmelidir.
- **Tedarik zinciri elemanları arası kusursuz iletişim:** Çapraz sevkiyat operasyonunun başarılı şekilde yapılabilmesi için tedarik zincirindeki tüm elemanlar arasında bilginin aktarımı kusursuz olmalıdır. Bu durumu sağlayabilmek için, şirketler arası entegre edilmiş bilgi teknolojilerinden faydalanılması ve bu ekipmanları kullanabilen iyi eğitilmiş personelin istihdam edilmesi gerekmektedir.
- **Operasyonlardaki karmaşıklığın doğru yönetilmesi:** Çapraz sevkiyatın tedarik zincirine en büyük katkılarından birisi stok miktarlarını düşürmesidir. Ancak saklanan malzemelerin miktarlarının azaltılmasından ve hatta stoklanmamasından doğan malzeme yetersizliği durumunun ortaya çıkmaması için, tedarik zincirinde malzeme akışının çok iyi şekilde planlanması ve kontrol edilmesi gerekmektedir. Bunun yanı sıra, tüm tedarik zinciri boyunca bu planlar doğrultusunda hareket edilmesi sağlanmalıdır.
- **Çapraz sevkiyatın maliyetlerinin, kazanımlarının ve risklerinin paylaşılması:** Çapraz sevkiyat ile ortaya çıkan tüm maliyetlerin, kazanımların

ve risklerin sürdürülebilir bir hale gelebilmesi için tedarik zincirindeki tüm paydaşlar ile paylaşılması gerekmektedir.

- **Kaynak kullanımları üzerinde uzlaşma sağlanması:** Çapraz sevkiyatta planlamalar, tedarik zincirindeki tüm elemanların kapasite kullanım oranlarını arttırmayı sağlamalıdır. Kapasitenin yetersiz kaldığı yoğun zamanlarda yoğunluğun azaltılması için, teslimatların kapasitenin atıl kaldığı zamanlara doğru kaydırılarak kaynak dengeleme yapılmalıdır. Kaynak dengeleme için zaman çizelgelerinin hazırlanması ve bu çizelgeye tedarik zincirinin tüm elemanlarının uyması gerekmektedir.
- **Mükemmel kalite gereklilikleri:** Çapraz sevkiyatın gerçekleştirilebilmesi için tedarik edilen hammaddelerin, ara montaj parçalarının ve bitmiş ürünlerin kalitesinde herhangi bir şüphe olmamalıdır. Tedarikçilerin sürekli kaliteli malzeme temin etmeleri gerekmektedir. Bu sayede nitelik ve nicelik kontrolü yapılmasına gerek kalmaz; hem hızlı teslim alma sağlanır, hem de kalite kontrol için harcanan zaman ve iş gücü azaltılır.

Bu bölümde çapraz sevkiyat ile birlikte depolardaki tüm süreçler irdelenmiştir. Depo süreçlerinde yapılan tüm işlemler ve süreçlerde kullanılan yöntemler açıklanmıştır. Bunun yanı sıra, daha iyi bir depolama süreci elde edilebilmesi için dikkat edilmesi gereken noktalar açıklanarak; sürecin gelişmesini sağlayacak prensipler verilmiştir.

2.2. DEPO YERLEŞİM DÜZENLEMESİ

Bu bölümde depo yerleşim düzenlemesinin önemi ve depo yerleşim düzenlemesinin aşamaları üzerinde durulacaktır. Ayrıca, depo yerleşim düzenlemesi tasarımları ve bu tasarımlar ile ilgili detaylı bilgi verilecektir. Bu bölümün son kısmında ise, depo yerleşim düzenlemesi ile ilgili literatürde yer alan çalışmalar detaylı şekilde anlatılacaktır.

2.2.1. Depo Yerleşim Düzenlemesinin Önemi

Depo yerleşim düzenlemesinin öneminin daha iyi anlaşılabilmesi için depo yerleşim düzenlemesini de içine alan depo tasarımı ve tasarımın önemi üzerinde durulması gerekmektedir. Depo yerleşim düzenlemesi yapılırken depo tasarımında kullanılan tüm aşamalardan yararlanılmaktadır.

Doktora tezinin “Depo ve Depolama” başlıklı bölümünde açıklandığı gibi depolar, tedarik zincirinin ve lojistik yönetiminin vazgeçilmez bir öğesidir. Karşılıklı malzeme akışlarının (tedarikçiden müşteriye ve müşteriden tedarikçiye) ve bu akışlar ile ilgili tüm bilginin entegre ve eş zamanlı sağlanabilmesi için önemli yere sahip olan depoların, bu akışların planlanmasına, yönetilmesine ve kontrolüne en üst düzeyde katkı sağlaması gerekmektedir. Bu önemli öğenin tedarik zinciri içerisindeki malzeme ve bilgi akışı üzerinde direkt etkisi bulunmaktadır. Bu etkiden olumlu şekilde yararlanabilmek için, depoların aşağıdaki hedeflere ulaşması beklenmektedir (Queirolo ve diğ., 2002).

- Depolama alanının kullanımını enbüyüklemek
- Elleçleme ekipmanlarının kullanımını enbüyüklemek
- Tüm malzemelere erişilebilirliği enbüyüklemek
- Emeğin kullanımını enbüyüklemek
- Tüm malzemelerin korunmasını enbüyüklemek

Yukarıda sayılan hedeflere ulaşılabilinmesinde depo tasarımı önemli rol oynamaktadır. Depo tasarımı, deponun amacına ve türüne bağlı olarak deponun tüm fonksiyonlarını dikkate alarak; depo boyutlarının belirlenmesi, raf sistemlerinin seçilmesi, rafların uzunluklarının, yüksekliklerinin ve derinliklerinin belirlenmesi, malzeme elleçleme ekipmanının seçimi, depolama politikasının seçimi, yerleşim planının hazırlanması, yükleme/boşaltma için kullanılacak yerlerin belirlenmesi ve sipariş toplama yöntemlerinin tespiti gibi birçok unsuru içinde barındıran bir süreçtir. Bunlara ek olarak, deponun tüm fonksiyonlarının daha iyi yönetilmesi ve kontrol edilmesi için kullanılacak depo bilgi sisteminin de bu süreç içerisinde düşünülmesi gerekmektedir. Depo tasarımı yüksek karmaşıklık içeren bir süreçtir. Bu sürecin zor ve karmaşık olmasının temel nedeni ise yukarıda sayılmakta olan unsurların amaçlarının süreç içerisinde birbiri ile çelişiyor olması ve sürecin birçok alternatife sahip olmasıdır. Tasarım sırasında verilmesi gereken birçok karar birbiri ile ilişkili ve entegre bir şekilde verilmelidir. Bunun yanı sıra, her bir karar aşamasının deponun amacına ve türüne bağlı olarak performans kriterlerinin belirlenmesi ve bu kriterlere göre hareket edilmesi gerekmektedir.

İyi ve doğru planlanmış bir depoya sahip olmak için depo tasarımı süreci aşamalarında yer alan tüm karar noktaları detaylı şekilde tespit edilmelidir. Depo tasarımı yapılırken

süreçte kullanılacak sistemlerin birbirleri ile uyumunun sağlanması ile, depolardan beklenen ihtiyaçların karşılanması kolaylaşacaktır. Başarılı bir depo tasarımı sayesinde, depolardan çıkan malzeme miktarının yükselmesi, sipariş toplama sürelerinin ve mesafelerinin kısalması, stok kullanım oranlarının yükselmesi gibi birçok kriterde yüksek performans elde edilir.

Gu ve diğ., (2010) depo tasarımı üzerine yapılan çalışmaları incelediğinde, depo tasarımının aşağıdaki beş temel kararı içerdiğini belirtmiştir. Bunlar depo genel yapısının belirlenmesi, kapasitenin belirlenmesi ve boyutlandırılması, depo bölümleri yerleşiminin belirlenmesi, ekipman seçimi ve operasyon stratejisi seçimidir (Gu ve diğ., 2010).

Depo genel yapısının belirlenmesi: Fonksiyonel departmanlar, depolama alanları, kullanılacak teknolojiler ve sipariş toplama yöntemleri gibi tüm kararların verildiği aşamadır. Depo genel yapısının belirlenmesi tasarımcıların ve şirketlerin stratejik kararlarındandır. Bu aşamada daha çok malzeme akışının ve bilgi akışının yönetilmesi ile ilgili kararlar verilir. Bunların yanı sıra, depolardan beklenen stok ve çıkış miktarları ile bina ve operasyon maliyetleri gibi performans kriterleri göz önünde bulundurulur. Depoların genel işleyişi ile ilgili tüm kararları içermektedir.

Depo bölümleri yerleşiminin belirlenmesi: Depo içindeki stoklama bölümlerinin ve stoklama alanlarının yerleşimlerinin belirlenmesi ile ilgilidir. Bu karar aşamasının, zemin stoklama yerleşimi, depolama alanları yerleşimi ve otomatik yerleştiren/alan sistem yerleşimi olmak üzere üç alt kısmı bulunmaktadır. Zemin stoklama yerleşiminin belirlenmesinde; zeminde stoklanacak alanları, zemin stoklama alanlarındaki sıraların uzunlukları, sıra yükseklikleri ve sıra sayıları ile ilgili kararlar verilir. Zemin stoklama yerleşimi belirlenmesinde stok yoğunluğu, stok kapasite miktarı, boşluk oluşumu ve ulaşılabilirlik gibi performans kriterleri dikkate alınmaktadır. Depolama alanı yerleşiminin belirlenmesi ise; depolama ve malzeme elleçleme ile ilgili maliyetleri azaltmaya yarayan depo yerleşim düzenlemesinin belirlenmesidir. Doktora tezi kapsamında depo yerleşim düzenlemesi probleminin çözümüne ilişkin yapılan çalışmada sözü edilen depo yerleşim düzenlemesi, burada bahsedilmekte olan depolama alanı yerleşiminin belirlenmesidir. Depo yerleşim düzenlemesi; raf sisteminde kaç koridor olacağı, raf uzunluğunun ve yüksekliklerinin ne olacağı, depo teslim

alma/sevkiyat için kaç adet kapı kullanılacağı ve bu kapıların nasıl konumlandırılacağı gibi sorulara cevaplar bulma amacını içermektedir. Depo yerleşim düzenlemesi yapılırken; ortalama toplayıcı mesafesi, toplam toplayıcı mesafesi ve bunların maliyetleri ile inşaat ve elleçleme maliyetleri gibi değerlendirme kriterleri üzerinde karar verilmeye çalışılır. Depo bölümleri yerleşiminin belirlenmesinin son kısmı olan otomatik yerleştiren/alan sistem yerleşiminin belirlenmesi; raf uzunlukları ve yükseklikleri, toplama aracı sayısı ve seçimi gibi kararları içermektedir. Bu son aşamada da, depo yerleşim düzenlemesi aşamasında dikkat edilmesi gereken kriterler göz önüne alınmalıdır.

Kapasitenin belirlenmesi ve boyutlandırılması: Bu aşama kapasite belirleme ve boyutlandırma kararları olmak üzere iki bölüme ayrılır. Kapasite belirlemede stok seviyelerinin ve stok politikalarının belirlenmesi gerekmektedir. Depolama maliyeti, inşaat maliyeti veya besleme maliyeti gibi maliyetler, stoklanacak malzemelerin stok miktarlarının veya dönemsel stoklama kapasitelerinin belirlenmesinde etkilidir. Depo boyutlandırma ise, inşaat ve operasyon maliyetine göre kapasitenin alanlara ayrılmasıdır.

Ekipman seçimi: Ekipman seçimi aşamasında kullanılacak raf sistemi (sırt sırta, çift derinlikli, akış tipi raf sistemi vb.) ve elleçleme ekipmanının belirlenmesi ve belirlenen ekipmanların sayıları gibi konular üzerinde durulmaktadır. Ekipman seçimi depo genel yapısının belirlenmesi sırasında alınması gereken stratejik bir karardır. Ekipman seçimi yaparken; birden fazla ekipman ve raf alternatifinin arasından depo genel yapısı sırasında belirlenen performans kriterleri göz önüne alınmalıdır.

Operasyon stratejisi seçimi: Operasyon stratejisi seçiminde stoklama politikası ve sipariş toplama yöntemi belirlenmektedir. Stoklama politikasının belirlenmesi, tasarlanacak deponun malzeme yerleşimi kararının depolama politikasına göre belirlendiği aşamadır. Stoklama politikasında depo tasarımı yapılırken; rassal depolama, sabit depolama, ABC sınıflandırma depolama gibi politikalardan hangisinin tercih edileceği belirlenmektedir. Sipariş toplama yöntemi belirlenmesi aşaması ise; hangi sipariş toplama yönteminin seçilmesi gerektiğine ve seçilen yönteme göre yapılması gereken operasyonların belirlenmesine yöneliktir.

Bu çerçevede gerçekleştirilen doktora tezi, depo tasarımı ile ilgili kararların verilmesinde yer alan problemlerin birçoğuna cevap verir niteliktedir. Bu noktada doktora tezinde cevap verilmesi gereken ilk problem, depo bölümleri yerleşimi belirlenmesine ilişkin kararlar ile ilgilidir. Depo bölümleri yerleşimi belirlenmesine ilişkin kararlar verilirken, yalnızca raf uzunlukları, raf yükseklikleri ve raf sayısı üzerinde durulacaktır. Doktora tezi kapsamında depo bölümleri yerleşimi belirlenmesine ilişkin verilecek kararlar; kapasitenin belirlenmesi ve boyutlandırılması, ekipman seçimi ve operasyon stratejisi seçimi konularında verilecek kararları da etkileyecektir.

İkinci olarak, kapasitenin belirlenmesi ve boyutlandırılması ile ilgili kararlar verilirken, depoda saklanacak malzemelerin ve malzeme gruplarının miktarlarının belirlenmesi gerekmektedir. Malzeme gruplarının sayısı ve kapasiteleri bilinmediğinde raf yükseklikleri, raf uzunlukları ve raf sayıları belirlenememektedir. Malzeme gruplarının sayısı ve kapasiteleri, sayısal yöntemler kullanılarak hesaplanabildiği gibi depo tasarımcılarının veya karar vericilerin deneyimlerinden de faydalanılarak belirlenebilir. Doktora tezi kapsamında, malzeme sayıları ve kapasiteleri depo tasarımcıları tarafından önerilmiştir.

Ekipman seçiminde, depo yerleşim düzenlemesinde kullanılacak raf sistemi ve elleçleme ekipmanları belirlenir. Depo yerleşim düzenlemesinin en iyi şekilde tasarlanabilmesi için, raf sistemi ve elleçleme ekipmanları arasındaki uyumun üst seviyeye çıkarılması gerekmektedir. Doktora tezi kapsamında, depo yerleşim düzenlemesi probleminde belirlenen raf sistemi, sırt sırta raf sistemidir. Elleçleme ekipmanı olarak da, dar koridor forklifti seçilmiştir.

Depo yerleşim düzenlemesi probleminin matematiksel olarak modellenmesi, operasyon stratejisi seçimine de bağlıdır. Operasyon stratejisi seçimi, stoklanacak malzemelerin yerleşim lokasyonlarının belirlenmesinde kullanılan depolama politikası seçimidir. Doktora tezi kapsamında, depo yerleşim düzenlemesi probleminde belirlenen operasyon stratejisi, ABC sınıflandırma politikasıdır.

2.2.2. Depo Yerleşim Düzenlemesinin Aşamaları

Depo yerleşim düzenlemesi aşamaları, depo yerleşim düzenlemesinde takip edilmesi gereken aşamalar dizisini ve her bir aşamada yapılması gereken süreçleri göstermektedir. Depo yerleşim düzenlemesinin her bir aşamasında belirlenmesi gereken birçok süreç bulunmaktadır. Depo yerleşim düzenlemesinin aşamalarında süreçler çözümlenirken hem karmaşık matematiksel yöntemler kullanılır, hem de karar vericilerin veya depo tasarımcılarının tecrübelerinden ve deneyimlerinden faydalanılır.

Depo yerleşim düzenlemesi aşamaları için literatürde ortak karar verilmiş aşamalar dizisi bulunmamaktadır. Ancak, en detaylı aşamalar dizisi Baker ve Canessa (2009) tarafından hazırlanmıştır. Söz konusu çalışma, depo tasarım kullanılan depo tasarım aşamalarını göstermektedir. Ancak, Baker ve Canessa (2009) tarafından önerilen depo tasarım aşamaları, depo yerleşim düzenlemesinin aşamaları için de rahatlıkla kullanılabilir. Baker ve Canessa (2009) tarafından önerilen depo tasarımı aşamaları aşağıda detaylı şekilde açıklanmıştır.

Sistem gereksinimlerinin tanımlanması: Sistem gereksinimlerinin tanımlanması aşaması; tasarlanacak deponun amacının, deponun tedarik zincirindeki konumunun ve malzeme akışındaki rolünün, depo tipinin, deponun müşteri hizmet düzeyinin, depodan beklenilen çıkış miktarının ve buna benzer gereksinimlerin belirlendiği aşamadır. Sistem gereksinimleri tanımlanırken yasal düzenlemeler ve hukuksal durumlarda göz önünde bulundurulması gerekmektedir.

Verilerin tanımlanması ve toplanması: Verilerin tanımlanması ve toplanması aşaması, tüm sistem gereksinimleri tanımlandıktan sonra söz konusu gereksinimler için gerekli olan verilerin tanımlandığı ve toplandığı aşamadır. Verilerin tanımlanması ve toplanması aşaması, istenilen verilerin özelliklerine göre toplanması zor ve uzun zaman alabilmektedir. Geçmiş verilerin toplanmasının yanı sıra gelecek planlarının da göz önünde bulundurularak verilerin toplanması gerekmektedir.

Veri analizi: Veri analizi aşaması, tanımlanmış ve toplanmış verilerin istenilen bilgiye dönüştürülmesi için analiz edildiği aşamadır. Veri analizi aşamasında, veri analizi için kullanılan bilgisayar programlarından ve uzman kişilerden destek alınarak yapılır. İyi planlanmış depo tasarımı yapılabilmesi için birtakım istatistikî analizler yapılması

gerekmektedir. Bu istatistiki analizler, aşağıda maddeler halinde açıklanmıştır (Frazelle, 2002).

- Müşteri siparişi profili çıkarma (Sipariş karışımı dağılımı, siparişteki malzeme çeşitliliği dağılımı, sipariş hacmi dağılımı, malzeme çeşitliliği ve hacmi dağılımı)
- Malzeme operasyonları profili çıkarma (ABC analizi, hacim dağılımı, ABC-hacim dağılımı, sipariş tamamlama dağılımı, talep korelasyon dağılımı, talep çeşitliliği dağılımı)
- Envanter profili çıkarma (Aile bazlı stok dağılımı ve malzeme elleçleme birimi bazlı stok dağılımı)
- Takvim-saat profili çıkarma (Sezonluk malzeme dağılımı, günlük iş dağılımı)
- Operasyon ilişkileri profili çıkarma
- Yatırım profili çıkarma (Frazelle, 2002)

Stok tutma birimi belirlenmesi: Stok tutma birimi belirlenmesi aşaması, tedarik zincirindeki her bir elemanın mal akışında kullanacağı stok tutma birimlerinin belirlendiği aşamadır. Stok tutma birimleri, tasarlanacak depoya göre belirlenmemeli, tedarik zincirindeki diğer elemanlarda düşünülerek karar verilmelidir(Baker ve Canessa, 2009). Stok tutma biriminin belirlenmesi ile stok tutma birimleri için ayrılacak alanların da boyutları belirlenir.

Çalışma prosedürünün ve yöntemlerinin belirlemesi: Çalışma prosedürünün ve yöntemlerinin belirlemesi aşamasında, doktora tezinin ikinci bölümünde yer alan depolamadaki temel süreçler bölümü içerisindeki yerleştirme süreci, depolama süreci ve sipariş toplama süreci içerisinde anlatılan çalışma prosedürleri ve yöntemleri belirlenmektedir. Seçilen prosedürlere ve yöntemlere göre depolarda ayrılması gereken depolama bölgeleri büyüklükleri ve yerleri, malzemenin depoya yerleşimi, sipariş toplama yöntemi, sipariş toplama rotaları, toplanan malzemelerin ayırma ve birleştirme alanlarının büyüklükleri gibi birtakım kararlar verilir.

Olası ekipman türlerine ve özelliklerine karar verilmesi: Olası ekipman türlerine ve özelliklerine karar verilmesi aşaması, depo içerisinde kullanılacak raf sistemlerinin ve elleçleme ekipmanlarının türlerine ve özelliklerine karar verildiği aşamadır. Manuel

elleçleme ekipmanları alternatiflerinden, tam otomatik elleçleme ekipmanları alternatiflerine kadar malzeme hareketi için kullanılan birçok ekipman arasından seçim yapılmaktadır. Seçilen malzeme elleçleme ekipmanı, birçok raf sistemi alternatifi içerisinden seçilen raf sistemi ile uyum içerisinde olması gerekmektedir. Hem raf sistemi hem de elleçleme ekipmanları, çalışma prosedürü ve yöntemlerinin tüm gereksinimlerine yanıt verebilmelidir. Bunlara ek olarak, depoda kullanılacak bilgi teknolojilerinin seçimi ve bu sistemin diğer sistemler ile entegrasyonu da olası ekipman türlerine ve özelliklerine karar verilmesi aşamasında yapılmalıdır.

Ekipman kapasitelerinin ve miktarlarının hesaplanması: Ekipman kapasitelerinin ve miktarlarının hesaplanması aşamasında, belirlenmiş çalışma prosedürleri ve yöntemleri ile olası ekipman türleri ve özellikleri ile bağlantılı olan ekipman kapasiteleri ve miktarları hesaplanır. Ekipman kapasitelerini ve miktarlarını hesaplamak için depo içerisindeki malzemenin akışı, sezonluk değişimler, günün yoğunluk zamanları gibi birçok bilgiye ihtiyaç duyulmaktadır (Rushton ve diğ., 2010). Ekipman kapasitelerinin ve miktarının hesaplanmasında, basit matematiksel yöntemler kullanıldığı gibi daha karmaşık ve zaman alan simülasyon programları da hazırlanarak kapasiteler ve miktarlar belirlenebilir.

Hizmetlerin ve yardımcı işlemlerin tanımlanması: Hizmetlerin ve yardımcı işlemlerin tanımlanması aşaması, deponun amacı doğrultusunda depodan verilmesi gereken hizmetlerin ve yardımcı işlemlerin tanımlandığı aşamadır. Bu hizmetlerin ve yardımcı işlemlerin içerisinden en önemlisi katma değerli operasyonlardır. Katma değerli operasyonların neler olacağı, nerelerde yapılacağı ve nasıl bir süreç izleyeceği gibi birçok sorunun tanımlandığı ve çözüm üretildiği aşamadır.

Alternatif yerleşim düzenlenmesi: Alternatif yerleşim düzenlenmesi aşamasına kadar depolarda kullanılacak elleçleme ekipmanlarının belirlenmesi, raf sisteminin belirlenmesi ve çalışma prosedürü ve yöntemleri gibi birçok aşamadan geçilerek kararlar verilmiştir. Alternatif yerleşim düzenlenmesi aşamasında kadar olan tüm kararlarda tek bir doğru çözüm kararı bulunmamakta ve her bir karar aşamasında alternatif çözümler bulunabilmektedir. Alternatif bir raf sistemi, alternatif bir depolama yöntemi, alternatif sipariş toplama rotası, alternatif bir elleçleme ekipmanı ve buna

benzer alternatif ekipman ve süreçler kullanılarak alternatif depo yerleşimi düzenlemesi elde etmekte mümkündür.

Değerlendirme ve karşılaştırma: Değerlendirme ve karşılaştırma aşaması, alternatif yerleşim düzenlenmesi aşamasından elde edilen alternatif depo yerleşim düzenlemelerinin farklı performans kriterleri altında değerlendirildiği ve karşılaştırıldığı aşamadır. Önerilen alternatif depo yerleşim düzenlemesi çözümleri, operasyonel ve teknik fizibilite doğrulama, sermaye ve operasyonel maliyetleri gibi birtakım kriterlere göre değerlendirilir ve karşılaştırılır. Alternatiflerin değerlendirmesi ve karşılaştırması için birçok yöntem bulunmasına rağmen en çok tercih edilen simülasyon yöntemidir (Baker ve Canessa, 2009).

En uygun tasarımın seçilmesi: En uygun tasarımın seçilmesi aşaması, depo tasarımın veya depo yerleşim düzenlemesinin son aşamasıdır. Söz konusu aşama, birçok değerlendirme kriteri altında değerlendirilen ve karşılaştırılan alternatiflerden sistem gereksinimleri aşamasında belirlenen gereksinimlerine en uygun depo yerleşim düzenlemesinin seçildiği aşamadır.

2.2.3. Depo Yerleşim Düzenlemesi Tasarımları

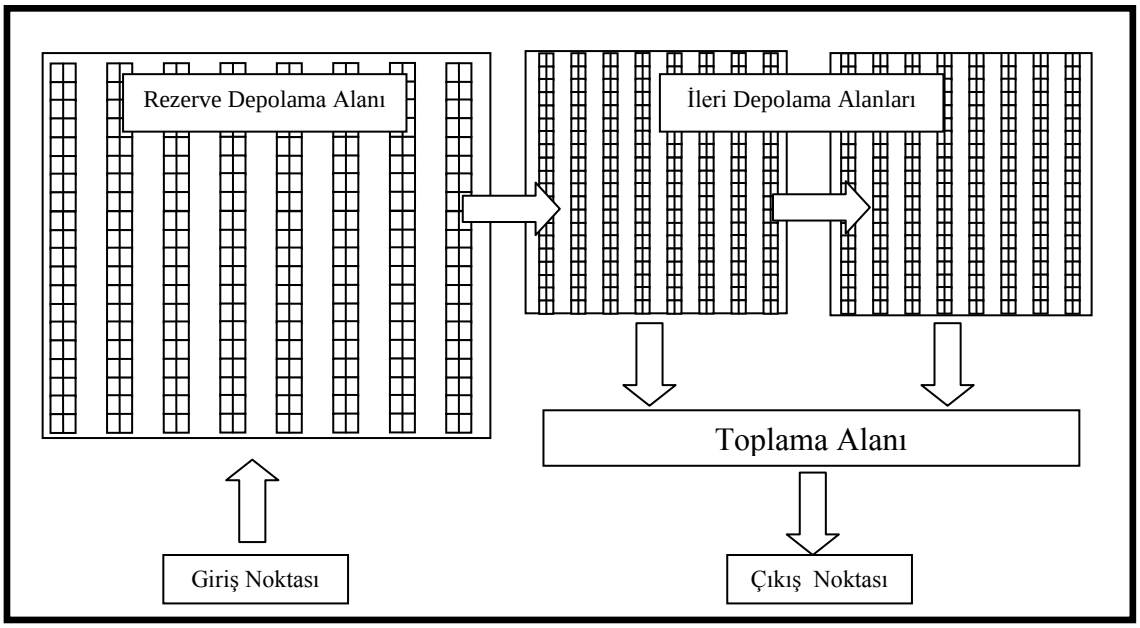
Bu bölümde, literatürde karşılaşılan depo yerleşim düzenlemesi tasarımları ve söz konusu tasarımların özellikleri üzerinde durulacaktır. Depo yerleşim düzenlemesi tasarımları, geleneksel tasarımlar ve modern tasarımlar olarak iki bölümde incelenecektir.

2.2.3.1. Geleneksel Depo Yerleşim Düzenlemesi Tasarımları

Geleneksel depo yerleşim düzenlemesi tasarımları, endüstride ve literatürde en çok kullanılan tasarımlardır. Bu tasarımlar U-şekilli depo yerleşim tasarımı, düzgün akış depo yerleşim tasarımı ve I-şekilli depo yerleşim tasarımı olarak adlandırılmaktadır.

U-şekilli depo yerleşim tasarımı: U-şekilli depo yerleşim tasarımında rezerve depolama alanı ve ileri stoklama alanı olmak üzere iki farklı depolama alanı mevcuttur. Bu depo yerleşim tasarımının "U şekilli" adlandırılmasının nedeni, depo içerisinde malzeme akışının U şeklinde gerçekleşmesidir. Rezerve depolama alanı, birbirine paralel birden fazla raftan oluşan ve bu raflara genellikle paletler halinde malzemelerin depolandığı alandır. Birbirine paralel raflardan oluşan ileri depolama alanı ise,

malzemelerin kutular/sandıklar (bozulmuş palet) halinde stoklandığı depolama alanıdır. U-şekilli depo yerleşim tasarımlarında sipariş toplama süreci, daha küçük alan olan ileri depolama alanlarında gerçekleştirilir. Malzeme teslim alma süreci ve malzeme sevkiyat süreci için kullanılan kapılar, deponun ön tarafına yerleştirilmiştir. Ancak, teslim alma süreci ve sevkiyat süreci aynı kapıdan yapılmamaktadır. Her iki süreç için kullanılacak kapılar ayrılmıştır. U-şekilli depo yerleşimi, Şekil 2.2 üzerinde gösterilmiştir (Lambert ve diğ., 1998).

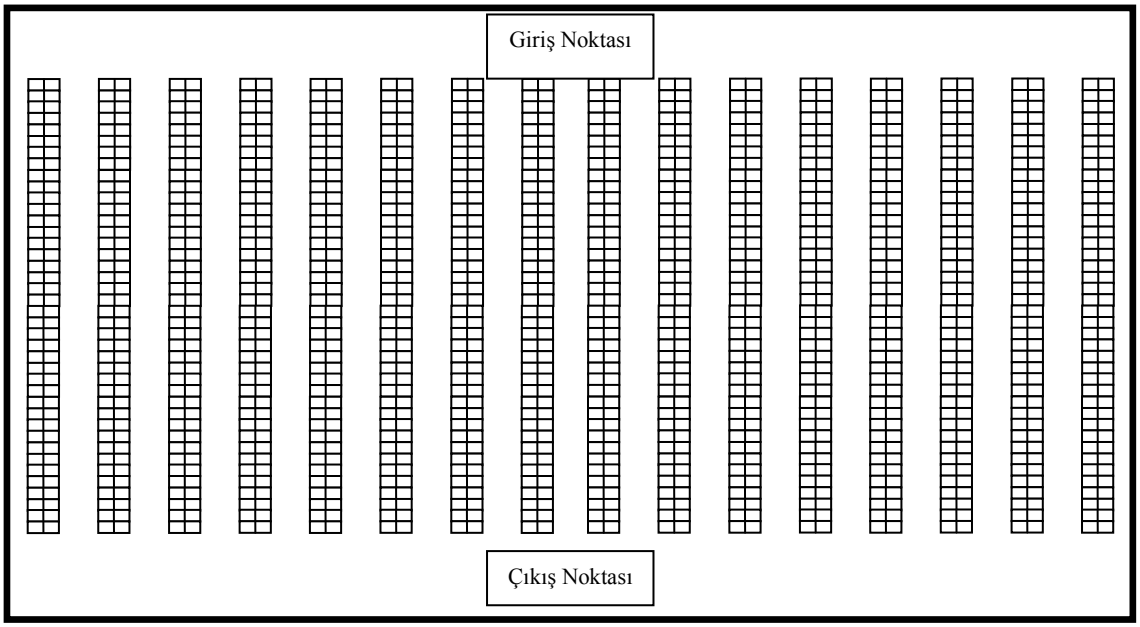


Şekil 2.2: U-Şekilli depo yerleşim tasarımı.

U-şekilli depo tasarımının ileri depolama alanları, birden fazla depolama blokları olacak şekilde ayrılabilir. Blokları ayırma kararı verilirken, istenirse çapraz koridor da ilave edilebilir. Bloklar arasına çapraz koridor eklenmesi ile koridorlar arası geçiş kolaylaşmaktadır. Bu tasarım şekli için farklı alanlara ayrılmadan da depo yerleşimi yapılabilir. Bu durumda aynı raf sistemi deponun tümüne uygulanır. U-şekilli depo yerleşim tasarımının faydaları, yüksek kapı (yükleme ve boşaltma noktası) kullanımı sağlaması ve çapraz sevkiyatı desteklemesidir (Richards, 2003).

Düzgün akış depo yerleşim tasarımı: Düzgün akış depo yerleşim tasarımı, birbirine paralel olarak yerleştirilmiş birçok yatay toplama koridorundan ve dikey iki çapraz koridordan oluşan yerleşim tasarımıdır. Çapraz koridorlardan birisi deponun ön tarafında, diğeri ise deponun arkada tarafındadır. Çapraz koridorlar herhangi bir stok pozisyonu içermemekte ve toplayıcının bir koridordan diğerine geçmesinde

kullanılmaktadır. Depoya gelen malzemeler arkadaki çapraz koridordan teslim alınmaktadır. Depodan çıkan malzemeler ise öndeki çapraz koridordan sevk edilmektedir. Malzeme giriş noktası/noktaları ile malzeme çıkış noktası/noktaları deponun ön ve arka duvarlarında karşılıklı olarak yerleştirilmektedir. Düzgün akış depo yerleşimi, Şekil 2.3 üzerinde gösterilmiştir.



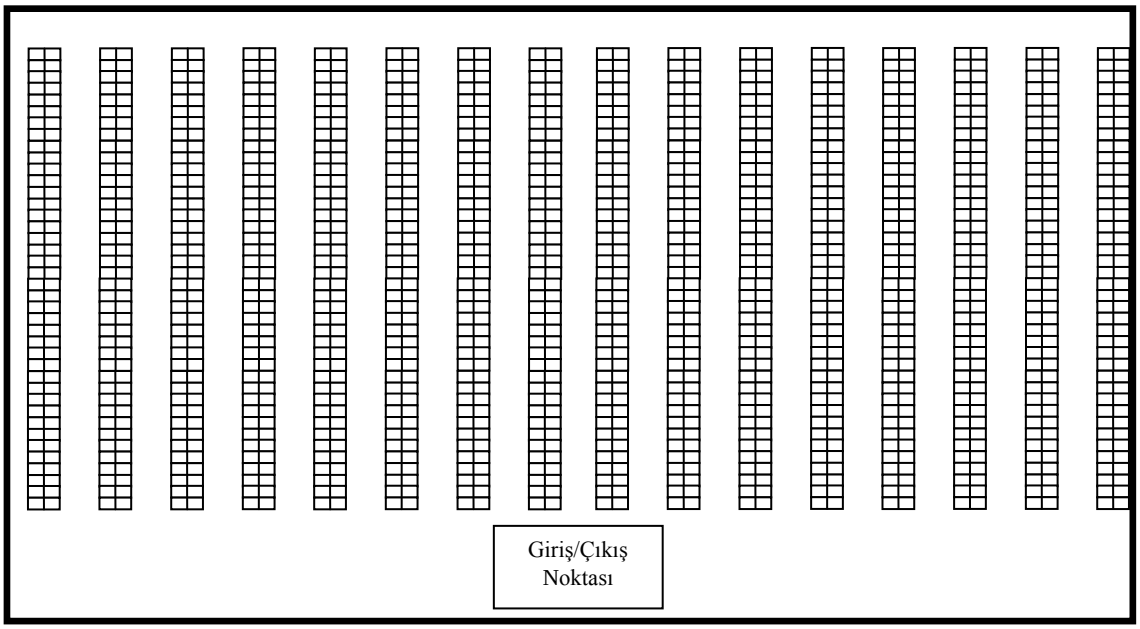
Şekil 2.3: Düzgün akış depo yerleşim tasarımı.

Şekil 2.3'de görselleştirildiği gibi depolama alanı tek bir bloktan oluşmaktadır. Ancak, tek bloklu depolama alanı, çapraz koridorlar ilavesi ile birden fazla depolama bloklarına ayrılabilir. Daha çok çapraz koridor varlığı ile toplayıcının toplama koridorları arası geçişi kolaylaşır. Ancak, her çapraz koridor eklenmesiyle, depolama kapasitesinde azalma gerçekleşir.

Düzgün akış depo yerleşim tasarımında, malzeme teslim alınması ve sevkiyatı farklı depo taraflarında yapıldığından, depo araç sahasında taşıma aracı trafiği olmamaktadır. Bunun yanı sıra, yükleme boşaltma için kullanılan elleçleme ekipmanı trafiği de azaltılmaktadır. Ancak, deponun ön ve arka tarafı için fazladan güvenlik planlaması sağlanması, taşıyıcı araçlar için ise park yerleri ve yolları yapılması gerekmektedir (Richards, 2003).

I-şekilli depo yerleşim tasarımı: I-şekilli depo yerleşim tasarımı, U-şekilli depo yerleşim tasarımına benzemektedir. Her iki tasarımda da malzeme teslim alma ve

malzeme sevkiyatı deponun ön tarafına yerleştirilmiş kapılarından yapılmaktadır. I-şekilli depo yerleşim tasarımında U-şekilli depo yerleşim tasarımından farklı olarak, malzeme teslim alma ve malzeme sevkiyat süreci aynı kapılardan gerçekleşmektedir. I-şekilli depo yerleşimi, Şekil 2.4'te görselleştirilmiştir.



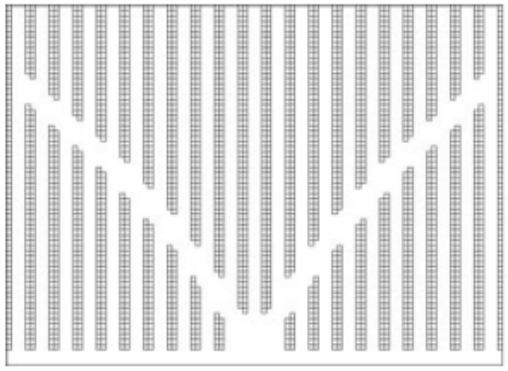
Şekil 2.4: I-şekilli depo yerleşim tasarımı.

Bu depo şeklinin kullanılması ile daha yüksek kapı (yükleme/boşaltma noktası) kullanımı sağlanır ve çapraz sevkiyat desteklenir. Malzeme teslim alınması ve sevk edilmesi süreçleri her kapıdan yapılabildiği için U-şekilli depo tasarımına göre daha az kapı ve rampaya ihtiyaç duyulmaktadır. Teslim alma ve sevkiyat süreçleri için ayrılan alanların kullanım oranları yükselir. Ancak, söz konusu kapıların ve alanların ortak kullanılması ile hem teslim alma hem de sevkiyat sürecinin yönetilmesi zorlaşmaktadır.

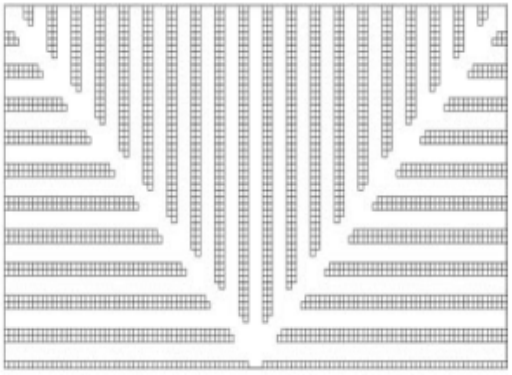
I-şekilli depo yerleşim düzenlemesi tasarımı, aynı diğer tasarımlarda olduğu gibi tek bloktan oluşmayabilir. Birden fazla çapraz koridor tasarlanarak depo stoklama alanı bloklara ayrılabilir.

2.2.3.2. Modern Depo Yerleşim Düzenlemesi Tasarımları

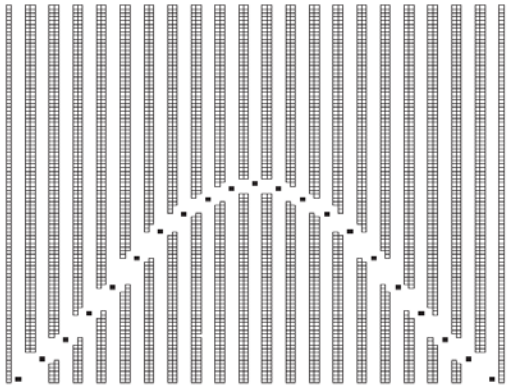
Modern depo yerleşim tasarımları Şekil 2.5'te gösterilmiştir.



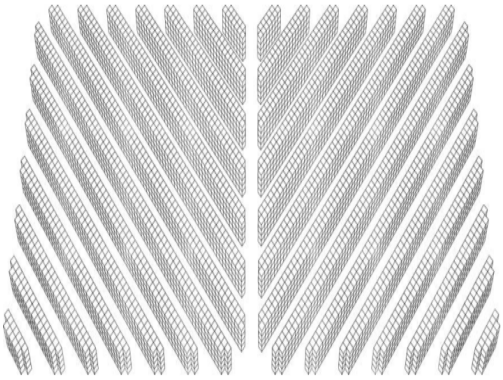
a. V-dizilimi tasarımı (Flying-V)



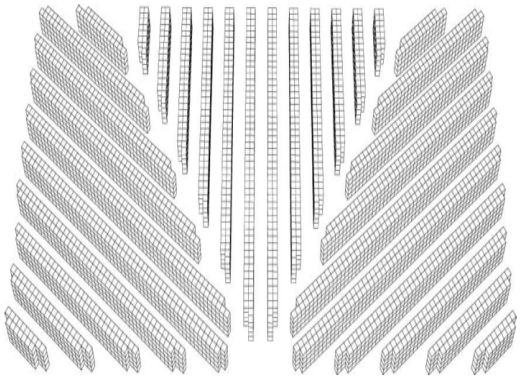
b. Balık kılçığı tasarımı (Fishbone)



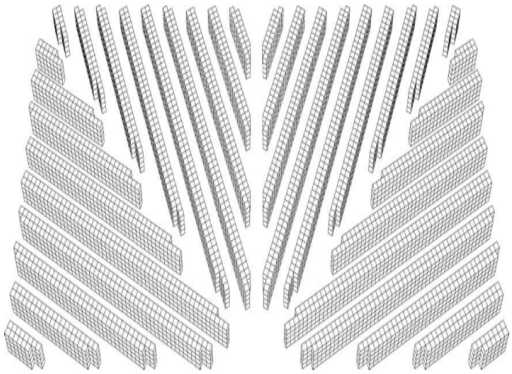
c. Ters V-dizilimi tasarımı (Inverted-V)



d. Zikzak tasarımı (Chevron)



e. Yaprak tasarımı (Leaf)



f. Kelebek tasarımı (Butterfly)

Şekil 2.5: Modern depo yerleşim düzenlemesi tasarımları.

Şekil 2.5'te gösterilen V-dizilimi ve balık kılçığı depo yerleşim düzenlemesi tasarımları Gue ve Meller (2009) tarafından önerilmiştir. Ters V-dizilimi, Gue ve diğ. (2012) tarafından tasarlanmıştır. Zikzak, yaprak ve kelebek depo yerleşim tasarımları ise Öztürkoğlu ve diğ. (2012) tarafından hazırlanmıştır.

Modern depo yerleşim tasarımları, birden fazla çapraz koridor kullanılarak elde edilen tasarımlardır. Çapraz koridorlar, depoya yatay şekilde yerleştirilmek yerine diyagonal olacak şekilde yerleştirilmiştir. Tüm modern tasarımların amacı, raflar arasındaki düz

çizgisel taşıma mesafelerini azaltırken, diyagonal taşıma mesafelerini ise arttırmaktır. Düz çizgisel taşıma mesafelerinin yerine diyagonal taşıma mesafelerinin kullanılması ile ortalama toplayıcı mesafesi azaltılmaktadır. Ancak, modern depo yerleşim tasarımlarında çapraz koridorların köşegen olacak şekilde yerleştirilmesi ile depo kapasitesinde azalma olmakta ve kullanım zorlukları ortaya çıkmaktadır.

2.2.4. Depo Yerleşim Düzenlemesi İle İlgili Çalışmalar

Literatürde depo yerleşim düzenlemesi ile ilgili yapılmış birçok çalışma mevcuttur. Bu çalışmalara aşağıda kısaca yer verilmektedir.

Berry (1968) tarafından depo yerleşiminin temel ihtiyaçları açıklanmıştır. İki farklı depo yerleşimi önerilmiş ve önerilen tasarımlar matematiksel olarak ifade edilmiştir. Ortalama mesafe, ortalama süre ve stok pozisyonu kullanımı ilgili tüm maliyetler dikkate alınarak depo boyutları ve raf düzenlemeleri belirlenmeye çalışılmıştır. Matematiksel modeller karmaşık yapıda olduğu için bilgisayar yardımıyla çözümlenmiştir.

Rosenblatt ve Roll (1984)'a ait çalışmada, depo tasarım problemi matematiksel olarak modellenmiş ve bu modelin optimum sonucunu bulmak için arama prosedürü önerilmiştir. Matematiksel modelde üç tür maliyet dikkate alınmıştır. Bunlar, ilk yatırım (inşaat ve elleçleme tesisleri) maliyeti, kapasite yetersizliği maliyeti ve depolama politikası ile ilgili maliyetlerdir. Optimum depo tasarımını bulan arama prosedürü, analitik optimizasyon ve simülasyon tekniklerini bir araya getirerek geliştirilmiştir. Farklı depo kapasitelerine ve depolama politikalarına sahip depo alternatiflerinin karşılaştırılması ile depo büyüklüğü, depo yerleşimi ve deponun stok politikası elde edilmiştir.

Ashayeri ve Gelders (1985) tarafından depolama tasarımının (depo tasarımı ve depo içi operasyonları içeren) optimizasyonu için çözüm prosedürlerini içeren literatür çalışması yapılmıştır. Ashayeri ve Gelders (1985), çözüm prosedürlerini analitik, simülasyon ve sezgisel olarak üç gruba ayırarak; o zamana kadar yapılan tüm çalışmaları incelemiştir.

Park ve Webster (1989) tarafından üç boyutlu ve palet yerleştirilen depolama sistemleri için optimizasyon prosedürü geliştirilmiştir. Park ve Webster (1989)'in çalışmasında depoya malzeme yerleştirme politikasına, iş emri prosedürüne, elleçleme ekipmanı

tipine ve raflar arası ekipman hareketine bağlı olarak alternatif depo yerleşim düzenlemeleri önerilmiştir. Elleçleme ekipmanının tur süresi, depo alanı gereksinimi ve ortalama tur zamanı ile ilişkili maliyetler göz önüne alınarak alternatiflerin toplam maliyetleri belirlenmiştir. Önerilen tüm alternatif depo yerleşim düzenlemeleri arasından en uygun maliyetli alternatif çözüm olarak tercih edilmiştir.

Gray ve diğ. (1992) çalışmalarında sipariş konsolidasyon deposunun sistem tasarımı probleminin ve depo operasyonları probleminin çözümleri için hiyerarşik karar yaklaşımını önermiştir. Önerilen karar yaklaşımı depo yerleşimi, ekipman ve teknoloji seçimi, malzeme yerleştirme yöntemi, fonksiyonel olarak bölgelere ayırma, toplayıcı rotası, toplama listesi ve sipariş birleştirme gibi problemlere de cevap verebilmektedir. Hiyerarşik karar yaklaşımını kullanarak mevcut depoya uygun alternatif depo tasarımları geliştirilmiş ve bu alternatifler simülasyonla değerlendirilerek uygun depo yerleşim tasarımı belirlenmiştir.

Brito (1992), depo tasarımı için bilgisayar destekli tasarım ile simülasyon tekniğini bir araya getirerek, bir karar destek sistemi yazılımı geliştirmiştir. Geliştirilen karar destek sistemi yazılımı sayesinde farklı depo yerleşim modelleri görselleştirilerek, bu modellerin karşılaştırılması yapılmıştır. Yerleşim modellerinde, koridor sayıları ve koridorlarda çalışacak elleçleme ekipmanları sayıları değiştirilerek denemeler yapılmıştır. Ancak, geliştirilen karar destek sistemi optimal sonucu verecek şekilde tasarlanmamış; önerilen tasarımlar arasından en uygun sonucu veren alternatifin belirlenmesi için tasarlanmıştır.

Dowlatshahi (1994) çalışmasında malzeme elleçleme, tesis yerleşimi ve depolama içeren depolama tesisi sistemini geliştirmek için sistematik ve entegre bir yöntem sunmaktadır. Karmaşık yapıya sahip tesis tasarımı problemi malzeme elleçleme, tesis yerleşimi ve depolama sistemleri başlıkları altında alt sistemlere ayrılmış; ayrılan tüm alt sistemler ayrı ayrı çözümlenerek depolama tesisi geliştirilmiştir. Her bir alt sistemin diğer alt sistemler ile olan ilişkileri de ortaya konmuştur.

Larson ve diğ. (1997), depolama politikalarından birisi olan ABC sınıflandırma politikasına sahip depo için malzeme yerleştirme prosedürü geliştirmiştir. Üç aşamadan oluşan prosedürün ilk aşaması, malzeme akışına bağlı olarak raf yerleşiminin ve

boyutlarının belirlenmesidir. İkinci aşamada saklanacak malzemelerin stok miktarları bulunmuş; son aşamada ise, malzemelerin depodaki raflara atanması yapılmıştır. Son olarak, mevcut sistem ile geliştirilen prosedür kullanılarak elde edilen depo yerleşimleri karşılaştırılmıştır.

Caron ve diğ. (2000a), farklı yükseklikli ve toplayıcının malzemeye eriştiği depolar için yerleşim tasarımlarında kullanılmak üzere toplayıcı mesafesini minimize ederek, optimum raf sayısını belirlemeye yarayan analitik bir yöntem geliştirmiştir. Bu yöntemde toplam raf uzunluğu, toplama sırasında uğranacak lokasyon sayısı ve toplama oranı dikkate alınarak depodaki raf sayısı belirlenmiştir. Söz konusu çalışmanın devamı niteliğinde olan bir diğer çalışmada ise, ilk çalışmada yer alan toplam raf uzunluğu, uğranacak lokasyon sayısı ve toplama oranı değişimlerinin depo sistemine etkileri simülasyon ile analiz edilmiştir (Caron ve diğ., 2000b).

Roodbergen ve Koster (2001) tarafından orta (çapraz) koridora sahip depolar için toplayıcının ortalama toplama süresini minimize eden, toplayıcı rotası algoritması geliştirilmiştir. Orta koridora sahip depolar ile orta koridora sahip olmayan depolar; farklı büyüklükte toplayıcı listesi, farklı ekipman, farklı depo tipi, farklı büyüklük ve farklı sayıda raf sayısı olacak şekilde simülasyon modelleri hazırlanmış ve söz konusu modeller karşılaştırılmıştır. Simülasyon sonucunda, hem orta koridora sahip hem de orta koridora sahip olmayan depolar için farklı büyüklükteki depolar ve farklı büyüklükteki toplama listelerine göre en uygun raf sayısı belirlenmiştir.

Lai ve diğ. (2002) tarafından kağıt ruloları için depo yerleşim problemi üzerine çalışma yapılmıştır. NP-zor niteliğe sahip bu problemin çözümü iki aşamada ele alınmıştır. Birinci aşamada kağıt ruloları sabit depolama politikasına göre gruplandırılarak, problemin tamsayılı modellemesi yapılmıştır. Matematiksel yöntemler ile çözümü zor olan bu problem, standart tavlama yöntemi ve iki yeni geliştirilmiş tavlama benzetim yöntemi kullanılarak çözülmüştür. Tüm yöntemlerle elde edilen sonuçlar karşılaştırılarak, yöntemler arasından en uygun olan belirlenmiştir.

Macro ve Salmi (2002)'ye ait çalışmada, farklı depo büyüklüğüne sahip depolar için depo kapasitesinin ve raf kullanım etkililiğinin analizinde kullanılacak simülasyon modelleri hazırlanmıştır. Farklı raf sistemi kombinasyonları ile hazırlanmış alternatif

depolar, söz konusu depolarda kullanılan raf tiplerine göre kullanım oranlarına, ortalama palet sayısına ve maksimum palet sayısına göre karşılaştırılmış ve elde edilen sonuçlar analiz edilmiştir.

Queirolo ve diğ. (2002), genetik ve simülasyon tabanlı hibrit depo yerleşim algoritması geliştirmiştir. Toplayıcı mesafesini ve süresini minimize etmeye yarayan algoritmanın kullanılması ile malzemelerin depo içerisindeki yerleştirileceği lokasyonlar belirlenmiştir.

Petersen (2002) tarafından gerçekleştirilen çalışmada, rassal depolama ve hacim bazlı depolama politikalarını sahip depolar için koridor uzunluğunun ve koridor sayısının, toplam sipariş toplayıcı mesafesine olan etkisi simülasyon kullanılarak analiz edilmiştir.

Zhang ve diğ. (2002) tarafından farklı seviyeli depolama alanlarına sahip depolarda malzemelerin depoya yerleştirilmesi problemi modellenmiş ve modelin çözümü için genetik algoritma tabanlı sezgisel algoritma önerilmiştir. Söz konusu modelin amacı, malzemelerin yatay ve dikey hareketlerden ortaya çıkan elleçleme maliyetlerini dikkate alarak, toplayıcının toplam elleçleme maliyetini minimize etmektir. Önerilen model, NP-zor olduğu için genetik algoritma tabanlı sezgisel yöntem kullanılmıştır.

Yang ve Sun (2004), bulanık rastgele simülasyon tabanlı tabu arama algoritmasından oluşan hibrit algoritma kullanarak, uygun malzeme yerleşimini önermiştir. Sınıf bazlı depolama politikalı depolarda ortalama taşıma maliyetini enküçüklemeye yarayan depo malzeme yerleşimi modellenmiş ve söz konusu model bulanık rastgele simülasyon tabanlı tabu algoritması ile sezgisel olarak çözülmüştür. Önerilen hibrit yöntem iki sayısal örnek üzerinde uygulanmış ve söz konusu yöntemler ile elde edilen sonuçlar sunulmuştur.

Heragu ve diğ. (2005), fonksiyonel alanlara malzeme atanmasında ve fonksiyonel alan büyüklüklerinin belirlenmesi probleminin çözümünde kullanılan sezgisel bir algoritma önermiştir. Fonksiyonel depo alanı büyüklüğünü belirleme için öncelikle bu alanlarda karşılaşılan akışlar belirlenmiştir. Ayrıca, rezerve/ileri depolama politikası yerleşimli depolarda dört farklı malzeme akışı belirlenmiştir. Problemin modellenmesinde ifade edilmiş matematiksel modelin amacı, ortalama elleçleme ve depolama maliyetlerini enküçüklemektir. Daha iyi bir değerlendirme yapılabilmesi için, önerilen sezgisel

algoritma ile karma tamsayılı programlamada kullanılan dal-sınır algoritması karşılaştırılmıştır.

Le-Duc ve Kosner (2005) tarafından ABC depolama politikasını kullanan 2 bloklu depo için depolama alanları optimizasyonu probleminin çözümü için sezgisel bir çözüm prosedürü önerilmiştir. Sözü edilen çalışmada, malzemelerin olasılıklarına bağlı olarak toplayıcının ortalama mesafesini hesaplayan matematiksel bir model önerilmiştir. Önerilen olasılıklı model, simülasyon yaklaşımı kullanılarak doğrulanmıştır.

Zhang ve Lai (2006)'ya ait çalışmada çok seviyeli depo yerleşim probleminin çözümü için genetik algoritma ve tekrar yol bağlama stratejisi (path relinking strategy) kullanan sezgisel algoritmalar önerilmiştir. Söz konusu sezgisel algoritmalar kullanılarak çok seviyeli depolar için malzemelerin yerleşimleri belirlenmiştir. Çok seviyeli depolar için depo yerleşiminde farklı seviyelerden malzeme taşımaları gerçekleştirildiği için yatay malzeme taşıma maliyetinin yanı sıra dikey malzeme taşımalarından kaynaklanan dikey taşıma maliyeti de göz önünde bulundurulmuştur.

Roodberden ve Vis (2006), depo yerleşim optimizasyonu için analitik bir model önerilmiştir. Bu model, toplama alanında toplayıcının ortalama mesafesini enküçükleyerek malzemelerin yerleşimini belirlemektedir. Ortalama toplayıcı mesafesinin hesaplanmasında iki farklı toplayıcı rotalama politikası dikkate alınarak hesaplama yapılmaktadır. Söz konusu rotalama politikaları, S-şeklinde rotalama politikası ve en büyük boşluk rotalama politikasıdır. Analitik modelin değerlendirilmesinde simülasyon yaklaşımı ve literatürde kabul görmüş iki farklı modelin karşılaştırılması yapılmıştır.

Hsieh ve Tsai (2006) tarafından simülasyon programı yardımıyla tasarım faktörlerinin belirlendiği bir depo tasarımı önerilmiştir. Depo tasarımını etkileyen faktörler, çapraz koridor sayısı, depo yerleşim stratejisi, sipariş kombinasyonu, sipariş toplama politikası ve koridordaki toplama yoğunluğu olarak belirlenmiştir. Bu faktörlerin farklı kombinasyonları ile çok sayıda alternatif tasarımlar hazırlanmış ve simülasyon programında hazırlanan alternatif tasarımlar birbirleriyle karşılaştırılmıştır.

Huertas ve diğ. (2007), farklı depo yerleşim alternatiflerinin operasyon maliyetlerini hesaplayan bir model önermiştir. Aynı depolama kapasitesine sahip üç depo alternatifi

(U-şekilli depo yerleşim tasarımı, akış-tipi depo yerleşim tasarımı ve fonksiyonel alanlı akış tipi depo yerleşim tasarımı) belirlenmiş ve bu alternatifler karşılaştırılmıştır.

Önüt ve diğ. (2008), farklı seviyeli depolama alanlarına sahip ABC depolama politikalı depolarda depo yerleşim düzenlemesi problemi için matematiksel bir model ve bu modelin çözümü için sezgisel bir algoritma önermiştir. Toplam toplayıcı maliyetini enküçüklemek için uygun raf uzunlukları, raf yükseklikleri ve raf sayıları belirlenmeye çalışılmıştır. Önerilen sezgisel algoritma, parçacık sürü algoritması tabanlıdır ancak kısıtlı problemlere uygulanabilmesi için modifiye edilmiştir.

Pohl ve diğ. (2009), çift emir operasyonlu birim yük depoları için bir depo yerleşim tasarımı önermiştir. Endüstride en çok karşılaşılan üç depo yerleşim düzenlemesi sunulmuş ve bu yerleşimlere ait modeller matematiksel olarak ifade edilmiştir. Toplama/gönderme noktasına ve çapraz koridorun konumuna göre depo tasarımları farklılaştırılmıştır. Önerilen depo yerleşim tasarımları çözülerek, toplam malzeme elleçleme mesafesini enküçükleyecek depo raf uzunlukları ve çapraz koridor konumları belirlenmiştir.

Gue ve Meller (2009) tarafından geleneksel depo yerleşim düzenlemesi tasarımlarına alternatif modern depo yerleşim düzenlemesi tasarımları önerilmiştir. Önerilen V-dizilimi ve balık kılçığı depo yerleşim düzenlemesi tasarımlarında, yatay çapraz koridorlar yerine köşegen çapraz koridor kullanılması ile toplayıcının toplam mesafesi azaltılmaya çalışılmıştır. Köşegen çapraz koridorun rafı bölme miktarına ve koridor sayısına bağlı olarak toplayıcının toplam mesafesini enküçüklemeye yarayan matematiksel bir model önerilmiştir.

Zhang ve Lai (2010) tarafından komşuluk kısıtlı çok seviyeli depolar için malzeme yerleşim problemine ait bir model önerilmiştir. Sözü geçen problemin modeli NP-zor niteliğe sahip olduğundan problemin çözümü için sezgisel algoritmalardan faydalanılmıştır. Sipariş hacim indeksi tabanlı sezgisel yöntem, standart tabu arama, açgözlü tabu arama, dinamik komşuluk ekleme tabanlı tabu arama yöntemleri kullanılmıştır. Toplayıcının toplam toplama maliyeti dikkate alınarak çözüm yöntemlerinin sonuçları karşılaştırılmış ve en uygun malzeme atama yerleşimi belirlenmiştir.

Gue ve diğ. (2012) tarafından ters V-dizilimi depo yerleşim düzenlemesi tasarımı ve söz konusu tasarımın matematiksel modeli önerilmiştir. Sözü edilen depo yerleşim düzenlemesi tasarımı ve V-dizilimi depo yerleşim düzenlemesi tasarımları ile hazırlanmış farklı depo yerleşim düzenlemesi tasarımlarının sonuçları karşılaştırılmıştır.

Öztürkoğlu ve diğ. (2012) tarafından birim yük depolarında kullanılmak üzere üç modern depo yerleşim düzenlemesi tasarımı önerilmiştir. Zikzak, yaprak ve kelebek olarak adlandırılmış yeni depo yerleşim düzenlemesi tasarımlarının modelleri matematiksel olarak ifade edilmiştir. Söz konusu tasarımlar kullanılarak hazırlanan ve farklı koridor sayısına göre değişen depo yerleşim düzenlemesi sonuçları, ortalama toplayıcı mesafesi dikkate alınarak karşılaştırılmıştır.

3. MALZEME VE YÖNTEM

Bu bölümde, depo yerleşim düzenlemesi probleminin çözümü için kullanılacak metasezgisel yöntem açıklanacaktır. Bunun yanı sıra, matematiksel olarak ifade edilecek depo yerleşim düzenlemesi problemi, sabit yükseklikli paletler ve farklı yükseklik paletler için depo yerleşim düzenlemesi olacak şekilde iki farklı versiyon da modellenecektir. Daha sonra, PSO algoritması, depo yerleşim düzenlemesi problemin uyarlanması anlatılacaktır. PSO algoritması ile problemin çözümünün daha hızlı ve daha doğru yapılabilmesi için kullanılan mevcut ve önerilen sınırlandırma yöntemleri ve parçacıkların başlangıç çözüm uzayını daraltan başlangıç minimum değerleri için atama algoritması açıklanacaktır. Standart PSO algoritması ve önerilen sınırlandırma yöntemlerini kullanan PSO algoritması ile farklı yükseklik paletler için depo yerleşim düzenlemesi problemi ayrı ayrı çözülecek ve tablolar halinde özetlenecektir. Son olarak da, farklı depo yerleşim düzenlemesi örnekleri uygulanarak sonuçları elde edilmiş standart PSO algoritması ile önerilen sınırlandırma yöntemlerini kullanan PSO algoritması, tek bir tablo üzerinde karşılaştırılarak özetlenecektir.

3.1. OPTİMUMU BULMADA YARARLANILAN METASEZGİSEL YÖNTEMLER

Ele alınan problemin ne kadar zor olduğunun bilinmesi, problemi çözmek için kullanılacak en iyi yöntemin seçilmesini kolaylaştırmaktadır. Metasezgisel yöntemler ile polinomal (P) denklemlerin çözülmesi kolay olup kısa sürede çözüme ulaşılabilmektedir. Fakat denklem ve ele alınan sistem polinomal değilse, (NP; Nonpolynomially Bounded) bu tür problemlerin çözülmesi zorlaşacak veya çözümü çok uzun zaman alacaktır. Bu tür NP-zor problemlerin kısa sürede çözülmesi zor olduğundan gerçek çözüme en yakın sonucu bulma amacıyla yaklaşık çözüm algoritmaları geliştirilir (Elmas, 2010).

Yaklaşık çözüm algoritmaları, problem için en uygun çözümü bulmada kullanılan çözüm algoritmalarıdır. Genellikle bu tür çözüm algoritmalarında gerçek çözüme en yakın değerin hızlı şekilde bulunması amaçlanmaktadır. Bu durumda, tüm problemlere

doğrudan uygulanabilir çözüm geliştirmek zor olduğundan probleme özgü sezgisel çözüm yöntemleri geliştirilir (Elmas, 2010).

Sezgiseller çözüm yöntemleri çoğu zaman probleme özgü geliştirilmiş yöntemler olduğundan bir problem çözümü için kullanılan bir sezgisel yöntem, başka bir problem için kullanılamamaktadır. Sezgisel yöntemlerin çok dar uygulama alanı olduğundan, araştırmacıların daha çok çeşitli problemlere uygulanabilen genel özelliğe sahip olan metasezgisel tekniklere (Tavlama Benzetimi, Tabu Arama, Parçacık Sürü, vb.) ilgisi son yıllarda giderek artmıştır. Bu teknikler çok sayıda problemlere uygulanmış ve oldukça güçlü bulunmuşlardır (Yiğit ve Türkbey, 2003).

Çalışmanın bu bölümünde, NP-zor problemlerin çözümünde kullanılan bazı metasezgisel yöntemler açıklanacaktır. Bu yöntemler, tavlama benzetimi, tabu arama, parçacık sürü, karınca kolonisi ve genetik algoritmalardır.

3.1.1. Tavlama Benzetimi

Tavlama Benzetimi (TB), büyük çözüm kümesi sahibi olan NP-zor sınıfındaki optimizasyon problemlerin için kısa sürede gerçek çözüme en yakın sonucu üreten stokastik yöntemlerden birisidir. TB algoritması, birbirlerinden bağımsız olarak, Kirkpatrick, Gelatt ve Vecchi (1983) ve Cerny (1985) tarafından ortaya konmuştur. Bu algoritma doğadan ilham alınarak istatistiksel mekanikten türetilmiş ve fiziksel sistemlerin sıcak banyoda diğer bir deyişle fiziksel tavlanmasında sistemin davranışlarının analiz edilmesi ile türetilmiştir. Tavlama benzetimi ismi, katıların fiziksel tavlanma süreci ile olan benzerlikten ileri gelmektedir (Johnson ve diğ., 1989).

TB süreci, ısı banyosundaki bir malzemenin yüksek enerjili eriyik durumundan düşük enerjili katı durumlarının elde edilmesidir. Katı malzeme, öncelikle katının eriyebileceği sıcaklığa kadar ısıtılır ve katı hale geçinceye kadar tekrar soğutulur. Katı malzemenin soğutulma esnasındaki kristal yapısı soğutma oranına bağlı olur. Erimiş kristaller, çok yavaş bir soğutma işlemi görürse büyüyebilir. Çok hızlı bir soğutma işlemi görür ise, kristal yapısında uygun bir atomik dizilim elde edemeyeceği için yarı kararlı bir halde olacaktır (Gülsün ve diğ., 2008).

Tavlama benzetimi algoritması, yüksek bir T başlangıç sıcaklığı ile başlanır ve bu sıcaklık değeri belli bir sıcaklık düşürme prensibiyle yavaş yavaş düşürülerek çok sayıda komşu çözümler üretilir. Üretilen komşu çözümler, amaç fonksiyonu exp((-

Δ)/kT) Boltzman denklemi dikkate alınarak reddedilir veya kabul edilir (Gülsün ve diğ., 2008). Eğer bir iyileşme durumu varsa, komşu çözüm yeni mevcut çözüm olarak kabul edilir. Eğer iyileşme durumu yoksa, mevcut çözüm değişmez. İlk başlarda, T değeri yüksek olduğundan algoritma iyi olmayan çözümlerin arama sürecine katılmasını engellemektedir. Ancak algoritma ilerledikçe ve T değeri azalmaya başladıkça algoritma iyi olmayan çözümleri eleyerek iyi çözümlerden devam etmektedir. T değeri sıfıra yaklaştıkça amaç fonksiyonunda artışa sebep olan çözümlerin çoğu reddedilecektir.

3.1.2. Tabu Arama

Tabu arama (TA), ilk olarak Glover (1986) tarafından önerilmiştir. Bugünkü kullanılan modern haline gelişi ise, Glover (1989; 1990) tarafından yayınlanan iki farklı çalışma sayesinde olmuştur. Tabu arama, bir komşuluk bazlı, kısa hafızaya sahip iteratif bir metasezgisel yöntemdir (Dreo ve diğ., 2006).

TA, elde edilen lokal en iyi çözümlerden yola çıkılarak daha iyi çözümlerin araştırılmasını içeren metasezgisel bir yöntemdir. Tabu arama, kısa dönemli hafızasında daha önceki aramalara ait bazı bilgileri saklayarak ve bazı çözümlerin tekrar elde edilmesine "yasak" koyarak sezgisel algoritmayı lokal optimumlardan uzaklaştırmaktadır ve yeni bölgeleri araştırmayı hedeflemektedir (Salvendy, 2001).

Tabu arama, tüm kısıtları sağlayan başlangıç çözümü ile başlar. Her adımda geçerli çözümün komşuluklarından oluşan bir "aday liste" hazırlanır ve bir değerlendirme fonksiyonu (amaç fonksiyonu) ile değerlendirilir. Değerlendirme sonunda, aday listedeki en uygun çözüm yeni mevcut çözüm olur. Bu çözüme ulaştıran hareketin tersi de yasaklanarak "tabu listesine" eklenir. Tabu listesi yasaklanmış hareketlerden oluşan bir liste olduğu için bu listedeki hareketler belirli bir iterasyon sayısı boyunca tekrar mevcut çözüm olarak seçilemezler. Daha önce seçilen çözüme ulaştıran hareketlerin yasaklanması ile, lokal optimumlara yakalanma olasılığı azaltılır ve arama uzayı genişletilerek daha uygun sonuçlar elde edilebilir (Nowicki ve Smutnicki, 1996).

3.1.3. Genetik Algoritma

Genetik Algoritma (GA), ilk kez John Holland tarafından 1975 yılında geliştirilmiş olup, doğal seçim ve genetik popülasyonların simüle edilmesidir. (Beasley ve diğ., 1993). Genetik algoritmalar, doğada gözlemlenen evrimsel sürece benzer şekilde, iyi

nesillerin hayatlarını devam ettirirken, kötü nesillerin yok olması ilkesine dayanmaktadır. Bir önceki nesilden (anne ve baba) doğan yeni bireylerin ortam koşullarına ayak uydurup yaşama prensibine dayanır. Bu yeni doğan bireyler, önceki nesilin iyi genlerine sahip olabildiği gibi önceki nesilin kötü genlerine de sahip olabilirler. Fakat ortama uyum sağlamayan kötü nesiller hayatlarını devam ettiremeyeceklerdir (Elmas, 2010).

Genetik algoritma, matematiksel modelleme yapılamayan ve yapılması zor olan, çok değişkenli, karmaşık problemlerin çözümünde kullanılabilen popülasyon temelli sezgisel bir yöntemdir. Algoritma, karmaşık problemin çözümü için gerekli bilgi ve varsayımlar olmadan sadece amaç fonksiyonu ile çalışabilmektedir. Tavlama benzetimi ve tabu arama yöntemlerinin aksine çözüm için tek bir değerin geliştirilmesi yerine, birden çok değerin oluşturulduğu çözüm kümesi elde edilir (Keskintürk ve Şahin, 2009).

Genetik Algoritma, çaprazlama, mutasyon ve inversiyon gibi genetikten ilham alınmış operatörleri ile "doğal seleksiyonu" birlikte kullanarak "kromozomların" bir nüfustan yeni nüfusa hareketinde kullanılan bir yöntemdir. Her bir kromozom genler içermekte ve her bir gen ise "alel" içermektedir. Alel ise kromozomun en küçük bilgi taşıyan parçasıdır. Seçim operatörü, popülasyon içerisindeki daha iyi olan kromozomların yeni nesilin üretiminde daha çok kullanılmasına izin vermektedir. Çaprazlama, iki kromozomun bazı gen parçalarının karşılıklı değiştirilmesidir. Çaprazlama ile seçilen iki kromozomun genleri çaprazlanarak yeni bireyler elde edilir. Mutasyon, kromozomdaki rastgele seçilen bir veya daha fazla alel değerinin kalıcı olarak değiştirilmesidir. İnversiyon, kromozomun bitişik gen parçalarını değiştirir. Bu sayede kromozomun dizilimi değişmektedir (Mitchell, 1998).

3.1.4. Karınca Kolonisi

Karınca kolonisi (KK) algoritması, zor kombinatoryal optimizasyon problemlerinin çözümü için M. Dorigo ve arkadaşları tarafından doğadan ilham alınarak geliştirilmiştir. Karınca kolonisi algoritmasının ilham kaynağı gerçek karıncaların yiyecek arama davranışlarıdır. Gerçek karınca kolonisinde, karıncalar yiyecek aramaya çıktıkları zaman başlangıçta yuvalarının çevresini rastgele keşfe çıkarlar. Keşfe çıkan karıncalardan birisi yiyecek kaynağı bulduğunda, yiyecek kaynağının miktarını ve

kalitesini değerlendirir ve yiyecek kaynağının birazını yuvasına geri taşır. Karınca, kaynağa giden yol üzerinde "feromon" olarak adlandırılan bir kimyasal madde bırakır. Yiyecek kaynağının miktarı ve kalitesine bağlı olarak bırakılan feromon miktarı, diğer karıncalara yiyecek kaynağı için rehberlik edecektir. Feromon kimyasalının bırakılma nedeni ise karıncaların kör olmasıdır. Bırakılan feromon maddesi kalıcı değildir ve belirli bir süre sonra buharlaşmaktadır. Yiyecek kaynağına kısa yoldan giden karıncalar, daha önceden karıncalar tarafından bırakılan feromon çok fazla buharlaşmadan tekrar üstünden geçtiklerinden, bu yolda daha yüksek feromon maddesi bulunacaktır. Belirli bir zaman sonunda, tüm karıncalar aynı yolu izledikleri için yoldaki feromon miktarı yükselecektir ve en kısa yoldan yiyeceğe ulaşmış olacaklardır (Dorigo ve Blum, 2005).

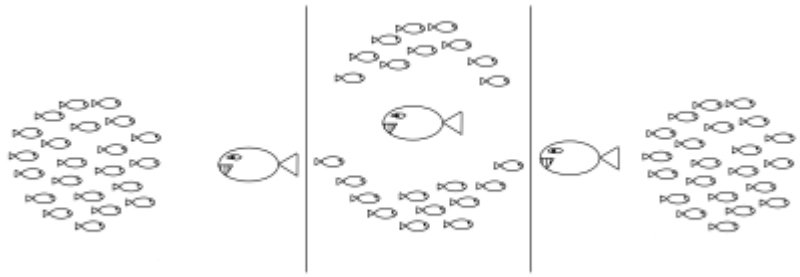
KK algoritması ise gerçek karınca kolonisinin bazı davranışlarını taklit etmektedir. Bu davranışlar, karıncalar arasında feromon yardımıyla iletişimini, kaynak ile yuva arasındaki kısa yolda daha çok feromon maddesi bulunmasını ve karıncaların feromon miktarının çok olduğu yolları tercih etmeleridir. Ancak karınca kolonisi algoritmasında gerçek karınca davranışlarına ek olarak bazı davranışlar eklenmiştir. Yapay karıncalar gerçek karıncalar gibi kör değildirler, hafıza sahiptirler ve uğradıkları yerlere geri dönmemeleri için tabu listesine sahiptirler. Bunlara ek olarak yol tercihi sırasında feromon maddesinin yanı sıra, gelişi güzellik de söz konusudur (Karaboğa, 2011).

3.1.5. Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu (PSO), Eberhart ve Kennedy (1995) tarafından kuş ve balık sürülerinin sosyal davranışlarından esinlenerek, doğrusal olmayan nümerik problemlere optimal sonuçlar bulmak için geliştirilmiştir. PSO, popülasyon tabanlı stokastik optimizasyon yöntemi olup çok parametreli ve çok değişkenli optimizasyon problemlerine çözümler üretmek için kullanılmaktadır.

Kuş ve balık sürüleri, yiyecek kaynağını nerelerde olduğunu bilmemelerine rağmen, bu kaynaktan uzaklıklarını öğrenmeye çalışırlar. Bunu öğrenmenin yolu ise, yiyecek kaynağına en yakın olan kuşu izlemektir (Akyol ve Alataş, 2012). Bunun yanı sıra, kuş ve balık sürülerindeki bireyler tarafından sınırlı bilgi (en yakın komşu bireyin hızı ve konumu) paylaşımı olduğu durumlarda, karmaşık yer değişimlerinde kolektif davranışlar gözlemlenebilmektedir. Şekil 3.1'de şematik olarak anlatıldığı gibi, balık

sürüleri yırtıcılardan kurtulmak için kolektif davranışlarda bulunmaktadır (Dreove diğ., 2006).



Şekil 3.1: Balık sürüsünün yırtıcıdan kurtulmasının şematik anlatımı (Dreo ve diğ., 2006).

PSO algoritmasında, optimizasyon probleminin her bir çözümü arama uzayında parçacık olarak ifade edilmiştir. Çözüm uzayında her bir parçacığın konumu ve hızı bulunmaktadır. Bu hızın yönü ve mesafesi tüm parçacıklar için belirlenmiştir (Zhu, 2009). PSO yaklaşımında parçacıklar, amaç fonksiyonuna göre çözüm uzayındaki en iyi parçacık ve bireysel en iyiyi izleyerek; sık sık pozisyonlarını ve uygunluklarını değiştir, uçma yönünü ve hızını belirler. Diğer bir anlatımla parçacık, hareket yönünü bireysel en iyiye ve sürünün en iyisine göre belirlemektedir.

Bu tez kapsamında ele alınan problemin çözümü için parçacık sürü optimizasyon algoritması seçilmiştir. PSO algoritmasının seçilme nedenleri şu şekilde sıralanabilir:

- PSO algoritması, popülasyon bazlı arama algoritmasına sahip olduğundan birden fazla sonuç elde edilmekte ve daha fazla parçacık çözüme katkı sağlamaktadır.
- Tabu arama ve tavlama benzetim gibi metasezgisel yöntemlerde tek bir sonuç elde edilirken; PSO algoritmasında sürüdeki parçacık sayısı kadar sonuç elde edilmektedir.
- PSO algoritması, diğer evrimsel algoritmalara göre daha kolay uygulanmaktadır. PSO algoritmasında diğer evrimsel algoritmalarda olduğu gibi mutasyon ve çaprazlama operatörlerine ihtiyaç bulunmamaktadır.
- PSO algoritmasında merkezi kontrol mekanizmasının olmaması bireysel hataların göz ardı edilmesini kolaylaştırmaktadır. Böylece, bazı bireylerin başarısız olması tüm sürünün doğrudan etkilenmesine neden olmamaktadır.

- PSO algoritması sürü davranışı gösterdiğinden, kullanıcının modele etki edeceği parametre sayısı azdır. Bu durum kullanıcın modele daha az müdahale etmesini ve dolayısıyla, PSO algoritmasının doğal akışının bozulmamasını sağlamaktadır.

Yukarıda tez kapsamında ele alınan problemin çözümünde kullanılmakta olan PSO algoritmasının seçilme nedenleri yer almaktadır. Diğer taraftan, PSO algoritmasının kısıtlı problemlerin çözümünde yetersiz kaldığı bilinmektedir. PSO algoritmasının bu yetersizliğinin üstesinden gelinebilmesi ve söz konusu algoritmanın kısıtlı problemlerin çözümünde de kullanılmasını sağlayan birtakım sınırlandırma yöntemleri bulunmaktadır. Ancak PSO algoritması için geliştirilmiş ve mevcut olarak kullanılmakta olan söz konusu sınırlandırma yöntemleri ile de kısıtlı problemlerin çözümünde tam olarak istenilen sonuçlar elde edilememektedir. Bunun nedeni parçacıkların çözüm uzayının sınırlarını fazla zorlamamaları ve daha kısıtlı uzayda hareket etmeleridir. Bu bağlamda, tez kapsamında ele alınan problemin çözümü için yalnızca PSO algoritması için geliştirilmiş mevcut sınırlandırma yöntemlerinden faydalanılmakla kalınmayıp; aynı zamanda söz konusu sınırlandırma yöntemlerinin yetersiz kaldığı çözüm uzayının sınırlarını zorlayacak yöntemler de geliştirilecektir.

3.2. DEPO YERLEŞİM DÜZENLEMESİ PROBLEMİ

Bu bölümde, öncelikle sabit ve farklı yükseklikli paletler için iki farklı depo yerleşim düzenlemesi matematiksel olarak ifade edilecektir. Daha sonra, depo yerleşim düzenlemesi problemine ait çözüm yönteminin, depo yerleşim düzenlemesi üzerine uygulanmasının adımları anlatılacaktır. Ayrıca, PSO algoritmasının kısıtlara sahip problemlerin çözümü için geliştirilmiş mevcut sınırlandırma yöntemleri ve doktora tezi kapsamında önerilen sınırlandırma yöntemleri açıklanacaktır. Son olarak, PSO algoritmasının daha hızlı çözüme ulaşmasında parçacıkların başlangıç çözüm uzayını daraltan başlangıç minimum değerleri için geliştirilen atama algoritması sunulacaktır.

3.2.1. Depo Yerleşim Düzenlemesi Probleminin Modellenmesi

Bu bölümde, depo yerleşim düzenlemesi problemi matematiksel olarak ifade edilecektir. Bu bölüm iki alt bölümden oluşmakta olup; söz konusu bölümlerde sabit yükseklikli paletler için depo yerleşim düzenlemesi ve değişken yükseklikli paletler için depo yerleşim düzenlemesi problemleri modellenecektir. Her iki depo yerleşim düzenlemesinde ortak olan birtakım terimler ve formüller bulunmaktadır. Bu terimler ve

formüller, her iki alt bölüm içinde ayrı ayrı tekrar edilmemesi ve karışıklığa yol açmaması için bu bölümde ele alınacaktır.

Depo yerleşim düzenlemesi problemi; depo çeşidi, depolanacak malzemenin özelliği, depolanan malzemenin elleçleme veya taşıma türü, kullanılan depolama ve yerleştirme politikası ve sipariş toplamanın tek/çok iş emirli olması gibi özelliklere göre çeşitlilik göstermektir. Söz konusu özelliklere ilişkin olarak, doktora tezine konu olan depo yerleşim düzenlemesi problemi aşağıdaki şekilde ele alınmıştır.

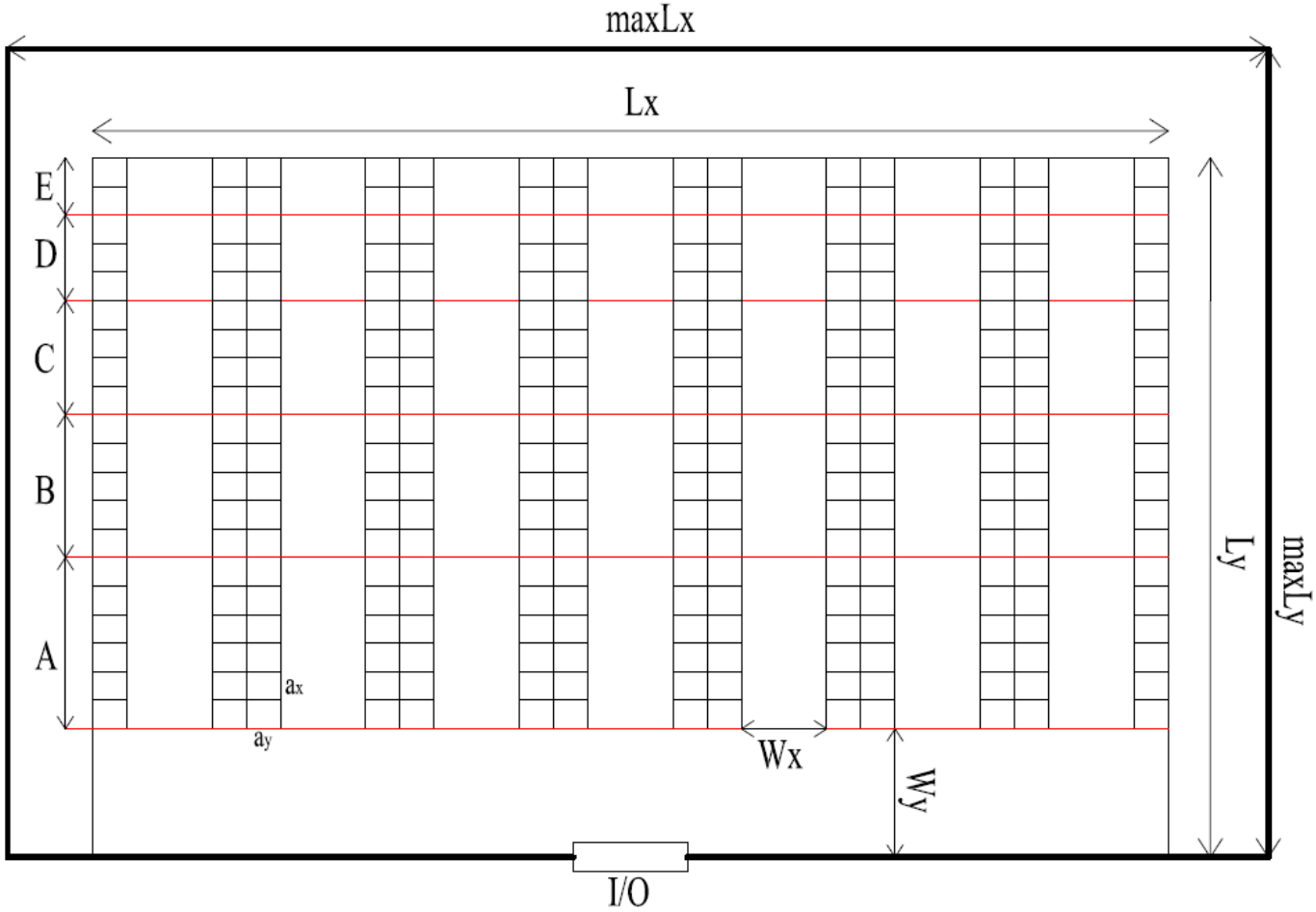
- Depo yerleşim düzenlemesi yapılacak olan depo çeşidi dağıtım merkezi deposudur.
- Depoya gelen malzemeler birim yük olarak gelmekte ve birim yük olarak gönderilmektedir.
- Birim yük olarak tanımlanan malzeme elleçleme birimi palettir.
- Depolama politikası olarak ABC sınıflandırma politikası kullanılmaktadır. Buna göre, depoya gelen malzemeler depoya giriş çıkış miktarına göre gruplara ayrılır. Ayrılan malzemeler ait oldukları malzeme sınıfına göre depoya yerleştirilir.
- Sipariş toplama sürecinde sipariş toplayıcı birim yükü (paleti), daha önceden malzeme grubu için ayrılmış raf alanına yerleştirmekte veya alması gereken malzemeyi raf alanından almaktadır. Bu süreç, tek iş emirli olup; malzemelerin rafa yerleştirilmesi veya raftan alınması operasyonudur.
- Depo yerleşim düzenlemesinde kullanılacak raf sistemi, endüstride yaygın olarak uygulanmakta olan sırt-sırta raf sistemidir.

Depo yerleşim düzenlemesi problemindeki depo tipi, elleçleme birimi, depolama politikası ve operasyon türü açıklandıktan sonra, depo yerleşim düzenlemesi probleminin matematiksel olarak ifade edilmesi için bazı terimlere ihtiyaç duyulmaktadır. Depo yerleşim düzenlemesi probleminde kullanılacak tüm terimlere Tablo 3.1'de yer verilmiştir.

Kullanılan terimlerin daha iyi anlaşılabilmesi için, bu terimler Şekil 3.2'de görselleştirilmiştir. Şekil 3.2'de malzemeler beş farklı malzeme grubuna ayrılarak gösterilmiştir. Ancak, malzeme grubu sayısı isteğe bağlı olarak değişebilmektedir.

Tablo 3.1: Depo yerleşim düzenlemesinde kullanılan terimler.

K	Toplam gerekli olan stok pozisyonu sayısı
Ps	Yıllık depodan çıkan palet sayısı
C_h	Elleçleme maliyeti
d	Deponun malzeme girişini/çıkışını sağlayan kapının sayısı
gs	Ürün grupları sayısı
N_i	i malzeme grubu için ayrılan stok pozisyonu sayısı
P_i	i ürün grubu için talepte bulunma olasılığı
L_{Xmax}	x-ekseni boyunca maksimum uzunluk
L_{Ymax}	y-ekseni boyunca maksimum uzunluk
L_{Zmax}	z-ekseni boyunca maksimum uzunluk
T_x	x-ekseni boyunca katedilen ortalama toplayıcı mesafesi
T_y	y-ekseni boyunca katedilen ortalama toplayıcı mesafesi
T_z	z-ekseni boyunca katedilen ortalama toplayıcı mesafesi
a_x	x-ekseni boyunca palet için ayrılan mesafe
a_y	y-ekseni boyunca palet için ayrılan mesafe
a_z	z-ekseni boyunca palet için ayrılan mesafe
a_{zi}	i ürün grubu için z-ekseni boyunca palet için ayrılan mesafe
w_x	Ara koridor genişliği
w_y	Ana koridor genişliği
n_X	x-ekseni boyunca raf sayısı
n_Y	y-ekseni boyunca raf sayısı
n_Z	z-ekseni boyunca raf sayısı
n_{Xmax}	x-ekseni boyunca maksimum raf sayısı
$n_{Yi\text{-}max}$	i ürün grubu için y-ekseni boyunca maksimum raf sayısı
$n_{Zi\text{-}max}$	i ürün grubu için z-ekseni boyunca maksimum raf sayısı
n_{Xmin}	x-ekseni boyunca minimum raf sayısı
$n_{Yi\text{-}min}$	i ürün grubu için y-ekseni boyunca minimum raf sayısı
$n_{Zi\text{-}min}$	i ürün grubu için z-ekseni boyunca minimum raf sayısı
n_{Xi}	i ürün grubu için x-ekseni boyunca ayrılan stok pozisyonu sayısı
n_{Yi}	i ürün grubu için y-ekseni boyunca ayrılan stok pozisyonu sayısı
n_{Zi}	i ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı



Şekil 3.2: Örnek depo şekli.

Şekil 3.2'den de görüleceği gibi deponun boyu (x-ekseni boyunca uzunluğu), deponun eni (y-ekseni boyunca uzunluğu) ve deponun yüksekliği (z-ekseni boyunca uzunluğu) gibi uzunluklar, depodaki raf sisteminin durumuna bağlıdır. Söz konusu uzunluklar deponun eni, deponun boyu ve deponun yüksekliği Denklem 3.1'de gösterilmektedir.

$$L_X = (a_x + w_x/2)n_X$$

$$L_Y = a_y n_Y + w_y$$

$$L_Z = (n_Z\, a_z) \qquad (3.1)$$

Denklem 3.1'de gösterilen ilk eşitlik deponun x-ekseni boyunca uzunluğunu göstermektedir. İkinci eşitlik ise, deponun y-ekseni boyunca enini göstermektedir. Son eşitlik ise, deponun z-ekseni boyunca yüksekliğini göstermektedir. Denklem 3.1'den elde edilen eşitlikler, daha sonraki alt bölümlerde anlatılacak olan sabit yükseklikli paletler için depo yerleşimi modelinde ve değişken yükseklikli paletler için depo yerleşimi modelinde kullanılacaktır.

3.2.1.1. Sabit Yükseklikli Paletler için Depo Yerleşim Düzenlemesi Probleminin Modeli

Bu bölümde, sabit yüksekli paletler için depo yerleşim düzenlemesi probleminin modeli açıklanacaktır. Depo yerleşim düzenlemesinin amacı ortalama toplayıcı mesafesi ile ilişkili olan ortalama toplama maliyetini enküçüklemektir. Deponun üç boyutu olduğu için üç farklı ortalama toplayıcı mesafesi karşımıza çıkmaktadır. Buna göre toplam toplayıcı mesafesi; x-ekseni boyunca, y-ekseni boyunca ve z-ekseni boyunca kat edilen ortalama toplayıcı mesafelerinin toplamıdır. Her üç ortalama toplama mesafesinin matematiksel ifadesi ayrı ayrı hesaplanacaktır.

İlk hesaplanan mesafe, x-ekseni (depo boyu) boyunca kat edilen ortalama toplayıcı mesafesidir. Deponun malzeme girişi/çıkışı sağlayan kapılarının konumları ortalama toplayıcı mesafelerini etkilemektedir. Deponun malzeme girişi/çıkışı tek bir kapıdan yapılıyorsa, deponun kapısı depo yatay duvarının (deponun x-ekseni boyunca uzunluğunun) orta noktasına yerleştirilir. Depo yatay duvarının orta noktasına yerleştirilmiş kapının sağ dikey duvar ile mesafesi, yatay duvarının uzunluğunun (L_x / 2) yarısıdır. Hizmet kalitesini ve malzeme çıkış miktarını arttırmak için yatay duvarda birden fazla kapı ihtiyacı söz konusu olduğunda, depo tasarımına yeni bir kapı

eklenebilir. İlave kapılar sayesinde hizmet almak için bekleyen taşıyıcı araç trafiği de azaltılmış olur.

Örneğin, 2*a* genişliğine sahip *d* adet depo kapısı olsun. Denklem 3.2'nin kullanılması ile deponun sol dikey depo duvarı ile k'nıncı kapının ortası arasındaki mesafe (l_k) elde edilir. Söz konusu k'nıncı depo kapısından malzeme yerleştirme veya alma için toplayıcının k'nıncı kapının sağına ve k'nıncı kapının soluna gitme olasılığı l_k /Lx ve $(Lx - l_k)/Lx$ olarak hesaplanır. Kapının sağına ve soluna ortalama taşıma mesafesi ise, deponun sol dikey depo duvarı ile k'nıncı kapının ortası arasındaki mesafenin yarısı ($l_k /2$) ve k'nıncı kapının ortasının sağında kalan depo boyunun mesafesinin yarısı ($(Lx - l_k)/2$) olacaktır. K'nıncı kapıyı kullanan toplayıcının x-ekseni boyunca kat edeceği ortalama mesafe Denklem 3.3'te gösterilmiştir. Ancak, söz konusu denklemde hesaplanan ortalama mesafe, sadece k'nıncı kapıyı kullanan toplayıcı için geçerlidir. Toplam toplayıcının ortalama mesafesi ise, her bir kapı için ayrı ayrı Denklem 3.3 kullanılarak elde edilen ortalama toplayıcı mesafelerinin toplamı şeklinde hesaplanmaktadır. Tüm kapıların denkleme ilave edilmesi ile x-ekseni boyunca kat edilen ortalama toplayıcı mesafesi Denklem 3.4'e göre hesaplanır (Önüt ve diğ., 2008).

$$l_k = (k\,(L_x + 2a)/d + 1\,) - a \tag{3.2}$$

$$T_X = (l_k\,/L_x)(\,l_k\,/2) + ((L_x - l_k)\,/L_x)(\,(L_x - l_k)\,/2) \tag{3.3}$$

$$T_X = -\frac{w_x\,(w_x + n_x\,(a_x + w_x/2))}{2\,n_x\,(a_x + w_x/2)} + \frac{(n_x(a_x + w_x/2) + 2w_x)^2}{n_x(a_x + w_x/2)\,(d+1)^2}\sum_{k=1}^{d} k^2 \tag{3.4}$$

İkinci ortalama toplama mesafesi, y-ekseni boyunca katedilen ortalama toplayıcı mesafesidir. Eğer tüm ürün gruplarına eşit olasılıkla erişim olsaydı, tek bir malzeme (palet) yerleştirme veya alma için toplayıcının katedeceği mesafe aşağıdaki Denklem 3.5'teki gibi olacaktı.

$$T_y = (a_y n_y/2) + w_y \tag{3.5}$$

Ancak, Denklem 3.5'te gösterilen matematiksel ifade, her ürün grubu farklı olasılığa sahip olduğundan kullanışsız olacaktır. Bu sebepten dolayı, Denklem 3.5'te verilen matematiksel ifade, her ürün grubunun farklı olasılığa sahip olduğu dikkate alınarak

yeniden düzenlenmelidir. Düzenlenecek matematiksel ifadenin daha kolay anlaşılabilmesi için, öncelikle beş farklı ürün grubu için hazırlanacak; daha sonra ürün grubu sayısından bağımsız olacak şekilde genelleştirilecektir. Bu bağlamda, farklı olasılığa sahip ürün gruplarının y-ekseni boyunca katettiği ortalama toplayıcı mesafesi şu şekilde hesaplanmaktadır (Önüt ve diğ., 2008).

$$T_Y = w_Y + P_A\,(n_{Ya}\,a_Y/2) + P_B\,(n_{Ya}\,a_Y + n_{Yb}\,a_Y/2) + P_C\,(n_{Ya}\,a_Y + n_{Yb}\,a_Y + nYc\ aY2 + PD\ nYa\ aY + nYb\ aY + nYc\ aY + nYd\ aY2 + PE\ nYa\ aY + nYb\ aY + nYc\ aY + nYd\ aY + nYe\ aY2 \quad (3.6)$$

Denklem 3.6'yı açıklanacak olursa, tüm A grubu ürünlerin olasılığı eşit olduğu için A ürün grubu için ortalama katedilen mesafe, A ürün grubu için ayrılmış rafın tam orta noktasına kadar olan mesafedir. B ürün grubunun ortalama mesafesi ise, A ürün grubu için ayrılmış raf mesafesi ile B ürün grubu için ayrılmış rafın tam ortasına kadar olan mesafelerin toplamıdır. Her bir ürün grubu için ortalama mesafe bu şekilde hesaplanırsa, yukarıdaki Denklem 3.6 elde edilmektedir. Daha sonra parantez önündeki olasılık değerleri parantez içine dağıtılması ile Denklem 3.7 elde edilmiş olur.

$$T_Y = w_Y + (P_A\,n_{Ya}\,a_Y/2) + P_B n_{Ya}\,a_Y + (P_B n_{Yb}\,a_Y/2) + P_C\,n_{Ya}\,a_Y + P_C\,n_{Yb}\,a_Y + (P_C\,n_{Yc}\,a_Y/2) + P_D\,n_{Ya}\,a_Y + P_D n_{Yb}\,a_Y + P_D\,n_{Yc}\,a_Y + (P_D n_{Yd}\,a_Y/2) + P_E\,n_{Ya}\,a_Y + P_E n_{Yb}\,a_Y + P_E\,n_{Yc}\,a_Y + P_E\,n_{Yd}\,a_Y + (P_E n_{Ye}\,a_Y/2) \quad (3.7)$$

$$T_Y = w_Y + n_{Ya}\,a_Y(P_A/2 + P_B + P_C + P_D + P_E) + n_{Yb}\,a_Y(P_B/2 + P_C + P_D + P_E) + n_{Yc}\,a_Y\,(P_C/2 + P_D + P_E) + n_{Yd}\,a_Y\,(P_D/2 + P_E) + n_{Ye}\,a_Y\,(P_E/2) \quad (3.8)$$

Denklem 3.7, her bir grup için y-ekseni boyunca ayrılmış stok pozisyonu değerine göre ortak çarpan parantezine alındığında Denklem 3.8 elde edilir. Denklemin genelleştirilmesi için $n_{Yb}\,a_Y(P_B/2 + P_C + P_D + P_E)$ matematiksel ifadesine sırası ile aşağıda verilen dönüşümler yapılır ve Denklem 3.9 elde edilir.

$$n_{Yb}\,a_Y\,(P_A + P_B + P_C + P_D + P_E) = n_{Yb}\,a_Y$$

$$P_A\,n_{Yb}\,a_Y + P_B\,n_{Yb}\,a_Y + P_C\,n_{Yb}\,a_Y + P_D\,n_{Yb}\,a_Y + P_E\,n_{Yb}\,a_Y = n_{Yb}\,a_Y$$

$$P_B\, n_{Yb}\, a_Y/2 + (P_A\, n_{Yb}\, a_Y/2 + P_B\, n_{Yb}\, a_Y/2 + P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y +$$
$$PE\ nYb\ aY = nYb\ aY - PA\ nYb\ aY2$$

$$(P_B\, n_{Yb}\, a_Y/2 + P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y) =$$
$$n_{Yb}\, a_Y\, (1 - P_A/2) - (P_B\, n_{Yb}\, a_Y/2 + P_A\, n_{Yb}\, a_Y/2)$$

(3.9)

Gerekli dönüşümler yapıldıktan sonra elde edilen matematiksel ifade (Denklem 3.9) yerine yazılır ve Denklem 3.10 elde edilir.

$$T_Y = w_Y + n_{Ya}\, a_Y(1 - P_A/2) + n_{Yb}\, a_Y\,(1 - P_A/2) - (P_B\, n_{Yb}\, a_Y/2 +$$
$$P_A\, n_{Yb}\, a_Y/2) + n_{Yc}\, a_Y\,(P_C/2 + P_D + P_E) + n_{Yd}\, a_Y\,(P_D/2 + P_E) +$$
$$n_{Ye}\, a_Y\,(P_E/2) \qquad (3.10)$$

Denklem 3.10 elde edildikten sonra, denklem içerisindeki $(P_B\, n_{Yb}\, a_Y/2 + P_A\, n_{Yb}\, a_Y/2)$ matematiksel ifade için Denklem 3.11'deki dönüşümler yapılıp; Denklem 3.10'daki yerine yerleştirilirse Denklem 3.12 elde edilmiş olur.

$$n_{Yb}\, a_Y\,(P_A + P_B + P_C + P_D + P_E) = n_{Yb}\, a_Y$$

$$(P_A\, n_{Yb}\, a_Y + P_B\, n_{Yb}\, a_Y + P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y) = n_{Yb}\, a_Y$$

$$P_A\, n_{Yb}\, a_Y + P_B\, n_{Yb}\, a_Y + P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y/2 = n_{Yb}\, a_Y/2$$

$$P_A\, n_{Yb}\, a_Y + P_B\, n_{Yb}\, a_Y/2 = n_{Yb}\, a_Y/2 - P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y/2$$

(3.11)

$$T_Y = w_Y + (n_{Ya}\, a_Y + n_{Yb}\, a_Y)(1 - P_A/2) - n_{Yb}\, a_Y/2 + P_C\, n_{Yb}\, a_Y +$$
$$P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y/2 + n_{Yc}\, a_Y\,(P_C/2 + P_D + P_E) + n_{Yd}\, a_Y\,(P_D/2 +$$
$$PE + nYe\ aY\ PE\ 2 \qquad (3.12)$$

Denklem 3.12 elde edildikten sonra $P_C\, n_{Yb}\, a_Y + P_D\, n_{Yb}\, a_Y + P_E\, n_{Yb}\, a_Y/2 + n_{Yc}\, a_Y\,(P_C/2 + P_D + P_E) + n_{Yd}\, a_Y\,(P_D/2 + P_E) + n_{Ye}\, a_Y\,(P_E/2)$ matematiksel ifadesinin parantez içerisi tekrar düzenlendiğinde Denklem 3.13 elde edilir.

$$T_Y =$$
$$w_Y + (n_{Ya}\, a_Y + n_{Yb}\, a_Y)(1 - P_A/2) - n_{Yb}\, a_Y/2 +$$
$$(n_{Yb}\, a_Y + n_{Yc}\, a_Y)\,(P_C + P_D + P_E)/2 + (n_{Yc}\, a_Y + n_{Yd}\, a_{Y)}\,(P_D/2 + P_E/2) +$$

$$(n_{Ye}\, a_Y + n_{Yd}\, a_Y)\,(P_E/2)$$

(3.13)

Tüm dönüşümler yapıldıktan sonra rahatlıkla genelleştirme yapılabilmektedir. Grup isimlerin yerine rakamlar kullanılırsa Denklem 3.14 dönüştürülür ve en genel hali ile Denklem 3.15 elde edilir.

$$T_Y = w_Y + (n_{Y1}\, a_Y + n_{Y2}\, a_Y)\,(1 - P_1/2) - n_{Y2}\, a_Y/2 + (n_{Y2}\, a_Y +$$

nY3 aYP3 +P4+ P5 *2*+(nY3 aY+ nY4 aY) P4+ P5 *2*+ (nY4 aY+nY5 aY) P5

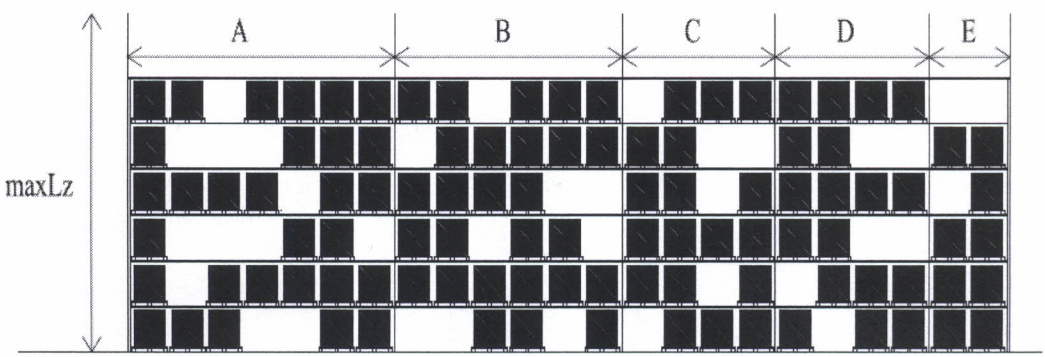
2 (3.14)

$$T_Y = w_Y + (n_{Y1}\, a_Y + n_{Y2}\, a_Y)(1 - P_1/2) - n_{Y2}\, a_Y/2 +$$

$$a_Y \sum_{i=1}^{gs-2} \left[(n_{Yi+1} + n_{Yi+2})/2 \sum_{i=1}^{gs-2} \frac{P_{i+2}}{2} \right] \qquad (3.15)$$

Denklem 3.15 ile farklı olasılığa sahip ürün gruplarının y-ekseni boyunca kat edilen ortalama toplayıcı mesafesi hesaplanmaktadır.

Sabit yükseklikli paletler için depo yerleşim düzenlemesinde her ürün grubunun yükseklikleri veya her ürün grubu için ayrılan stok pozisyonları birbirine eşittir. Bu durumda, her bir ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı eşit olacaktır. Sabit yükseklikli paletler için depo yerleşim düzenlemesine ait karşıdan görünüş Şekil 3.3'te verilmektedir.



Şekil 3.3: Sabit yükseklikli paletler için rafların karşıdan görünüşü.

Son ortalama mesafe, sabit yükseklikli paletlerin yerleşimi ve alınması için z-ekseni boyunca katlanılan ortalama toplayıcı mesafesidir. Şekil 3.3'te görüleceği gibi her bir

$$T_Z = a_z\, n_z/2 \tag{3.16}$$

Depo eksenleri boyunca elde edilen ortalama toplayıcı denklemler, sabit yüksekli paletler için depo yerleşim düzenlemesi probleminin modellenmesindeki amaç fonksiyonunu oluşturmaktadır. Söz konusu depo yerleşim düzenlemesinde kullanılan matematiksel modelin amaç fonksiyonu Denklem 3.17'de gösterilmiştir.

$$\begin{aligned} min\, z = {} & C_h Ps\,(w_Y + (n_{Y1}\, a_Y + n_{Y2}\, a_Y)(1 - P_1/2) - n_{Y2}\, a_Y/2 \\ & + a_Y \sum_{i=1}^{gs-2} \left[\frac{(n_{Yi+1} + n_{Yi+2})}{2} \sum_{i=1}^{gs-2} \frac{P_{i+2}}{2} \right] - \frac{w_x \left(w_x + n_x \left(\frac{a_x + w_x}{2} \right) \right)}{2\, n_x \left(\frac{a_x + w_x}{2} \right)} \\ & + \frac{(n_x(a_x + w_x/2) + 2w_x)^2}{n_x(a_x + w_x/2)\,(d+1)^2} \sum_{k=1}^{d} k^2 + a_z\, n_z/2 \end{aligned} \tag{3.17}$$

Sabit yüksekli paletler için depo yerleşim düzenlemesinin matematiksel modelinin kısıtları aşağıdaki şekilde ifade edilmektedir.

$$\sum_{i=1}^{gs} n_{Yi}\, n_Z\, n_X > Ni\,, \qquad \forall i \tag{3.18}$$

$$\sum_{i=1}^{gs} a_Y\, n_{Yi} + w_y < L_{Ymax} \qquad \forall i \tag{3.19}$$

$$(a_X + w_X/2)\, n_x < L_{Xmax} \tag{3.20}$$

$$a_Z\, n_z < L_{Zmax} \tag{3.21}$$

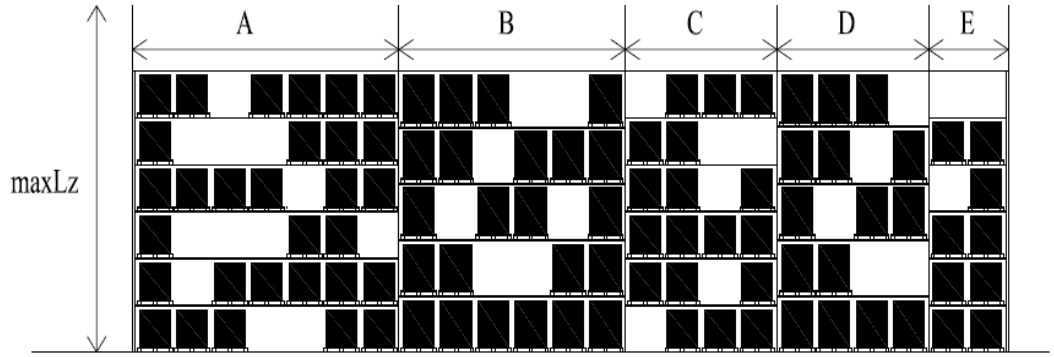
$$n_X,\, n_{Yi}\,, n_Z \in \{0\} \cup \mathbb{Z}^+ \qquad \forall i \tag{3.22}$$

3.18 numaralı kısıt denklemi, modelin kapasite kısıtıdır ve modelin depoda stoklanacak her ürün grubu için sağlaması gereken en az stok kapasitesini garanti etmektedir. 3.19, 3.20 ve 3.21 numaralı kısıt denklemleri ise, söz konusu depo yerleşim düzenlemesi için ayrılan alan sınırını ifade etmektedir. 3.19 numaralı kısıt denklemi, her bir ürün grubu için ayrılmış yan yana yerleşmiş rafların toplam genişliğinin deponun yerleşimi için ayrılan alanın enini geçmemesini göstermektedir. 3.20 numaralı kısıt denklemi, x-ekseni boyunca yerleştirilecek rafların sayısının depo yerleşimi için ayrılan alanın boyunu geçmemesini sağlamaktadır. 3.21 numaralı kısıt denklemi, z-ekseni boyunca yerleştirilecek rafların sayısının deponun çatı yüksekliğini geçmesini engelleyen ifadedir. Son kısıt denklemi olan 3.22 ise, tüm değişkenlerin pozitif tamsayı olmasını garanti etmektedir.

3.2.1.2. Değişken Yükseklikli Paletler için Depo Yerleşim Düzenlemesi Probleminin Modeli

Sabit yükseklik paletler için depo yerleşim düzenlemesi probleminin modelinde her bir ürün grubunun paletleri sabit yükseklikteydi. Ancak bu söz konusu model, değişken yüksekli paletlerin depolanmasında en uygun sonucu vermemektedir. Değişken yükseklikli paletlerin depolanmasında ortaya çıkan olumsuz durumun ortadan kaldırılması için bu bölümde değişken yükseklikli paletlerin depolanmasında kullanılacak depo yerleşim düzenlemesi modellenecektir.

Değişken yükseklikli paletler için depo yerleşim düzenlemesinde her ürün grubunun yükseklikleri veya her ürün grubu için kolondaki stok kapasitesi birbirine eşit değildir. Bu durumda, her bir ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı birbirine eşit olmayacaktır. Şekil 3.4'ten de görüleceği gibi her bir ürün grubu için ayrılmış stoklama alanlarının kolonlarında farklı sayıda stok pozisyonu bulunmaktadır.



Şekil 3.4: Değişken sabit yükseklikli paletler için rafların karşıdan görünüşü.

Bazı durumlarda, stok pozisyonları veya paletlerin ebatları stoklanan ürüne göre değişim göstermektedir. Örneğin, bebek maması ile bebek bezi aynı gruba ait olmasına rağmen bebek maması için ayrılan stok pozisyonu yüksekliği ile bebek bezi için ayrılan stok pozisyonu yüksekliği birbirine eşit olmamaktadır. Bu yükseklik farkları dikkate alınmadan tasarlanmamış depolarda bu tür ürünlerin stoklanması problemlere neden olmaktadır.

Doktora tezi kapsamında, palet yüksekliğinin değişkenliğinden ortaya çıkan problemlerin üstesinden gelebilmek için her bir ürün grubunun stok pozisyonunun

yüksekliği birbirlerine göre değişiklik gösterecek şekilde matematiksel model geliştirilecek ve yeni bir depo yerleşim düzenlemesi modellemesi önerilecektir.

Her bir ürün grubunun değişken yüksekliğe sahip olması ile z-ekseni boyunca toplayıcının ortalama mesafe denklemi (Denklem 3.16) değişecektir. Denklem 3.16 yerine, her bir ürün grubunun siparişte bulunma olasılığı dikkate alınarak Denklem 3.23 oluşturulmuştur. Elde edilen Denklem 3.23, amaç fonksiyonundaki yerine yerleştirilirse Denklem 3.24 elde edilmiştir.

$$\sum_{i=1}^{gs} {}_{a_{Zi}}\mathrm{P_i}\,\mathrm{z_i}/2 \tag{3.23}$$

$$min\, z = C_h Ps\,(w_Y + (n_{Y1}\,a_Y + n_{Y2}\,a_Y)(1 - P_1/2) - n_{Y2}\,a_Y/2 +$$

$$a_Y \sum_{i=1}^{gs-2}\left[\frac{(n_{Yi+1} + n_{Yi+2})}{2}\sum_{i=1}^{gs-2}\frac{P_{i+2}}{2}\right] - \frac{w_x\left(w_x + n_x\left(\frac{a_x + w_x}{2}\right)\right)}{2\,n_x\left(\frac{a_x + w_x}{2}\right)} +$$

$$\frac{(n_x(a_x + w_x/2) + 2w_x)^2}{n_x(a_x + w_x/2)\,(d+1)^2}\sum_{k=1}^{\mathrm{d}} k^2 + \sum_{i=1}^{gs} {}_{a_{Zi}}\mathrm{P_i}\,\mathrm{z_i}/2 \tag{3.24}$$

Modelin amaç fonksiyonunda yapıldığı olduğu gibi, kısıtlarda da birtakım değişiklikler yapılmıştır. 3.18 numaralı kapasite kısıtı, her bir ürün grubu için farklı kolon kapasitesine sahip olacağı için 3.25 olacak şekilde değişmiştir. Bunun yanı sıra, 3.21 numaralı yükseklik kısıtı, farklı ürün grupları farklı yüksekliklere sahip olabileceği için 3.28 haline dönüştürülmüştür. Buradaki n_{Zi} , notasyonu her bir ürün grubu için farklı kolon kapasitesini gösteren değişkendir. Son kısıt değişikliği ise, tüm depo yerleşim düzenlemesi belirlenmesinde kullanılan değişkenlerin pozitif olma kısıtıdır. Bir önceki Denklem 3.22'de tek yükseklik değişkeni var iken, 3.29 numaralı denklemde birden fazla yükseklik değişkeni bulunmaktadır.

Değişen kısıtlar ile birlikte tüm kısıt denklemleri aşağıdaki şekildedir.

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{gs}} \mathrm{n_{Yi}}\,\mathrm{n_{Zi}}\,\mathrm{n_X} > Ni\,, \qquad \forall \boldsymbol{i} \tag{3.25}$$

$$\sum_{i=1}^{gs} a_Y\,\mathrm{n_{Yi}} + \mathrm{w_y} < L_{Ymax} \qquad \forall \boldsymbol{i} \tag{3.26}$$

$$(a_X + \mathrm{w}_X/2)\,n_x < L_{Xmax} \tag{3.27}$$

$$a_{Zi}\,n_{zi} < L_{Zmax}\,, \qquad \forall \boldsymbol{i} \tag{3.28}$$

$$n_X,\, n_{Yi}\,, n_{Zi} \in \{0\} \cup \mathbb{Z}^+ \qquad \forall i \tag{3.29}$$

Tüm kısıtlar belirlendikten sonra 3.26, 3.27 ve 3.28 numaralı kısıt denklemlerinin sağ tarafındaki depo yerleşim düzenlemesi yapılacak alanın maksimum uzunluklarının, maksimum yerleşebilecek raf sayısına dönüştürülmesi gerekmektedir. Söz konusu dönüşüm, PSO algoritmasında da kullanılacaktır.

$$L_{Xmax}/(a_X + \mathrm{w_x}/2) = [\![n_{Xmax}]\!] \tag{3.30}$$

$$P_{\mathrm{i}}(L_{Ymax} - \mathrm{w_y})/a_Y = [\![n_{Yi-max}]\!] \qquad \forall \boldsymbol{i} \tag{3.31}$$

$$L_{Zmax}/a_{Zi} = [\![n_{Zi-max}]\!], \qquad \forall \boldsymbol{i} \tag{3.32}$$

3.30, 3.31 ve 3.32 numaralı denklemlerin tümünde her bir boyut ve her bir ürün grubu için maksimum raf sayısının tamsayı olması gerekmektedir. Bu yüzden hesaplamalarda her bir değişkenin maksimum değeri tamsayı olacak şekilde belirlenmektedir.

$$n_{Xmin} = 1 \tag{3.33}$$

$$n_{Yi-min} = 1 \qquad \forall \boldsymbol{i} \tag{3.34}$$

$$n_{Zi-min} = 1 \qquad \forall \boldsymbol{i} \tag{3.35}$$

3.33, 3.34 ve 3.35 numaralı denklemler, her bir depo boyutunun ve her bir grubun minimum raf sayısını belirtmektedir. Herhangi bir alt sınır bulunmadığı durumlarda bu değerler kullanılabilir.

3.2.2. Depo Yerleşim Düzenlemesi Problemi Modeli Çözümü için Parçacık Sürü Algoritması

Depo yerleşim düzenlemesi problemi modelinin çözümü için kullanılan orjinal PSO algoritmasının adımları aşağıda verilmektedir (Eberhart ve Kennedy, 1995).

Adım 1: D boyutlu rassal konumlara ve hız vektörüne sahip parçacıklardan oluşan başlangıç popülasyonu oluşturulur.

Adım 2: D boyutlu her bir parçacık için uygunluk değeri belirlenir.

Adım 3: Her bir parçacık için elde edilen uygunluk değeri, parçacığın daha önceki en iyi uygunluk değeri ile karşılaştırılır. Bu son değer, daha önceki değerden daha iyiyse

yeni değer en iyi uygunluk değeri olarak güncellenir. Son uygunluk değerini sağlayan parçacığın konumlarına göre parçacığın en iyi uygunluk değerini sağlayan konumları da güncellenir.

Adım 4: Sürünün en iyi uygunluk değeri, daha önceki popülasyonun en iyi uygunluk değeri ile karşılaştırılır. Son sürünün uygunluk değeri, daha önceki değerden daha iyiyse yeni değer sürünün en iyi uygunluk değeri olarak güncellenir. Sürünün en iyi uygunluk değerini sağlayan konumlar da güncellenir.

Adım 5: 3.36 ve 3.37 numaralı denklemleri kullanarak parçacıkların konumları ve hızları güncellenir.

$$v_{i,d}^{k+1} = w_k\, v_{i,d}^{k} + c_1 r_1\left(p_{i,d}^{k} - x_{i,d}^{k}\right) + c_2 r_2\left(g_d^{k} - x_{i,d}^{k}\right) \tag{3.36}$$

$$x_{i,d}^{k+1} = x_{i,d}^{k} + v_{i,d}^{k+1} \tag{3.37}$$

Adım 6: Hedeflenen durdurma kriteri oluşuncaya kadar Adım 2 ile Adım 5 arasındaki işlemler tekrar edilir.

PSO yaklaşımında parçacıkların çözüm uzayındaki hareketi için kullanılan hesaplamalar ve bu hesaplamalarda kullanılan notasyonlar aşağıda gösterilmiştir.

$v_{i,d}^{k}$: i. bireyin d. boyutu k. iterasyon hızı

$x_{i,d}^{k}$: i. bireyin d. boyutunun k. iterasyondaki konumu

$p_{i,d}^{k}$: i. bireyin d. boyutunun k. iterasyondaki bireysel en iyi değeri

g_d^{k} : d. boyutun k. iterasyondaki global en iyi değeri

$c_1 ve\ c_2$: Bilişsel ve sosyal parametreler

$r_1 ve\ r_2$: [0,1] aralığında uniform dağılmış rastgele sayılar

w_k : k. iterasyondaki atalet ağırlığı

$iter_{max}$: maksimum iterasyon sayısı

v_{max} : Hız vektörünün maksimum değeri

v_{min} : Hız vektörünün minimum değeri

D : Parçacıkların boyut sayısı

Doktora tezinin konusu olan depo yerleşim düzenlemesi probleminin çözümü için kullanılan PSO algoritmasının nasıl uygulandığını gösteren adımlar aşağıda açıklanmaktadır.

Adım 1: Bu adım, her bir parçacığın *D* rassal boyutu kadar konumlarının ve hızlarının belirlendiği aşamadır. Aşağıdaki denklemlerin yardımıyla parçacıkların rassal konumları ve hızları belirlenir. *D* rassal boyutunun sayısı ise, depolanacak grup sayısının iki katının bir fazlasıdır.

$$x_{i,d}^{0} = rand\,(n_{Yd-min}, n_{Yd-max}) \quad d = 1,2,\dots gs \tag{3.38}$$

$$x_{i,(d+gs)}^{0} = rand\,(n_{Zd-min}, n_{Zd-max}) \quad d = 1,2,\dots gs \tag{3.39}$$

$$x_{i,(2gs+1)}^{0} = rand\,(n_{Xmin}, n_{Xmax}) \tag{3.40}$$

$$v_{i,d}^{0} = rand\,(v_{min}, v_{max}) \quad d = 1,2,\dots .D \tag{3.41}$$

3.38 numaralı denklemde yer alan parçacığın ilk grup sayısı kadar olan değişkenler, depo yerleşim düzenlemesinde stoklanacak ürün gruplarının y-ekseni boyunca ayrılan stok pozisyonu sayısını göstermekte ve 3.38 numaralı denklemin kullanılması ile söz konusu değişkenlerin sayıları belirlenmektedir. 3.39 numaralı denklemde yer alan parçacığın ikinci grup sayısı kadar olan değişkenler, ürün gruplarının z-ekseni boyunca ayrılan stok pozisyonu sayısını göstermekte ve 3.39 numaralı denklemin kullanılması ile söz konusu değişkenlerin sayıları belirlenmektedir. 3.40 numaralı denklemde yer alan parçacığın en son değişkeni, depo yerleşim düzenlemesinde x-ekseni boyunca raf sayısını göstermekte ve 3.40 numaralı denklemin kullanılması ile söz konusu değişkenlerin sayısı belirlenmektedir. 3.38, 3.39 ve 3.40 numaralı denklemler kullanılarak parçacıkların rassal konumları belirlenir. 3.41 numaralı denklem kullanıldığında ise parçacıkların rassal hızları vektörleri belirlenir.

Adım 2: Bu adımda, Adım 1'de üretilen başlangıç değerleri Denklem 3.42'de yerine konularak her bir parçacığın uygunluk değeri hesaplanır.

$$f^{\mathrm{k}}(x_{\mathrm{i}}) = C_h Ps\,(w_Y + (x^0_{i,1}a_Y + x^0_{i,2}a_Y)(1 - P_1/2) - x^0_{i,2}a_Y/2 + a_Y \sum_{l=1}^{gs-2}\left[\frac{(x^0_{i,l+1} + x^0_{i,l+2})}{2}\sum_{\mathrm{l}=1}^{gs-2}\frac{P_{\mathrm{l}+2}}{2}\right] - \frac{w_x\left(w_x + x^0_{i,(2gs+1)}\left(\frac{a_x+w_x}{2}\right)\right)}{2\,x^0_{i,(2gs+1)}\left(\frac{a_x+w_x}{2}\right)} + \frac{(x^0_{i,(2gs+1)}(a_x + w_x/2) + 2w_x)^2}{x^0_{i,(2gs+1)}(a_x + w_x/2)\,(r+1)^2}\sum_{k=1}^{d}k^2 + \sum_{n=1}^{gs}\mathrm{a_{Zn}}\,\mathrm{P_n}\,x^0_{i,(n+gs)}/2 \tag{3.42}$$

Adım 3: Adım 3 iki aşamadan oluşmaktadır. Uygulanacak adım, iterasyon sayısına göre değişmektedir.

Adım 3.1: İterasyon sayısı sıfıra eşit olduğunda ($k = 0$) bu adım gerçekleştirilir. Aksi durumda ise adım 3.2'ye geçilir. Her bir parçacık için Adım 2'de hesaplanan uygunluk değerleri başlangıç uygunluk değerleri olarak belirlenir (Denklem 3.43) ve Adım 4'e geçilir. Parçacığa ait boyutların değeri, bireysel en iyi boyutlarına atanır.

$$p^0_{i,d} = x^0_{i,d} \quad , \forall \boldsymbol{d} \tag{3.43}$$

Adım 3.2: İterasyon sayısının sıfıra eşit olmadığı durumda ($k \neq 0$), her bir parçacık için elde edilen uygunluk değeri, parçacığın daha önceki en iyi uygunluk değeri ile karşılaştırılır. Son uygunluk değeri, daha önceki uygunluk değerinden daha iyiyse yeni uygunluk değeri en iyi uygunluk değeri olarak (Denklem 3.44) güncellenir. Parçacığa ait boyutlar kullanılarak, parçacığın bireysel en iyi boyutları da güncellenir.

$$p^k_{i,d} = \begin{cases} x^k_{i,d}, & f^{\mathrm{k}}(x_{\mathrm{i}}) \leq f^{\mathrm{k}-1}(x_{\mathrm{i}}) \\ p^{k-1}_{i,d}, & Aksi\ durumda \end{cases} \quad , \forall \boldsymbol{d} \tag{3.44}$$

Adım 4: Bu adım da, tıpkı Adım 3'te olduğu gibi iki aşamalıdır. Söz konusu adımda, iterasyon sayısına bağlı olarak izlenecek algoritma adımı değişecektir.

Adım 4.1: İterasyon sayısı sıfıra eşit olduğunda ($k = 0$), her bir parçacığın uygunluk değeri karşılaştırılır. En iyi uygunluk değerine sahip olan parçacığın uygunluk değeri global en iyi uygunluk değeri olarak belirlenir (Denklem 3.45). Parçacığın tüm boyutları, sürünün en iyi boyutlarına atanır.

$$g_{i,d}^{0} = x_{best,d}^{0} \qquad \forall \boldsymbol{d} \tag{3.45}$$

Adım 4.2: İterasyon sayısının sıfıra eşit olmadığı durumda ($k \neq 0$), sürüdeki her bir parçacık için elde edilen uygunluk değerlerinden en iyisi, daha önceki sürünün en iyi değeri ile karşılaştırılır. Son uygunluk değeri, daha önceki uygunluk değerinden daha iyiyse, yeni değer en iyi uygunluk değeri olarak revize edilir. Denklem 3.46'daki süreç uygulanır. Sürünün en iyi değerine sahip parçacığın boyutları kullanılarak sürünün en iyi boyutları güncellenir.

$$g_{i,d}^{k} = \begin{cases} x_{i,d}^{k}, & f^{\mathrm{k}}(x_{\mathrm{best}}) \leq f^{\mathrm{k-1}}(x_{\mathrm{best}}) \\ g_{i,d}^{k-1}, & Aksi\ durumda \end{cases}, \quad \forall \boldsymbol{d} \tag{3.46}$$

Adım 5: 3.36 ve 3.37 numaralı denklemler kullanılarak parçacıkların konumları ve hızları revize edilir. Ancak, hız denkleminde değişiklik yapılmıştır. Bu değişikliğin nedeni, parçacıkların hızları çok yükseldiğinde veya çok azaldığında parçacıkların hareketleri kararsızlaşmasıdır. Hızların kararsızlığını engellemek için, parçacıkların hız vektörleri belirli bir sınırın üstüne çıktığında veya belirli bir sınırın altına indiğinde hızlar sınırlandırılır. Denklem 3.47 ile hız sınırlamasının nasıl yapılacağı gösterilmiştir.

$$v_{i,d}^{k+1} = \begin{cases} v_{min}, & v_{i,d}^{k+1} < v_{min} \\ v_{i,d}^{k+1}, & v_{min} \leq v_{i,d}^{k+1} \leq v_{max} \\ v_{max}, & v_{max} < v_{i,d}^{k+1} \end{cases}, \quad \forall \boldsymbol{d} \tag{3.47}$$

Adım 6: Hedeflenen durdurma kriteri oluşuncaya kadar Adım 2 ile Adım 5 arasındaki işlemler tekrarlanır.

Yukarıda PSO algoritmasının tüm adımları takip edilerek, sabit ve değişen yükseklikli paletler için depo yerleşim düzenlemesi probleminin modellerine uygulanması gösterilmiştir.

3.2.3. Depo Yerleşim Düzenlemesi Problemi Modeli Çözümü için Parçacık Sürü Algoritmasının Sınırlandırma Yöntemleri

Kısıtların varlığı, kısıtsız problemlerde kullanılan sezgisel optimizasyon algoritmalarının performanslarını önemli ölçüde olumsuz etkilemektedir. Bu sezgisel

yöntemlerin daha iyi sonuç vermesi için farklı stratejiler kullanılmaktadır (Zhang ve Xie, 2009).

- **Ceza fonksiyonları tabanlı yöntemler:** Ceza fonksiyonu yöntemlerinde, sürüdeki parçacıklar uygun çözüm uzayının dışına çıktığında veya uygun bir çözüm üretemediğinde, kısıtların aşım miktarlarına bağlı olarak bazı ceza değerleri amaç fonksiyonuna ilave edilmektedir. Ceza değeri belirlenmesi önemli bir noktadır. Ceza sabitleri çok düşük seçilirse, algoritma uygun olmayan uzayda arama yapabilir ve uygun olmayan çözümler üretebilir. Diğer taraftan, ceza sabiti çok büyük seçilirse, yerel optimuma takılarak en uygun çözümü geliştiremeyebilir (Zhang ve Xie, 2009).
- **Geçerli çözümlerin korunması tabanlı yöntemler:** Bu yöntemlerde parçacıkların uygun çözüm uzayının dışına çıkmasına izin verilmektedir. Ancak, uygun olmayan çözümler üretildiği zaman elde edilen sonuçlar değerlendirilmez. Sadece uygun çözüm içeren sonuçlar değerlendirilir. Bu yöntemlerin zayıf yönü, parçacıkların uygun olmayan çözümlerde dolaşması ve problemin çözümü için daha çok zamana ihtiyaç duymasıdır.
- **Geçerli çözümlerde arama yapılmayı sağlayan yöntemler:** Bu yöntemde, çözüm uzayı dışına çıkan parçacıklar değerlendirme yapılmadan önce çözüm uzayı içerisine çekilmektedir. Bu durumda, ortaya çıkan tüm sonuçlar uygun değerler içerisinde olacaktır. Ancak, bu yöntemde karşılaşılan en önemli sorun parçacıkların çözüm uzayının çok küçülmesi ve yerel uygunluklara kolayla yakalanmasıdır (Coath ve Halgamuge, 2003).
- **Hibrit algoritmalar:** Hibrit algoritmalar ise, yukarıdaki yöntemler ile diğer sezgisel algoritmaların birleştirilmesi ile elde edilen yöntemlerdir.

PSO algoritması, diğer birçok sezgisel optimizasyon algoritmalarında olduğu gibi kısıtların varlığında etkin şekilde kullanılamamaktadır. Ancak, PSO algoritmasının kısıtlar altında daha iyi sonuç vererek çalışabilmesi için birtakım yöntemler geliştirilmiştir. Bu bölümün devamında, PSO algoritması için var olan sınırlandırma yöntemleri gösterilecek ve iki yeni sınırlandırma yöntemi önerilecektir.

3.2.3.1. PSO Algoritmasında Kullanılan Mevcut Sınırlandırma Yöntemleri

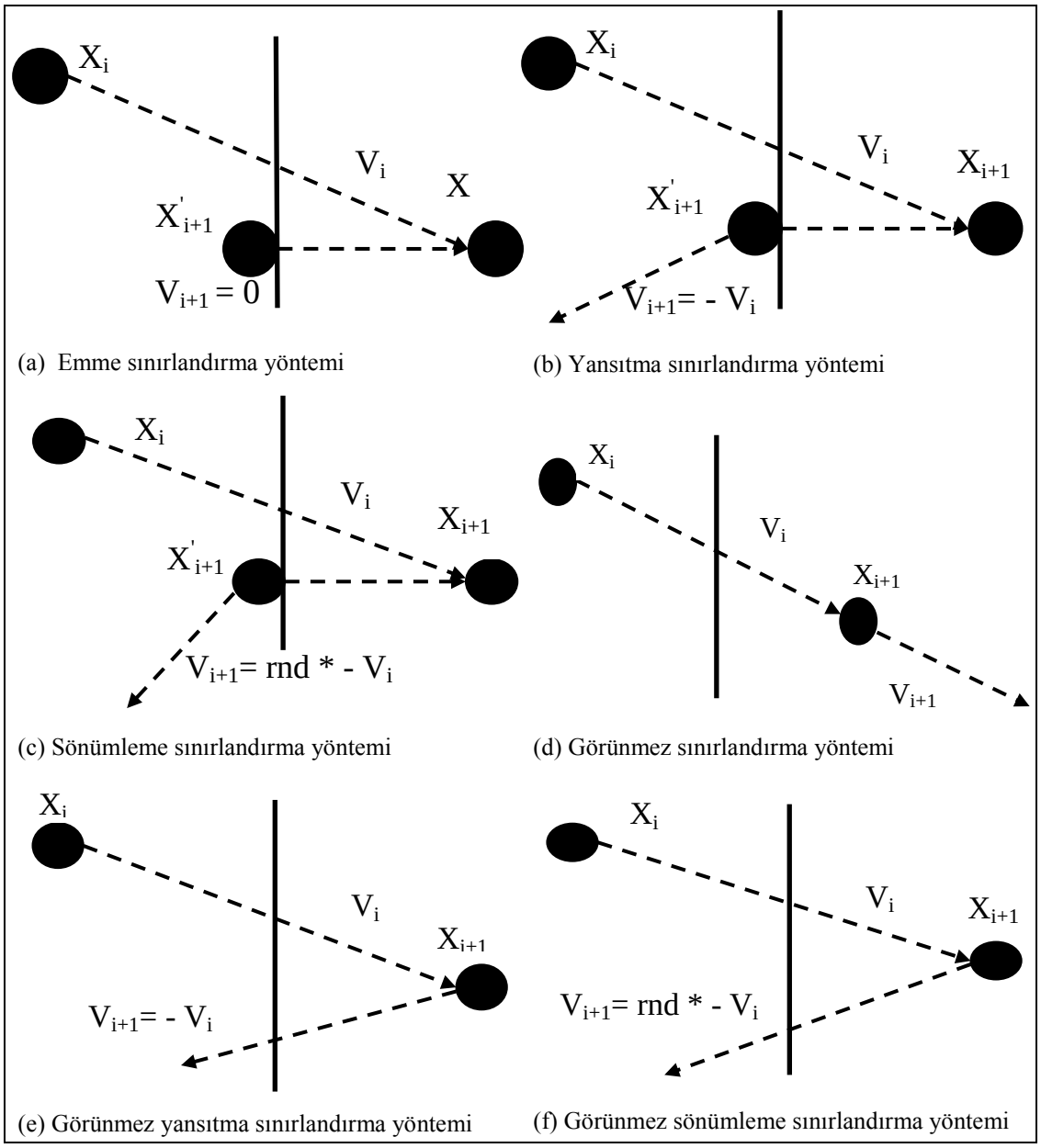
Bu bölümde, literatürde karşılaşılan sınırlandırma yöntemleri anlatılacaktır. Bu yöntemler emme sınırlandırma, yansıtma sınırlandırma, sönümleme sınırlandırma, görünmez sınırlandırma, görünmez yansıtma sınırlandırma ve görünmez sönümleme sınırlandırma yöntemleridir.

Emme sınırlandırma yöntemi: Parçacığın herhangi bir boyutu, kabul edilebilir çözüm uzayının dışına doğru hareket ederse sınırı aşan parçacığın çözüm uzayını aşan boyutu kabul edilebilir çözüm uzayı sınırına çekilir ve söz konusu boyuttaki hızı sıfıra indirilir. Emme sınırlandırma yöntemi Şekil 3.5 (a)'da gösterilmektedir (Li ve diğ., 2010).

Yansıtma sınırlandırma yöntemi: Bu yöntemde, parçacığın herhangi bir boyutu kabul edilebilir çözüm uzayı dışına çıkarsa parçacığın hem hızı hem de konumu değiştirilir. Sınırı aşan boyutun konumu kabul edilebilir çözüm uzayı sınırına çekilir ve parçacığın hızı tersine çevrilir. Yansıtma sınırlandırma yöntemi Şekil 3.5 (b)'de gösterilmektedir (Li ve diğ., 2010).

Sönümleme sınırlandırma yöntemi: Bu yöntemde, parçacığın herhangi bir boyutu kabul edilebilir çözüm uzayının dışına hareket ettiğinde parçacığın hem hızı hem de konumu değiştirilir. Yansıtma yönteminde olduğu gibi parçacığın sınırı aşan boyutu için konumu kabul edilebilir çözüm uzayı sınırına ötelenir ve hızı tersine çevrilir. Bu yöntemin yansıtma sınırlandırma yönteminden farkı, hızı tersine çevrilirken]0, 1[arasında rassal bir sayı ile çarpılmasıdır. Sönümleme sınırlandırma yöntemi Şekil 3.5 (c)'de gösterilmektedir (Li ve diğ., 2010).

Görünmez sınırlandırma yöntemi: Bu yöntem, bundan önceki yöntemlerden tamamen farklıdır. Parçacığın kabul edilebilir çözüm uzayı sınırını aşan boyutu olduğunda konumuna veya hızına herhangi bir müdahalede bulunulmamaktadır. Müdahale olmamasına rağmen, parçacığın kabul edilebilir çözüm uzayı içinde bulunmayan çözümü uygunluk değeri parçacığına atanır. Parçacığın çözüm uzayı içinde geri dönmesi ancak bireysel en iyi ve sürünün en iyisi ile gerçekleşebilir. Görünmez sınırlandırma yöntemi Şekil 3.5 (d)'de gösterilmektedir (Li ve diğ., 2010).



Şekil 3.5: PSO algoritmasında kullanılan sınırlandırma yöntemleri.

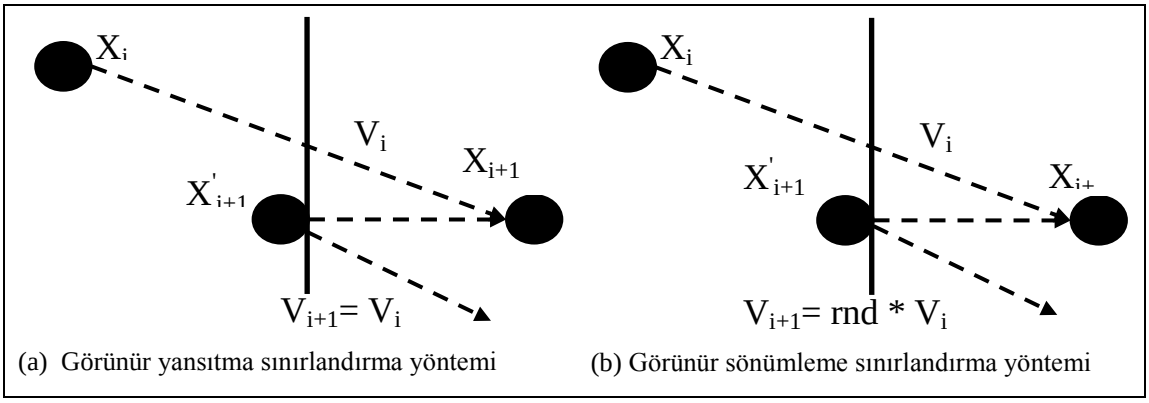
Görünmez yansıtma sınırlandırma yöntemi: Bu yöntem, yansıtma yöntemine benzemektedir. Yansıtma yönteminde olduğu gibi parçacığın kabul edilebilir çözüm uzayı sınırını aşan boyuttaki hızı tersine çevrilir. Ancak, parçacığın konumu yansıtma yönteminde sınıra çekilirken; görünmez yansıtma sınırlandırma yönteminde konuma müdahale edilmez. Görünmez yansıtma sınırlandırma yöntemi Şekil 3.5 (e)'de gösterilmektedir (Xu ve Rahmat-Samii, 2007).

Görünmez sönümleme sınırlandırma yöntemi: Bu yöntem, görünmez yansıtma sınırlandırma yöntemine benzemektedir. Görünmez sönümleme sınırlandırma

yönteminde kabul edilebilir çözüm uzayı sınırını aşan boyuttaki hız tersine çevrilirken;]0, 1[arasında rastgele üretilen bir sayı ile çarpılarak güncellenir. Görünmez sönümleme sınırlandırma yöntemi Şekil 3.5 (f)'de gösterilmektedir (Xu ve Rahmat-Samii, 2007).

3.2.3.2. PSO Algoritması için Önerilen Sınırlandırma Yöntemleri

Bu bölümde, doktora tezi kapsamında kullanılan PSO algoritması için iki yeni sınırlandırma yöntemi önerilmiştir. Önerilen yöntemlerin mevcut yöntemlere göre avantajı, parçacıkların çözüm uzayı sınırlarını sürekli zorlamasıdır. Önerilen sınırlandırma yöntemlerinde parçacıklar kabul edilebilir çözüm uzayı sınırına daha yakın arama yaptıkları için, daha iyi sonuç verme ihtimalleri yükselecektir. Önerilen sınırlandırma yöntemleri, "*görünür yansıtma sınırlandırma yöntemi*" ve "*görünür sönümleme sınırlandırma yöntemi*" olarak adlandırılmıştır.



(a) Görünür yansıtma sınırlandırma yöntemi (b) Görünür sönümleme sınırlandırma yöntemi

Şekil 3.6: Yeni parçacık sürü algoritmasının sınırlandırma yöntemleri önerisi.

Görünür yansıtma sınırlandırma yöntemi: Bu yöntem, görünmez sınırlandırma yöntemi ile yansıtma sınırlandırma yöntemi kullanılarak geliştirilmiştir. Bu yöntemde, yansıtma sınırlandırma yönteminde olduğu gibi kabul edilebilir çözüm uzayı sınırını aşan boyut, kabul edilebilir çözüm uzayı sınırına çekilmektedir. Diğer taraftan, görünmez sınırlandırma yönteminde olduğu gibi parçacığın hızı değiştirilmemektedir. Görünür yansıtma sınırlandırma yöntemi Şekil 3.6 (a)'da gösterilmektedir.

Görünür sönümleme sınırlandırma yöntemi: Bu yöntem görünmez sınırlandırma yöntem, emme sınırlandırma yöntemi ve sönümleme sınırlandırma yöntemi kullanılarak geliştirilmiştir. Bu yöntemde, emme sınırlandırma yönteminde olduğu gibi sınırı aşan boyut kabul edilebilir çözüm uzayı sınırına çekilir. Ayrıca, sönümleme sınırlandırma

yönteminde olduğu gibi hız rassal bir değer ile çarpılarak değiştirilir ve rassal sayı ile çarpılan hız, görünmez sınırlandırma yönteminde olduğu gibi müdahale edilmeyerek kabul edilebilir çözüm uzayı dışına doğru hareket etmesine izin verilir. Görünür sönümleme sınırlandırma yöntemi Şekil 3.6 (b)'de gösterilmektedir.

3.2.4. Depo Yerleşim Düzenlemesi Probleminin Çözümü için Atama Algoritma Önerisi

PSO algoritmasının ilk aşaması sürüdeki parçacıkların çözüm uzayı içerisinde rassal dağıtılması aşamasıdır. Ancak, sürüdeki parçacıkların çözüm uzayına dağıtılması sırasında tüm değişkenler için gerekli olan minimum ve maksimum değerlerin belirlenmesi gerekmektedir. Minimum ve maksimum değerlerin belirlenmesi PSO algoritması için çok önemli bir adımdır. PSO algoritmasının uygun sonuç verebilmesi için ilk iterasyonda sürünün en iyi değeri belirlenmeli ve belirlenen değere göre parçacıklar hareket etmelidir. İlk iterasyon sonunda sürünün en iyisi ve parçacıkların bireysel en iyi değeri belirlenmez ise PSO algoritması çözüme ulaşmada başarısız olmaktadır.

PSO algoritmasının başarısızlığının engellenmesi, değişken yükseklikli paletler için depo yerleşim düzenlemesi probleminin çözümünde uygun çözüm uzayının daraltılması ve üretilecek rassal değerlerin uygun çözüm uzayı içinde bulunması için *"parçacıkların başlangıç minimum değerleri için atama algoritması"* önerilmiştir. Doktora tez çalışmasının ilerleyen bölümlerinde, söz konusu parçacıkların başlangıç minimum değerleri için atama algoritması kısaca *"atama algoritması"* olarak adlandırılacaktır. Bu atama algoritmasının kullanılması ile PSO algoritmasındaki parçacıkların başlangıç minimum değerleri belirlenecektir. Atama algoritmasının kullanılması sayesinde, parçacıklar çözüm uzayına daha dar alandan atanmakta ve ilk iterasyon sonunda sürünün en iyisi ve parçacıkların bireysel en iyi değeri belirlenmektedir.

Atama algoritmasının adımları aşağıda açıklanmaktadır.

Adım 1: Tüm değişkenlerin eşitlikleri belirlenir. 3.48 numaralı denklemler kullanılarak tüm değişkenlerin eşitlikleri hesaplanır.

$$Ty_i = n_{Yi}\, a_Y \left(\frac{P_i}{2} + \sum_{i=i+1}^{gs} Pi\right) \qquad i = 1,2,..gs$$

$$T_X = -\frac{w_x\,(w_x + n_x\,(a_x + w_x/2))}{2\,n_x\,(a_x + w_x/2)} + \frac{(n_x(a_x + w_x/2) + 2w_x)^2}{n_x(a_x + w_x/2)\,(r+1)^2} \sum_{i=1}^{D} d^2$$

$$T_{Zi} = a_{Zi} Pi\, z_i/2 \qquad i = 1,2,..gs \qquad (3.48)$$

Adım 2: Bir önceki adımda belirlenen eşitliklere göre her bir değişkenin eğimi hesaplanır.

Adım 3: Hesaplanan eğimlerin karşılıklı olarak oranları alınır. 3.49 numaralı denklemler yardımıyla tüm geçiş değerleri hesaplanır.

$$\text{Geçiş } X/\,Y_i = \text{Eğim } T_X / \text{ Eğim } T_{Yi} \qquad i = 1,2,..gs$$

$$\text{Geçiş } X/\,Z_i = \text{Eğim } T_X / \text{ Eğim } T_{Zi} \qquad i = 1,2,..gs$$

$$\text{Geçiş } Y_i\,/\,Z_i = \text{Eğim } T_{Yi} / \text{ Eğim } T_{Zi} \qquad i = 1,2,..gs \qquad (3.49)$$

Adım 4: Hesaplanan geçiş oranlarının ortalaması alınır. Denklem 3.50 ile ortalama geçiş değerleri hesaplanır.

$$\overline{X/\,Y} = \sum_{i=1}^{gs} (X/\,Y_i)/\,gs \qquad i = 1,2,..gs$$

$$\overline{X/\,Z} = \sum_{i=1}^{gs} (X/\,Z_i)/\,gs \qquad i = 1,2,..gs$$

$$\overline{Y/\,Z} = \sum_{i=1}^{gs} (Y_i/\,Z_i)/\,gs \qquad i = 1,2,..gs \qquad (3.50)$$

Adım 5: Stoklanacak tüm ürün grupları için gerekli olan stok kapasitesinin ortalaması alınır. Elde edilen değer, Ortalama $\overline{X/\,Y}$, Ortalama $\overline{X/\,Z}$ ve Ortalama $\overline{Y/\,Z}$ çarpımına bölünür ve çıkan değerin 3. dereceden kökü alınır. Elde edilen değer, x-ekseni boyunca raf sayısını (n_X) verir. Eğer raf sayısı değeri n_{Xmax} değerinden büyükse, n_{Xmax} değerine eşitlenir; diğer durumlarda ise raf sayısı aynı şekilde kalır. Bu adımda kullanılan hesaplamalar Denklem 3.51'de gösterilmiştir.

$$nx = (\overline{N_i} \,/\, (\overline{X/\,Y} * \overline{X/\,Z} * \overline{Y\,/\,Z}))\ \hat{}\,(1/3)$$

$$n_X = \begin{cases} n_{Xmax}, & n_{Xmax} < n_X \\ n_X, & other \end{cases} \qquad (3.51)$$

Adım 6: Stoklanacak her ürün grubunun gerekli olan stok sayıları, x-ekseni boyunca raf sayısı (n_X) sayısı ile Geçiş Y_i / Z_i sayısının çarpımına bölünür ve elde edilen değerin karekökü alındığında her ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı belirlenir. Bu adımda kullanılan hesaplama Denklem 3.52'de verilmiştir.

$$n_{Zi} = (N_i / (n_X * \text{Geçiş } Y_i / Z_i)) \text{ ^}(1/2) \quad i = 1,2,..gs \tag{3.52}$$

Adım 7: Her bir ürün grubu sayısı, x-ekseni boyunca raf sayısı (n_X) sayısı ile her ürün grubu için z-ekseni boyunca ayrılan stok pozisyonu sayısı (n_{Zi}) çarpımına bölünürse her ürün grubu için y-ekseni boyunca ayrılan stok pozisyonu sayısı (n_{Yi}) elde edilir. Denklem 3.53 yardımıyla tüm y-ekseni boyunca ayrılan stok pozisyonu değerleri hesaplanır.

$$n_{Yi} = N_i / (n_X * n_{Zi}) \qquad i = 1,2,..gs \tag{3.53}$$

Adım 8: Belirlenen tüm değişkenler kısıtları sağlar ise, Adım 11'e geçilir. Eğer tüm kısıtlar sağlanmazsa ve x-ekseni boyunca raf sayısı, maksimum x-ekseni boyunca raf sayısından (n_{Xmax}) küçükse, x-ekseni boyunca raf sayısı arttırılır ve Adım 6'ya geri dönülür. Eğer tüm kısıtlar sağlanmazsa ve x-ekseni boyunca raf sayısı, maksimum x-ekseni boyunca raf sayısına (n_{Xmax}) eşitse bir sonraki adıma geçilir.

Adım 9: Eğer tüm kısıtlar sağlanırsa Adım 11'e geçilir. Eğer tüm kısıtlar sağlanmazsa ve sağlanmayan kısıtların karşılığı olan ürün gruplarının z-ekseni boyunca ayrılan stok pozisyonu sayıları (n_{Zi}), söz edilen gruplar için z-ekseni boyunca yerleştirilebilecek maksimum stok pozisyonu sayılarından (n_{Zi}) küçükse, stok pozisyonu sayısı (n_{Zi}) arttırılır ve Adım 7'ye geri dönülür. Ancak, kısıtı sağlamayan gruplar için z-ekseni boyunca yerleştirilebilecek stok pozisyonu sayısı, maksimum stok pozisyonu sayılarına eşit ise bir sonraki adıma geçilir.

Adım 10: Eğer tüm kısıtlar sağlanırsa Adım 11'e geçilir. Eğer tüm kısıtlar sağlanmazsa ve sağlamayan kısıtların karşılığı olan ürün gruplarının y-ekseni boyunca ayrılan stok pozisyonu sayıları (n_{Yi}), kısıtı sağlamayan ürün grupları için y-ekseni boyunca yerleştirilebilecek maksimum stok pozisyonu sayılarından (n_{Yi}) küçükse, stok pozisyonu sayısı (n_{Yi}) arttırılır ve Adım 7'ye geri dönülür. Ancak o gruplar için y-

ekseni boyunca yerleştirilebilecek stok pozisyonu sayısı, maksimum stok pozisyonu sayılarına eşit ise bir sonraki adıma geçilir.

Adım 11: Eğer tüm kısıtlar sağlanmışsa, her bir değişken için alt sınır değerleri elde edilir. Elde edilen değerler, PSO algoritmasında parçacıkların başlangıç değerlerinin belirlenmesinde minimum başlangıç değerleri olarak kullanılır.

Yukarıdaki adımları açıklanmakta olan önerilen atama algoritması, PSO algoritmasının ilk aşaması olan parçacıkların çözüm uzayına rassal dağılımında kullanılır. Parçacıklar uygun olmayan ve geniş çözüm uzayına atanmak yerine daraltılmış çözüm uzayına rassal olarak dağıtılır. Parçacıkların daha dar çözüm uzayına rassal olarak dağılmaları ile ilk iterasyonun sonunda sürünün en iyisi ve parçacıkların bireysel en iyi değerleri belirlenmektedir. Parçacıkların daha dar çözüm uzayına rassal olarak dağılmaları ile parçacıklar uygun olmayan sonuçlarda fazla zaman kaybetmezler ve daha hızlı ve daha iyi sonuçlara ulaşırlar.

4. BULGULAR

Bu bölümde, öncelikle değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemine ilişkin, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri için ve standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış bilgisayar programları sunulacaktır. Daha sonra, her iki program değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin 10 farklı örnek problem üzerinde çalıştırılacaktır. 10 örnek problemin her iki yöntem için hazırlanan programlar ile çözümünden elde edilen sonuçlar ayrı ayrı tablolaştırılacaktır. Ardından, her iki çözüme ait sonuç tabloları tek bir tablo haline getirilerek sonuçlar karşılaştırılacaktır. Karşılaştırmalar maliyet, isabet oranı, süre ve iterasyon sayısı kriterlerine göre yapılacaktır. Karşılaştırma sonucunda söz konusu kriterler dikkate alındığında, hangi yöntem ile yapılan çözümlerde daha iyi sonuçlar elde edildiği belirlenecektir. Son olarak ise kriterler dikkate alındığında en iyi sonucu veren yöntemlerin 10 farklı problem üzerinde depo raf sistemi konfigürasyonu (raf sayısı, raf uzunluğu, raf yüksekliği) ve depo yerleşim boyutlarına (depo eni, depo boyu, depo yüksekliği) ilişkin verdiği sonuçlar sunulacaktır.

4.1. DEPO YERLEŞİM DÜZENLEMESİ PROBLEMİ İÇİN UYGULAMALAR

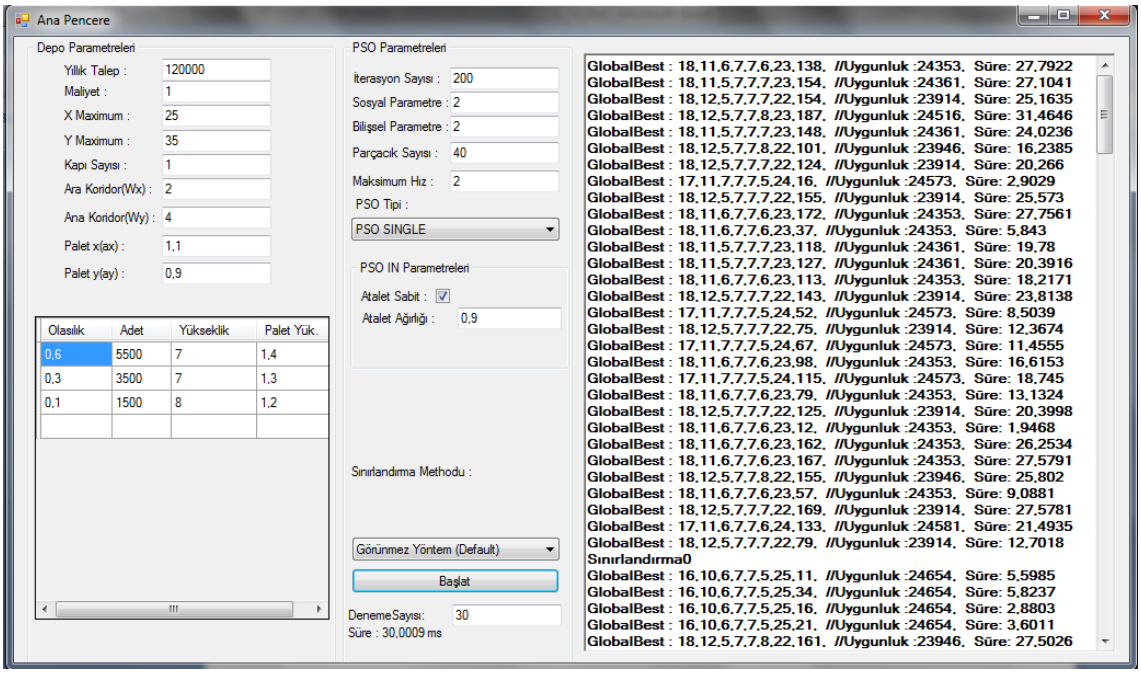
Bu bölümde, 3. Bölüm'de detaylı şekilde açıklanmış olan değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi probleminin çözümünde kullanılan atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ve standart PSO algoritmasının sınırlandırma yöntemleri için ayrı ayrı bilgisayar programları yazılmıştır. Daha sonra, her iki bilgisayar programı için gerekli olan PSO algoritmasının parametreleri ve depo yerleşim düzenlemesinin verileri açıklanmıştır. Son olarak, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin 10 farklı örnek problemin verileri gösterilmiştir.

Depo yerleşim düzenlemesi için iki farklı program hazırlanmıştır. Her iki program, Microsoft Visual Studio 2010 Ultimate platformunda Visual C# dili ile yazılmıştır. Yazılan bilgisayar programının kaynak kodları EK 1'de sunulmuştur. İlk program,

standart PSO algoritması ve PSO algoritması için tüm sınırlandırma yöntemlerini içerirken; diğer program ilk programa ek olarak PSO algoritması için önerilen parçacıkların başlangıç minimum değerleri için atama algoritmasını içermektedir.

Her iki programın ara yüzü birbirinin aynısıdır. Aynı ara yüze sahip olduğu için sadece bir programın ara yüzü gösterilmiştir. Şekil 4.1'de programın örnek bir ekran görüntüsü yer almaktadır. Hazırlanan program, ilk olarak depo yerleşim düzenlemesinde kullanılacak girdileri ve PSO parametrelerini okumaktadır. Ardından programa sürünün yineleme ve koşum sayıları girilmektedir. Son olarak, her bir sınırlandırma yöntemi için yöntem seçimi girdisi ve ürün sayısına göre ürünlerin parametreleri programa ayrı ayrı girilmektedir. Model çalıştırıldığında çıktı olarak sürünün en iyisinin konum vektörleri, çözüm süresi ve en iyi uygunluk bilgileri verilmektedir.

Sonraki bölümde, her iki program için gerekli olan PSO algoritması, depo yerleşim düzenlemesi parametreleri ve farklı depo uygulama verileri verilecektir. Bu veriler programın girdi verilerini oluşturmaktadır.



Şekil 4.1: Depo yerleşim düzenlemesi için geliştirilen programın örnek ekran görüntüsü.

4.1.1. PSO Algoritmasının Parametreleri ve Depo Yerleşim Düzenlemesinin Verileri

Bu bölümde, depo yerleşim düzenlemesi probleminin çözümünde programın girdi verisini oluşturan PSO algoritmasının ve depo yerleşim düzenlemesinin parametreleri verilecektir. Bu parametreler, Şekil 4.1'de örnek program görüntüsünde de görüldüğü gibi, Microsoft Visual Studio 2010 Ultimate platformunda Visual C# dili ile yazılmış her iki program için de ortak parametrelerdir.

Her iki program için de ortak olan parametreler, PSO algoritması parametreleri ve depo yerleşim düzenlemesi verileri olmak üzere iki gruba ayrılmıştır. Her iki programda kullanılan PSO algoritması parametreleri Tablo 4.1'de verilmiştir. Depo yerleşim düzenlemesi verileri ise, Tablo 4.2'de gösterilmiştir. Bu veriler, Önüt ve diğ. (2008) tarafından belirlenmiştir.

Tablo 4.1: PSO algoritması parametreleri.

Parçacık Sayısı	40
İterasyon Sayısı	200
Sosyal Öğrenme Parametresi	2
Bilişsel Öğrenme Parametresi	2
Maksimum Hız Vektörü Değeri	2
Atalet Ağırlığı	0,9

Tablo 4.2: Depo yerleşim düzenlemesi verileri.

Yıllık depodan çıkan palet sayısı	120,000 palet
Elleçleme maliyeti	$1.13 * 10^{-3}$ $/m
Deponun malzeme girişini/çıkışını sağlayan kapının sayısı	1 kapı
Ürün grupları sayısı	3 grup
Ara koridor genişliği	2 metre
Ana koridor genişliği	4 metre
x-ekseni boyunca palet için ayrılan mesafe	1,1 metre
y-ekseni boyunca palet için ayrılan mesafe	0,9 metre

4.1.2. Değişken Yüksekliğe Sahip Paletler İçin Depo Yerleşim Düzenlemesine İlişkin Örnek Problemler

Bir önceki bölümde verilen Tablo 4.1'de ve Tablo 4.2'de yer alan parametreler her iki programın da girdisini oluşturmaktadır. Ancak, her iki tabloda yer alan parametreler ve veriler dışında; örnek problemlere ait depo verilerine de ihtiyaç duyulmaktadır. Örnek problemlerin çözümünde, Tablo 4.3'te gösterilen değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin 10 farklı örnek problemin depo alanı sınırlarına ve malzeme ile ilgili verilere ihtiyaç duyulmaktadır.

Tablo 4.3: Farklı depo örneklerinin verileri.

	Ör. 1	Ör. 2	Ör. 3	Ör. 4	Ör. 5	Ör. 6	Ör. 7	Ör. 8	Ör. 9	Ör. 10
L_{Xmax}	42	31	31	42	52	52	63	63	63	52
L_{Ymax}	36	23	23	27	36	32	32	32	32	32
L_{Zmax}	14	14	14	11	11	11	9	8	12	10
P_A	0,6	0,7	0,5	0,5	0,7	0,6	0,6	0,7	0,5	0,6
P_B	0,3	0,2	0,3	0,3	0,2	0,3	0,3	0,2	0,3	0,3
P_C	0,1	0,1	0,2	0,2	0,1	0,1	0,1	0,1	0,2	0,1
$N_A * 1000$	3	3,5	3,5	4	5,5	5	6	6	6,5	5,5
$N_B * 1000$	2	2,5	2,5	3	4,5	4	4	4,5	5	3,5
$N_C * 1000$	1	0,9	1,5	2	1,9	2	2	2,5	3	1,5
a_{Za}	1	1,1	1,1	1,1	1,3	1,1	1	1,2	1,4	1,4
a_{Zb}	1	1	1,3	1,2	1,4	1	1,1	1	1,3	1,3
a_{Zc}	1,4	1,4	1,4	1,4	1,2	1,1	1,2	1	1,2	1,2
n_{Za-max}	14	12	12	10	8	10	9	6	8	7
n_{Zb-max}	14	14	10	9	7	11	8	8	9	7
n_{Zc-max}	10	10	10	8	9	10	7	8	10	8

Tablo 4.3'de gösterilen veriler, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesinde gerek duyulan tüm verileri göstermektedir. Tablo 4.3'ün ilk üç satırı depo yerleşim düzenlemesi yapılacak alanın boyutlarını göstermekte ve deponun boyut kısıtlarının oluşturulmasında kullanılmaktadır. İkinci üç satırda, stoklanan 3 farklı ürün grubunun siparişlerde bulunma olasılıkları gösterilmektedir. Üçüncü üç satır, her bir örnek depo yerleşim probleminde malzeme grupları için ayrılması gereken stok

pozisyonu sayısını göstermekte ve malzeme gruplarının ayrılması gereken stok pozisyonu kısıtları için kullanılmaktadır. Dördüncü üç satırlık kısımda ise, her bir malzeme grubunun palet yükseklikleri gösterilmektedir. Palet yüksekliği değerlerine göre son üç satırda gösterilen her bir malzeme grubu için z-ekseni boyunca yerleşebilecek maksimum raf sayısı hesaplanmaktadır.

4.2. DEPO YERLEŞİM DÜZENLEMESİ MODELİ PROBLEMİNİN ÇÖZÜMÜ

Bu bölümde, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin farklı örnek problem bölümünde açıklanmış olan 10 farklı depo yerleşim düzenlemesi problemlerinin iki farklı yöntem için hazırlanmış bilgisayar programları ile çözümleri yapılmıştır. Değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin farklı örnek problemlerin çözümündeki ilk çalıştırılan program, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ile çözümü için hazırlanmış programdır. İkinci çalıştırılan program ise, standart PSO algoritmasının sınırlandırma yöntemleri ile çözümü için hazırlanmış programdır. Tüm örnek problemlerin her iki programda çalıştırılarak çözülmesi ile elde edilen sonuçlar ayrı ayrı tablolaştırılmıştır. Ardından, her iki çözüme ait sonuç tabloları özet tablo halinde sunularak sonuçlar maliyet, isabet oranı, süre ve iterasyon sayısı kriterlerine göre karşılaştırılmıştır. Karşılaştırma sonucunda söz konusu kriterler dikkate alındığında, en iyi sonucu veren yöntemlerin 10 farklı problem üzerinde depo raf sistemi konfigürasyonu (raf sayısı, raf uzunluğu, raf yüksekliği) ve depo yerleşim boyutlarına (depo eni, depo boyu) ilişkin verdiği sonuçlar ortaya konmuştur.

4.2.1. Atama Algoritmalı Standart PSO Algoritmasının Sınırlandırma Yöntemleri ile Çözümü

Bu bölümde, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ile örnek problemlerin çözümleri yapılacaktır. Örnek problemlerin çözümlerinin yapılabilmesi için öncelikle parçacıkların çözüm uzayına rassal dağıtımı sırasında kullanılan her bir boyutun minimum başlangıç değerleri belirlenmelidir. Parçacıkların başlangıç minimum değerleri için atama algoritması tüm örnek depo yerleşim düzenlemesi problemlerine uygulanmış ve elde edilen sonuçlar Tablo 4.4'te gösterilmiştir.

Tablo 4.4: Parçacıkların minimum başlangıç değerleri.

	n_X	n_{Ya}	n_{Yb}	n_{Yc}	n_{Za}	n_{Zb}	n_{Zc}
Örnek 1	20	9	8	8	9	7	4
Örnek 2	15	10	8	5	12	11	6
Örnek 3	15	10	9	6	12	10	9
Örnek 4	20	10	9	11	10	9	5
Örnek 5	25	14	13	9	8	7	5
Örnek 6	25	10	10	9	10	8	5
Örnek 7	30	12	9	9	9	8	4
Örnek 8	30	17	10	7	6	8	6
Örnek 9	27	16	11	8	8	9	7
Örnek 10	25	16	10	8	7	7	4

Tablo 4.4'teki her bir boyut için belirlenmiş başlangıç değerlerinin kullanılması ile parçacıklar daha sınırlı çözüm uzayına rassal olarak dağılacaktır. Yeni başlangıç değerleri ile parçacıkların her bir ürün grubu için y-ekseni boyunca minimum raf sayıları, x-ekseni boyunca minimum raf sayısı ve her bir ürün grubu için z-ekseni boyunca minimum raf sayıları revize edilmektedir. Minimum değerler belirlendikten sonra her bir ürün grubu için y-ekseni boyunca maksimum raf sayıları, x-ekseni boyunca maksimum raf sayısı ve her bir ürün grubu için z-ekseni boyunca maksimum raf sayıları da depo yerleşim düzenlemesi yapılacak alanın boyutlarına göre belirlenmektedir. Her bir ürün grubu için z-ekseni boyunca maksimum raf sayıları sadece deponun yerleşim düzenlemesi için yapılacak alanın yüksekliğine bağlı değildir. Depo yerleşim düzenlemesinde kullanılacak maksimum raf sayıları, depo yerleşim düzenlemesine karar verenler tarafından belirlenen elleçleme ekipmanının teknik özelliklerine göre de belirlenebilmektedir. Parçacıkların çözüm uzayına rassal dağılımındaki kullanılacak tüm üst sınır değerler, Tablo 4.3'ün son üç satırında gösterilmektedir.

Parçacıkların çözüm uzayına rassal olarak dağılımında kullanılacak maksimum ve minimum değerler belirlendikten sonra atama algoritmalı PSO algoritması sekiz sınırlandırma yöntemi için de programlar ayrı ayrı 30 kez çalıştırılır. Programların çalıştırılması ile elde edilen sonuçlar Tablo 4.5 ve Tablo 4.6'da gösterilmektedir.

Tablo 4.5: Atama algoritmalı PSO algoritmasının maliyet sonuçları.

		Maliyet							
		Görünmez	Emme	Yansıtma	Sönümleme	Görünmez Sönümleme	Görünmez yansıtma	Görünür yansıtma	Görünür sönümleme
1. Örnek	En İyi Maliyet	15165	15165	15165	15165	15165	15165	15165	15165
	Ortalama Maliyet	15179	15179	15198	15185	15180	15184	15171	15199
	Maliyetin Standart Sapması	25	30	63	42	31	38	22	52
	İsabet Oranı	%57	%77	%70	%73	%60	%67	%87	%63
2. Örnek	En İyi Maliyet	16567	16567	16567	16567	16567	16567	16567	16567
	Ortalama Maliyet	16637	16624	16672	16660	16651	16635	16618	16627
	Maliyetin Standart Sapması	51	69	129	135	75	71	71	70
	İsabet Oranı	%3	%13	%13	%27	%17	%23	%27	%10
3. Örnek	En İyi Maliyet	18650	18650	18650	18650	18650	18650	18650	18650
	Ortalama Maliyet	18653	18650	18653	18653	18661	18650	18650	18650
	Maliyetin Standart Sapması	14	0	9	8	23	0	0	0
	İsabet Oranı	%93	%100	%87	%90	%77	%100	%100	%100
4. Örnek	En İyi Maliyet	20236	20735	20236	20236	20735	20236	20236	20236
	Ortalama Maliyet	20652	20940	20839	20837	20849	20663	20662	20675
	Maliyetin Standart Sapması	189	55	140	215	114	199	293	279
	İsabet Oranı	%17	%0	%3	%10	%0	%17	%30	%27
5. Örnek	En İyi Maliyet	22905	22905	22905	22905	22905	22905	22905	22905
	Ortalama Maliyet	23013	22995	22985	22985	23038	22988	22979	22966
	Maliyetin Standart Sapması	60	64	67	67	83	77	70	68
	İsabet Oranı	%23	%30	%37	%37	%17	%33	%47	%53
6. Örnek	En İyi Maliyet	20253	20253	20253	20253	20253	20253	20253	20253
	Ortalama Maliyet	20331	20311	20319	20305	20343	20308	20303	20306
	Maliyetin Standart Sapması	40	62	43	42	75	48	48	44
	İsabet Oranı	%20	%30	%13	%27	%23	%43	%47	%33
7. Örnek	En İyi Maliyet	22555	22555	22555	22555	22555	22555	22555	22555
	Ortalama Maliyet	23128	23002	23208	23133	23213	23000	22881	22972
	Maliyetin Standart Sapması	230	751	767	717	135	323	332	673
	İsabet Oranı	%13	%63	%40	%43	%30	%33	%50	%60
8. Örnek	En İyi Maliyet	25980	25980	25980	25980	25980	25980	25980	25980
	Ortalama Maliyet	26238	26408	26546	26570	26355	26230	26422	26330
	Maliyetin Standart Sapması	248	300	187	188	218	238	287	284
	İsabet Oranı	%47	%30	%7	%7	%23	%47	%27	%37
9.Örnek	En İyi Maliyet	27011	27011	27011	27011	27011	27011	27011	27011
	Ortalama Maliyet	27032	27063	27063	27080	27016	27013	27020	27063
	Maliyetin Standart Sapması	95	159	159	180	7	5	33	159
	İsabet Oranı	%77	%90	%90	%87	%70	%90	%93	%90
10. Örnek	En İyi Maliyet	23914	23914	23914	23914	23946	23914	23914	23914
	Ortalama Maliyet	24196	24428	24502	24389	24535	24295	24248	24426
	Maliyetin Standart Sapması	234	337	260	324	206	349	329	322
	İsabet Oranı	%30	%17	%13	%13	%0	%37	%43	%20

Tablo 4.5, 10 farklı depo yerleşim düzenlemesinin her bir sınırlandırma yönteminden elde edilen maliyetlerin ve isabet oranlarının bir arada gösterildiği tablodur. Her bir sınırlandırma yöntemi ile her bir örnek problemin çözümünden elde edilen maliyetlerin

karşılaştırılmasında değerlendirme kriterleri olarak en iyi maliyet, ortalama maliyet, maliyetin standart sapması ve isabet oranı dikkate alınmıştır.

Tablo 4.6: Atama algoritmalı PSO algoritmasının bilgisayar süresi sonuçları.

		Bilgisayar Süresi							
		Görünmez	Emme	Yansıtma	Sönümleme	Görünmez Sönümleme	Görünmez yansıtma	Görünür yansıtma	Görünür sönümleme
1. Örnek	En İyi Süre	2	8	12	8	6	9	2	6
	Ortalama Süre	19	22	21	22	23	23	19	19
	Sürenin Standart Sapması	7	9	8	13	8	7	6	9
	İterasyon Sayısı	13	48	74	47	31	55	16	34
2. Örnek	En İyi Süre	15	1	13	5	5	1	10	5
	Ortalama Süre	16	11	14	12	21	16	16	15
	Sürenin Standart Sapması	6	6	8	7	11	9	8	8
	İterasyon Sayısı	56	7	79	30	28	6	55	30
3. Örnek	En İyi Süre	1	1	1	2	1	1	1	1
	Ortalama Süre	10	5	8	5	18	7	6	5
	Sürenin Standart Sapması	8	7	7	4	11	6	5	5
	İterasyon Sayısı	2	2	3	14	5	2	1	5
4. Örnek	En İyi Süre	1	6	3	9	24	10	6	1
	Ortalama Süre	18	2	11	7	24	16	12	10
	Sürenin Standart Sapması	8	2	10	8	10	7	9	9
	İterasyon Sayısı	11	35	20	55	138	57	38	3
5. Örnek	En İyi Süre	1	5	1	2	11	6	3	1
	Ortalama Süre	16	10	12	12	18	13	13	12
	Sürenin Standart Sapması	9	6	8	9	7	7	7	9
	İterasyon Sayısı	7	34	7	14	62	35	18	2
6. Örnek	En İyi Süre	10	12	19	8	16	9	7	5
	Ortalama Süre	22	15	18	15	23	20	21	18
	Sürenin Standart Sapması	7	6	9	7	7	6	9	6
	İterasyon Sayısı	61	70	111	50	94	55	40	34
7. Örnek	En İyi Süre	13	4	9	5	25	11	5	7
	Ortalama Süre	19	11	18	12	23	16	15	15
	Sürenin Standart Sapması	7	5	7	7	7	7	7	7
	İterasyon Sayısı	84	24	51	32	149	66	32	46
8. Örnek	En İyi Süre	1	1	1	7	1	1	1	1
	Ortalama Süre	14	8	7	6	15	9	8	9
	Sürenin Standart Sapması	10	7	6	6	11	7	8	8
	İterasyon Sayısı	4	2	7	41	4	2	7	10
9.Örnek	En İyi Süre	1	1	1	1	1	1	1	1
	Ortalama Süre	11	9	12	10	16	12	11	11
	Sürenin Standart Sapması	8	7	7	8	12	9	9	9
	İterasyon Sayısı	5	4	5	10	4	9	9	4
10. Örnek	En İyi Süre	14	9	20	9	24	18	7	7
	Ortalama Süre	17	10	11	11	19	21	14	10
	Sürenin Standart Sapması	8	10	9	8	11	10	9	9
	İterasyon Sayısı	91	53	115	55	143	102	45	44

Tablo 4.6'da sunulan değerler, her bir sınırlandırma yöntemi ile örneklerin çözümünde geçen bilgisayar çözüm sürelerini göstermektedir. Tablo 4.6'dan da görüleceği gibi, değerlendirme kriterleri en iyi zaman, ortalama süre, sürenin standart sapması ve iterasyon sayısıdır. Ancak Tablo 4.6'da gösterilen en iyi süre ve iterasyon sayısı, Tablo 4.5'te gösterilen en iyi maliyetin süresi ve iterasyon sayısıdır.

Tablo 4.5 ve Tablo 4.6, örnek depo yerleşim düzenlemesi problemlerinin her bir sınırlandırma yöntemi ile yapılan çözümlerinde elde edilen sonuçların kriterlere göre bir arada değerlendirildiği tablolardır. Sonuçların daha kolay irdelenebilmesi için her iki tablonun en iyi sonuçlarından oluşan özet tablo olan Tablo 4.7 kullanılacaktır.

Tablo 4.7'den de görüleceği gibi tüm örnek depo yerleşim düzenlemesi problemlerinin çözümlerinden elde edilen en iyi maliyetler 15,165 $ ile 27,011 $ aralığındadır. Bunun yanı sıra, 30 deneme sonunda elde edilen sonuçların tümünün ortalamasının alınmasıyla elde edilen ortalama maliyet, 15,171 $ ile 27,020 $ aralığındadır. Tüm örnek problemlerin çözümlerindeki en düşük maliyetin standart sapması 0 $'dır ve bu değer 3. örnek problemin çözümünden elde edilmiştir. En büyük maliyet standart sapması değeriyse 751 $'dır ve bu değer 7. örnek problemin çözümünden gelmektedir.

Elde edilen sonuçlar, isabet oranı kriterine göre değerlendirildiğinde, tüm çözümlerin isabet oranları % 27 ile %100 aralığında değişmektedir. Dört farklı sınırlandırma yöntemi, 3. örneğin çözümünde %100'lük isabet oranına sahiptir. En düşük isabet oranına sahip çözüm 2. örneğin çözümündedir ve isabet oranı %27'dir.

Tablo 4.7'den çıkarılabilecek bir diğer sonuç, tüm örnekler arasından en iyi çözümün süresinin 1 ms ve en yüksek çözüm süresinin 10 ms olmasıdır. En düşük ortalama çözüm süresi, 6 ms çözüm süresine sahip olan 3. örnek problemin çözüm süresidir. En yüksek ortalama çözüm süresi ise 21 ms çözüm süresi ile 6. örnek probleme aittir. Tüm örnek problemlerin çözümlerinin standart sapma süreleri, 5 ms ile 9 ms arasında değişmektedir.

Örnek problemlerin çözümlerinin iterasyon sayılarına bakıldığında iterasyon sayılarının 1 ile 55 arasında değiştiği görülmektedir. Emme sınırlandırma yöntemi, görünmez yansıtma sınırlandırma yöntemi, görünür yansıtma sınırlandırma yöntemi ve görünür sönümleme sınırlandırma yöntemi ile 3. örnek problemin çözümünden elde edilen

sonuçlara bakıldığında söz konusu sınırlandırma yöntemlerinin en iyi maliyetlere ve en iyi isabet oranlarına sahip olduğu anlaşılmaktadır. Örnek problemlerin çözümleri incelendiğinde tüm çözümlerde büyük çoğunlukla en iyi sonucu veren yöntem görünür yansıtma sınırlandırma yöntemidir. Görünür yansıtma sınırlandırma yöntemi, on örnek problemin yedisinde en iyi sonucu vermiştir. İkişer kez en iyi sonuç veren yöntemler ise, emme sınırlandırma yöntemi, görünür emme sınırlandırma yöntemi ve görünmez yansıtma sınırlandırma yöntemidir. Görünmez sınırlandırma yöntemi ve sönümleme sınırlandırma yöntemi bir defa en iyi sonucu elde etmiştir. Bazı örnek problemlerin çözümünde birden fazla sınırlandırma yöntemi en iyi sonucu verdiği için örnek problem sayısından daha fazla en iyi sonucu veren yöntem elde edilmiştir.

Tablo 4.7: Atama algoritmalı PSO algoritmasının özet tablosu.

	1. Örnek	2. Örnek	3. Örnek	4. Örnek	5. Örnek	6. Örnek	7. Örnek	8. Örnek	9. Örnek	10. Örnek
En İyi Maliyet	15165	16567	18650	20236	22905	20253	22555	25980	27011	23914
Ortalama Maliyet	15171	16618	18650	20662	22966	20303	23002	26230	27020	24248
Maliyetin Standart Sapması	22	71	0	293	68	48	751	238	33	329
İsabet Oranı	%87	%27	%100	%30	%53	%47	%63	%47	%93	%43
En İyi Süre	2	10	1	6	1	7	4	1	1	7
Ortalama Süre	19	16	6	12	12	21	11	9	11	14
Sürenin Standart Sapması	6	8	5	9	9	9	5	7	9	9
İterasyon Sayısı	16	55	1	38	2	40	24	2	9	45
Sınırlandırma yöntemleri	Görünür yansıtma	Sönümleme /Görünür yansıtma	Emme / Görünmez yansıtma / Görünür yansıtma /Görünür sönümleme	Görünür yansıtma	Görünür sönümleme	Görünür yansıtma	Emme	Görünmez/ Görünmez yansıtma	Görünür yansıtma	Görünür yansıtma

4.2.2. Standart PSO Algoritmasının Sınırlandırma Yöntemleri ile Çözümü

Bu bölümde, standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış bilgisayar programı, değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin 10 farklı örnek problem üzerinde ayrı ayrı 30 kez çalıştırılmıştır. Söz konusu program bir önceki bölümde anlatılmış olan atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış programın çalıştığı aynı bilgisayarda ve aynı koşullarda çalıştırılmıştır. Elde edilen sonuçlar Tablo 4.8 ve Tablo 4.9'da verilmektedir.

Tablo 4.8'de, 10 farklı depo yerleşim düzenlemesinin her bir sınırlandırma yöntemi ile çözümünden elde edilen maliyetler ve isabet oranları bir arada gösterilmektedir. Bir önceki bölümde yer alan Tablo 4.5'te olduğu gibi, her bir sınırlandırma yöntemi kullanılarak örnek problemlerin çözümünden elde edilen maliyetler en iyi maliyet, ortalama maliyet, maliyetin standart sapması ve isabet oranı kriterleri dikkate alınarak değerlendirilmiştir.

Tablo 4.9, örnek depo yerleşim düzenlemesi problemlerinin her bir sınırlandırma yöntemi kullanılarak çözülmesi sırasında geçen bilgisayar çözüm süresi sonuçlarının bir arada gösterildiği tablodur. Bir önceki bölümde verilen Tablo 4.6'da olduğu gibi, Tablo 4.9'da da değerlendirme kriterleri olarak en iyi zaman, ortalama süre, sürenin standart sapması ve iterasyon sayısı belirlenmiştir.

Tablo 4.8 ve Tablo 4.9, örnek depo yerleşim düzenlemesi problemlerinin her bir sınırlandırma yöntemi ile yapılan çözümlerinde elde edilen sonuçların kriterlere göre bir arada değerlendirildiği tablolardır. Sonuçların daha kolay analiz edilebilmesi için her iki tablonun en iyi sonuçlarından oluşan özet tablo olan Tablo 4.10 oluşturulmuştur.

Tablo 4.8: Standart PSO algoritmasının maliyet sonuçları.

		Maliyet							
		Görünmez	Emme	Yansıtma	Sönümleme	Görünmez Sönümleme	Görünmez yansıtma	Görünür yansıtma	Görünür sönümleme
1. Örnek	En İyi Maliyet	15165	15165	15165	15165	15165	15165	15165	15165
	Ortalama Maliyet	15270	15278	15292	15228	15287	15242	15244	15289
	Maliyetin Standart Sapması	118	123	149	83	121	104	108	115
	İsabet Oranı	%7	%20	%13	%27	%10	%7	%13	%13
2. Örnek	En İyi Maliyet	16752	16670	16567	16884	16708	16912	16722	16830
	Ortalama Maliyet	17011	17125	17077	17160	17069	17160	17108	17061
	Maliyetin Standart Sapması	196	189	202	134	168	179	195	162
	İsabet Oranı	%0	%0	%3	%0	%0	%0	%0	%0
3. Örnek	En İyi Maliyet	18678	18677	19043	18650	19043	19092	18677	18677
	Ortalama Maliyet	18678	18677	19043	18650	19043	19092	18677	18677
	Maliyetin Standart Sapması	0	0	0	0	0	0	10	0
	İsabet Oranı	%0	%0	%0	%3	%0	%0	%0	%0
4. Örnek	En İyi Maliyet	20836	20757	20887	20757	20928	20763	20730	20236
	Ortalama Maliyet	20964	21344	21306	21174	20981	21246	21398	21078
	Maliyetin Standart Sapması	134	512	315	387	53	390	393	658
	İsabet Oranı	%0	%0	%0	%0	%0	%0	%0	%3
5. Örnek	En İyi Maliyet	23759	23084	23198	23043	23011	23076	23230	23076
	Ortalama Maliyet	24291	23998	23932	23887	23823	24076	23968	23935
	Maliyetin Standart Sapması	660	528	500	469	441	476	449	477
	İsabet Oranı	%0	%0	%0	%0	%0	%0	%0	%0
6. Örnek	En İyi Maliyet	20429	20283	20486	20283	20253	20353	20253	20353
	Ortalama Maliyet	21323	21542	21615	21271	21437	21497	21573	21436
	Maliyetin Standart Sapması	735	863	1084	743	681	824	933	800
	İsabet Oranı	%0	%0	%0	%0	%3	%0	%3	%0
7. Örnek	En İyi Maliyet	23125	22555	23255	23222	23442	23222	23230	23214
	Ortalama Maliyet	24948	24626	24468	24850	24905	24515	24617	24732
	Maliyetin Standart Sapması	870	1110	1020	1095	882	979	862	990
	İsabet Oranı	%0	%7	%10	%0	%0	%0	%0	%0
8. Örnek	En İyi Maliyet	26721	26772	26721	26721	27266	25980	26705	26802
	Ortalama Maliyet	26721	26772	26721	26721	27266	26351	26705	26802
	Maliyetin Standart Sapması	0	0	0	0	0	523	0	0
	İsabet Oranı	%0	%0	%0	%0	%0	%3	%0	%0
9.Örnek	En İyi Maliyet	27735	27141	27792	27092	28435	27890	27532	28337
	Ortalama Maliyet	28106	27772	28396	28519	28905	28354	28392	28665
	Maliyetin Standart Sapması	322	891	499	668	570	360	426	303
	İsabet Oranı	%0	%0	%0	%0	%0	%0	%0	%0
10. Örnek	En İyi Maliyet	25276	24418	24671	24394	25647	24638	24386	24394
	Ortalama Maliyet	25462	24903	24847	24665	25761	25143	24557	24734
	Maliyetin Standart Sapması	262	450	174	250	161	713	204	228
	İsabet Oranı	%0	%0	%0	%0	%0	%0	%0	%0

Tablo 4.9: Standart PSO algoritmasının bilgisayar süresi sonuçları.

		Bilgisayar Süresi							
		Görünmez	Emme	Yansıtma	Sönümleme	Görünmez Sönümleme	Görünmez yansıtma	Görünür yansıtma	Görünür sönümleme
1. Örnek	En İyi Süre	24	18	18	7	22	11	20	22
	Ortalama Süre	21	24	20	23	25	22	23	22
	Sürenin Standart Sapması	5	8	7	7	8	7	6	5
	İterasyon Sayısı	155	180	101	38	125	52	112	125
2. Örnek	En İyi Süre	45	15	47	60	41	54	57	22
	Ortalama Süre	35	34	38	35	40	36	32	41
	Sürenin Standart Sapması	7	10	10	18	12	14	17	11
	İterasyon Sayısı	171	41	173	170	146	186	163	58
3. Örnek	En İyi Süre	15	30	20	87	80	5	50	55
	Ortalama Süre	15	30	20	87	80	5	50	57
	Sürenin Standart Sapması	0	0	0	0	0	0	0	2
	İterasyon Sayısı	25	55	30	125	90	10	60	60
4. Örnek	En İyi Süre	32	3	42	23	15	32	40	19
	Ortalama Süre	24	24	27	24	23	27	27	22
	Sürenin Standart Sapması	13	12	13	13	7	7	12	7
	İterasyon Sayısı	173	16	189	102	65	175	141	108
5. Örnek	En İyi Süre	21	32	25	18	21	15	12	13
	Ortalama Süre	19	24	17	17	24	28	23	23
	Sürenin Standart Sapması	6	9	8	7	5	6	10	9
	İterasyon Sayısı	140	177	139	92	121	84	67	75
6. Örnek	En İyi Süre	32	12	25	12	24	26	36	11
	Ortalama Süre	23	23	20	25	23	22	26	21
	Sürenin Standart Sapması	8	9	8	7	10	10	10	9
	İterasyon Sayısı	198	69	115	67	132	139	199	64
7. Örnek	En İyi Süre	28	27	24	7	11	37	20	66
	Ortalama Süre	22	22	24	24	28	24	23	28
	Sürenin Standart Sapması	8	11	9	10	8	10	10	11
	İterasyon Sayısı	178	152	134	40	60	198	109	29
8. Örnek	En İyi Süre	33	37	37	37	37	38	13	25
	Ortalama Süre	33	37	37	37	37	29	13	25
	Sürenin Standart Sapması	0	0	0	0	0	12	0	0
	İterasyon Sayısı	110	197	110	110	110	188	75	135
9.Örnek	En İyi Süre	33	1	15	20	11	30	22	33
	Ortalama Süre	15	18	25	15	23	30	19	20
	Sürenin Standart Sapması	11	23	9	11	8	2	11	11
	İterasyon Sayısı	198	7	84	111	61	167	124	180
10. Örnek	En İyi Süre	9	31	14	26	17	19	34	15
	Ortalama Süre	12	23	23	24	21	18	28	26
	Sürenin Standart Sapması	4	9	8	6	5	2	7	7
	İterasyon Sayısı	55	175	79	144	98	101	179	85

Tablo 4.10: Standart PSO algoritmasının özet tablosu.

	1. Örnek	2. Örnek	3. Örnek	4. Örnek	5. Örnek	6. Örnek	7. Örnek	8. Örnek	9. Örnek	10. Örnek
En İyi Maliyet	15165	16567	18650	20236	23011	20253	22555	25980	27092	24386
Ortalama Maliyet	15228	17077	18650	21078	23823	21437	24626	26351	28519	24557
Maliyetin Standart Sapması	83	202	0	658	441	681	1110	523	668	204
İsabet Oranı	%27	%3	%3	%3	%3	%3	%7	%3	%3	%3
En İyi Süre	7	47	87	19	21	24	27	38	20	34
Ortalama Süre	23	38	87	22	24	23	22	29	15	28
Sürenin Standart Sapması	7	10	0	7	5	10	11	12	11	7
İterasyon Sayısı	38	173	125	108	121	132	152	188	111	179
Sınırlandırma yöntemleri	Sönümleme	Yansıtma	Sönümleme	Görünür sönümleme	Görünmez sönümleme	Görünmez sönümleme	Yansıtma	Görünmez yansıtma	Sönümleme	Görünür yansıtma

Tablo 4.10'da da görüldüğü gibi örnek depo problemlerinin çözümlerinden elde edilen en iyi maliyetler 15,165 $ ile 27,091 $ aralığındadır. Bunun yanı sıra, denemeler sonunda elde edilen sonuçların tümünün ortalamasının alınmasıyla elde edilen ortalama maliyet 15,228 $ ile 28,519 $ aralığındadır. En düşük maliyetin standart sapması 0 $'dır bu değer 3. örnek problemin çözümünden elde edilmiştir. Örnek problemlerin sonuçları arasında en büyük standart sapma değeri ise 1,110 $'dır ve bu değer 7. örneğin çözümünde bulunmuştur. İsabet oranının dikkate alınması durumunda, tüm çözümlerin isabet oranları %3 ile %27 aralığında değişmektedir.

Tablo 4.10'daki bilgisayar çözüm süreleri incelendiğinde, tüm örnekler arasından en iyi çözüm süresinin 7 ms, en yüksek çözüm süresinin ise 87 ms olduğu görülmektedir. En düşük ortalama çözüm süresi, 15 ms çözüm süresi ile 9. örneğin ortalama çözüm süresidir. En yüksek ortalama çözüm süresi, 87 ms'lik çözüm süresine sahip olan 3. örnek problemin çözüm süresidir. Örnek problemlerin çözümünün standart sapma süreleri 0 ms ile 12 ms arasında değişmektedir. 3. örneğin standart sapmasının 0 olmasının nedeni, 30 deneme içerisinden tek bir çözüm elde edilmiş olması ve diğer denemelerde de herhangi olurlu bir sonuç üretememiş olmasıdır. İterasyon sayılarına bakıldığında, en düşük iterasyon sayısının 38, en yüksek iterasyon sayısının ise 188 olduğu görülmektedir.

Örnek problemlerin çözümlerinde en fazla en iyi maliyeti veren sınırlandırma yöntemi sönümleme sınırlandırma yöntemidir ve bu yöntem ile üç kez en iyi maliyet bulunmuştur. İkişer kez en iyi sonuç veren sınırlandırma yöntemler, yansıtma sınırlandırma yöntemi ve görünmez sönümleme sınırlandırma yöntemidir. Görünmez yansıtma sınırlandırma yöntemi, görünür yansıtma sınırlandırma yöntemi ve görünür sönümleme sınırlandırma yöntemi ise birer kez en iyi sonucu bulan çözüm yöntemleridir.

4.2.3. Atama Algoritmalı Standart PSO Algoritması ile Standart PSO Algoritmasının Karşılaştırılması

Önceki bölümlerde, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemlerinin ve standart PSO algoritmasının sınırlandırma yöntemlerinin sonuçları sadece kendi içerisinde değerlendirilmiş ve özet tablolar olan Tablo 4.7 ve Tablo 4.10

elde edilmişti. Bu bölümde ise, söz konusu yöntemlerin özet tabloları birbirleri ile karşılaştırılmıştır ve karşılaştırma sonuçları Tablo 4.11 üzerinde verilmiştir.

Tablo 4.11'de gibi sekiz kritere göre atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ve standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış özet tablolar olan Tablo 4.7 ve Tablo 4.10 karşılaştırılmıştır. Tablo 4.11'de verilmekte olan yüzde değerlerini açıklamadan önce, söz konusu tabloda yer alan negatif değerlerin anlamını açıklamak gerekmektedir. Bu noktada en iyi maliyet, ortalama maliyet, maliyetin standart sapması, en iyi süre, ortalama süre, sürenin standart sapması ve iterasyon sayısı kriterlerinde birtakım negatif yüzde değerleri görülmektedir. Bu yüzde değerlerinin negatif olması, atama algoritmalı PSO algoritması ile elde edilen sonuçların, standart PSO algoritmasına göre daha iyi sonuç vererek; söz konusu kritelerde elde edilen sonuçlarda iyileştirme yapıldığını göstermektedir. İsabet oranı kriterinde ise negatif bir yüzde değeri bulunmamakta, aksine yüzde değerleri artmaktadır. İsabet oranı kriterinde yer alan yüzde değerlerinin artması atama algoritmalı PSO algoritması ile elde edilen sonuçların, standart PSO algoritmasına göre daha yüksek isabet oranına sahip olduğunu göstermektedir.

Tablo 4.11: Atama algoritmalı PSO algoritması ile standart PSO algoritması karşılaştırma tablosu.

	1. Örnek	2. Örnek	3. Örnek	4. Örnek	5. Örnek	6. Örnek	7. Örnek	8. Örnek	9. Örnek	10. Örnek
En İyi Maliyet	%0,0	%0,0	%0,0	%0,0	%-0,5	%0,0	%0,0	%0,0	%-0,3	%-1,9
Ortalama Maliyet	%-0,4	%-2,7	%0,0	%-2,0	%-3,6	%-5,3	%-6,6	%-0,5	%-5,3	%-1,3
Maliyetin Standart Sapması	%-73,5	%-64,9	%0,0	%-55,5	%-84,6	%-93,0	%-32,3	%-54,5	%-95,1	%61,3
İsabet Oranı	%222,2	%800,0	%3233,3	%900,0	%1666,7	%1466,7	%800,0	%1466,7	%3000,0	%1333,3
En İyi Süre	%-71,4	%-78,7	%-98,9	%-68,4	%-95,2	%-70,8	%-85,2	%-97,4	%-95,0	%-79,4
Ortalama Süre	%-17,4	%-57,9	%-93,1	%-45,5	%-50,0	%-8,7	%-50,0	%-69,0	%-26,7	%-50,0
Sürenin Standart Sapması	%-14,3	%-20,0	%0,0	%28,6	%80,0	%-10,0	%-54,5	%-41,7	%-18,2	%28,6
İterasyon Sayısı	%-57,9	%-68,2	%-99,2	%-64,8	%-98,3	%-69,7	%-84,2	%-98,9	%-91,9	%-74,9

Öncelikle, en iyi maliyet kriteri göz önüne alınarak değerlendirilme yapıldığında 5. örnekte, 8. örnekte ve 9. örnekte %0,3 ile %1,9 arasında iyileştirme gerçekleştirilmiştir. Her iki yöntemde de 5. örnek, 8. örnek ve 9. örnek dışındaki tüm örneklerde en uygun değerler elde edilmiştir. Ancak, ortalama maliyet kriterine göre değerlendirildiğinde tüm örneklerde atama algoritmalı PSO algoritmasının %0,4 ile %6,6 arasında iyileştirme gerçekleştirdiği görülmektedir. Bununla birlikte atama algoritmalı PSO algoritması, 10. örnek dışındaki tüm örneklerde standart sapmalarda büyük iyileştirmeler elde etmiştir. Gerçek anlamda en büyük iyileştirme ise, isabet oranları değerlendirme kriterinde gerçekleşmiştir. En düşük isabet oranı iyileştirmesi %222,2 ve en yüksek isabet oranı iyileştirmesi ise %3233,3 olmuştur. Bu durum, atama algoritması ile parçacıkların problem çözümünde en iyi değeri bulma oranını çok yükseltmiştir.

Ayrıca bilgisayar süresi ve iterasyon sayısı değerlendirme kriterleri dikkate alındığında, her bir örnek problem çözümünün neredeyse tümünde atama algoritmalı PSO algoritması daha iyi sonuçlar vererek öne çıkmaktadır. Önerilen atama algoritması ile çözüm için harcanan sürelerde yaklaşık olarak %70 oranında azalma gerçekleşmiştir. Aynı durum, ortalama süre kriteri göz önüne alındığında da görülmektedir. Bu bağlamda, yaklaşık olarak %10 ile %70 oranında ortalama çözüm sürelerinde azalma meydana gelmiştir. İterasyon sayısı azalmasında da, ortalama süre ile en iyi sürede olduğu gibi gözle görülür bir iyileştirme sağlanmıştır. Bu iyileştirme sayesinde parçacıklar daha az sayıda iterasyonla problemin çözümüne ulaşmaktadır. İterasyon sayısı kriterine göre değerlendirildiğinde iyileştirme oranlarının %58 ile %99 arasında gerçekleştiği görülmektedir.

Özetle, atama algoritmalı PSO algoritması ile standart PSO algoritması yukarıdaki kriterler göz önüne alınarak karşılaştırıldığında; atama algoritmalı PSO algoritmasının daha iyi sonuçlara ulaştığı görülmektedir. Önerilen atama algoritmasıyla, PSO algoritmasındaki parçacıklar daha hızlı şekilde, daha kısa sürede, daha az iterasyon sayısıyla ve daha yüksek isabet oranıyla en iyi sonuca ulaşmaktadır. Bu değerlendirmeler Tablo 4.11'den rahatlıkla çıkarılmaktadır. Tablo 4.11'de yer almamasına rağmen elde edilmiş olan bir başka sonuç ise, standart PSO algoritmalı yöntemin kullanılarak örnek problemlerin çözümünde farklı farklı sınırlandırma yöntemlerinin başarı gösterdiği; ancak baskın bir sınırlandırma yöntemi elde

edilememiş olmasıdır. Diğer taraftan, atama algoritmalı PSO algoritması kullanılarak çözülen örnek problemlerde %70'inde en iyi sonucu veren yöntem "Görünür yansıtma sınırlandırma yöntemi" olmuştur. Bu noktada, "Görünür yansıtma sınırlandırma yöntemi"nin diğer yöntemlere göre baskın olduğu anlaşılmaktadır.

Değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemine ilişkin, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri için ve standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanmış bilgisayar programlarının 10 farklı örnek problem üzerine uygulanması sonucu elde edilen bulgular aşağıdaki şekilde özetlenmektedir.

- Her iki çözüm yöntemi en iyi maliyet kriteri göz önüne alınarak değerlendirilirse; atama algoritmalı standart PSO algoritması sınırlandırma yöntemleri ile standart PSO algoritması sınırlandırma yöntemleri arasında pek bir fark görülmemektedir. Ancak, ortalama maliyet ve maliyetin standart sapması kriterlerine göre değerlendirildiğinde, atama algoritmalı standart PSO algoritması sınırlandırma yöntemleri çok daha iyi sonuçlar vermektedir.
- Atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri, her bir örnek depo yerleşim düzenlemesi probleminde standart PSO algoritmasının sınırlandırma yöntemlerine göre daha yüksek isabet oranlarına sahiptir.
- Her iki çözüm yöntemi tüm süre kriterleri dikkate alınarak incelendiğinde, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri neredeyse tüm süre kriterlerinde iyileştirme sağlamıştır.
- Atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri, standart PSO algoritmasının sınırlandırma yöntemlerine göre daha az iterasyon sayısı ile sonuçlara ulaşmaktadır.
- En son ama en önemli bulguda, standart PSO algoritmasının sınırlandırma yöntemleri ile çözülen örnek problemlerin sonuçlarına bakıldığında baskın bir sınırlandırma yöntemi bulunmadığı görülmektedir. Ancak, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ile çözülen örnek problemlerin sonuçlarına bakıldığında ise, "Görünür yansıtma sınırlandırma yöntemi"nin diğer yöntemler arasında bir hayli öne çıkarak etkili çözüm yöntemi olduğu görülmektedir.

Atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri ile ve standart PSO algoritmasının sınırlandırma yöntemleri ile çözülmüş olan problemlerden elde edilen sonuçlardan en iyi sonucu veren yöntemlerin 10 farklı problem üzerinde depo raf sistemi konfigürasyonu (raf sayısı, raf uzunluğu, raf yüksekliği) ve depo yerleşim boyutlarına (depo eni, depo boyu) ilişkin verdiği sonuçlar Tablo 4.12'de gösterilmiştir.

Tablo 4.12: Örnek problemlere ait sonuçlar.

	n_X	n_{Ya}	n_{Yb}	n_{Yc}	n_{Za}	n_{Zb}	n_{Zc}	**Maliyet ($)**	**Uzunluk (m)**	**Genişlik (m)**
Örnek 1	9	12	8	8	14	14	7	15.165	37,8	25,2
Örnek 2	12	13	8	4	12	14	10	16.567	50,4	22,5
Örnek 3	15	10	9	6	12	10	9	18.650	63	22,5
Örnek 4	17	12	10	8	10	9	8	20.236	71,4	27
Örnek 5	21	17	16	7	8	7	7	22.905	88,2	36
Örnek 6	16	16	12	7	10	11	9	20.253	67,2	31,5
Örnek 7	21	16	12	7	9	8	7	22.555	88,2	31,5
Örnek 8	28	18	11	6	6	8	8	25.980	117,6	31,5
Örnek 9	25	17	12	6	8	9	10	27.011	105	31,5
Örnek 10	22	18	12	5	7	7	7	23.914	92,4	31,5

Tablo 4.12'de değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesine ilişkin her bir örnek problemin raf sayıları, her bir malzeme grubuna ayrılan raf uzunlukları, her bir malzeme grubuna ayrılan raf yükseklikleri, toplam elleçleme maliyeti, deponun uzunluğu ve deponun genişliği verilmektedir.

5. TARTIŞMA VE SONUÇ

Modern tedarik zinciri ve lojistik yönteminde, müşteri isteklerinin en uygun şartlarda, hızlı ve eksiksiz olarak müşteriye ulaştırılabilmesi ve müşteri isteklerinin karşılanması sırasında tüm lojistik operasyonların entegre ve uyum içerisinde gerçekleştirilmesi gerekmektedir. Müşteri istekleri istenilen şekilde karşılanırken malzemelerin en az stok miktarları sağlanarak gönderilmesi gerekmektedir. Diğer taraftan, siparişlerdeki malzeme çeşitliliği giderek yükselirken sipariş hacimleri ise düşmektedir. Dolayısıyla sık aralıklarla verilerek kısa sürede teslim edilmesi istenen, çeşitliliği yüksek müşteri siparişlerinin karşılanması için; malzeme akışındaki tedarik zinciri ve lojistik sistemi, bu akışı kesintiye uğratmayacak bir düzende kurgulanmalıdır. Tedarik zincirinin ve lojistik sisteminin kurgulanması sırasında üzerinde durulması gereken en önemli unsurlardan birisi de depo ve depolama sürecidir.

Depolama süreci ile tedarik zinciri üzerindeki tüm elemanlar birbirlerine bağlanmaktadır. Malzeme veya hizmet üreten tüm şirketlerin söz konusu üretim sürecinde kullanacakları tüm hammadde, yarı mamul, yardımcı malzemeler, bakım onarım sarf malzemeleri, yedek parçalar ve üretim sürecinin sonunda elde edilen bitmiş ürünlerin uygun koşullarda saklanarak stoklanması ve bu stoklanma sırasında gerçekleştirilen operasyonların tümü depolama sürecinde yapılmaktadır. Depolama sürecinde yer alan operasyonların uyumlu bir düzen içerisinde gerçekleştirilmesi gerekmektedir. Diğer bir deyişle, depolarda stoklanan malzemelerin müşteri talebi gelinceye kadar uygun koşullarda korunarak saklanması; müşteri talebi geldiğinde ise hızlıca toplanarak, hatasız bir şekilde gönderilmesi için depolama operasyonlarının belirli bir sıra ile yapılması gerekmektedir. Birçok operasyonun bir arada gerçekleştirildiği depolarda, tüm operasyonların düzenli bir sıra ile başarılı bir şekilde yürütülmesi uygun bir depo yerleşim düzenlemesine bağlıdır.

Depo yerleşim düzenlemesi deponun kullanım amacı, bu amaca ulaşmak için gerekli olan depo fonksiyonları ile malzeme ve ekipman akışını dikkate alarak; deponun boyutlarını ve raf sistemlerini belirleme sürecidir. Bu süreçte, hangi raf sisteminin

seçileceği, seçilen raf sisteminin uzunluğunun ne olacağı, raf sisteminin yüksekliğinin ve derinliğinin ne olacağı, hangi depolama politikasının deponun amacına uygun olacağı, depo içerisinde nasıl bir yerleşim düzenleneceği, yükleme ve boşaltma alanlarının nerede olacağı ve bu alanların büyüklüğünün ne kadar olacağı gibi depo yerleşim düzenlemesine ilişkin birçok karar verilir. Bu çalışma, söz konusu depo yerleşim düzenlemesi kararlarının birçoğuna cevap verir niteliktir.

Bu çalışmada, literatürde daha önce çalışılmamış olan farklı palet yüksekliğine sahip malzemelerin saklanmasını dikkate alan depo yerleşim düzenlemesi problemi üzerinde durulmuştur. Bu çalışma kapsamında ele alınan depo yerleşim düzenlemesi problemini diğer depo yerleşim düzenlemesi problemlerinden ayıran en temel özellik, aynı raf sırasında yer alan paletler için ayrılmış stok alanlarının farklı yüksekliğe sahip olmasıdır. Diğer bir anlatımla, farklı yüksekliklere sahip paletlerin stoklanması için hazırlanmış her bir depo rafında farklı sayılarda raf yer almakta ve böylece her bir depo rafı farklı kapasiteye sahip olmaktadır. Böylece, depo yerleşimi düzenlemesi sırasında farklı yüksekliklere ve farklı kapasitelere sahip raf sistemleri elde edilmiştir.

Depo yerleşim düzenlemesi problemi NP-zor niteliğe sahiptir. NP-zor niteliğe sahip bir problem olan depo yerleşim düzenlemesi problemi, doktora tezi kapsamında geliştirilmiş ve değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemi haline getirilmiştir. Değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi probleminin çözümünde ise metasezgisel yöntemlerden birisi olan PSO algoritması kullanılmıştır. Ancak, PSO algoritması ile çözülecek söz konusu problemin yapısında kısıtlar bulunmakta; PSO algoritması ise kısıtlı çözüm uzayında etkili bir şekilde çalışamamaktadır. Bu durumun üstesinde gelebilmek için PSO algoritması ile uyumlu çalışan mevcut sınırlandırma yöntemleri kullanılmıştır. Ayrıca, bu çalışmada var olan sınırlandırma yöntemlerine ek olarak PSO algoritması ile uyum içerisinde çalışan “Görünür Yansıtma” ve “Görünür Sönümleme” isminde iki yeni sınırlandırma yöntemi önerilmiştir. Bu yeni sınırlandırma yöntemleri sayesinde, parçacıklar çözüm uzayının sınırları üzerinde daha rahat arama yapabilmektedir.

Depo yerleşim düzenlemesi probleminin çözümünde kullanılan PSO algoritmasının etkili bir şekilde çalışamaması durumu çözüm uzayı büyüdüğünde parçacıkların çözüm uzayına rassal dağılması sırasında ortaya çıkmaktadır. Parçacıkların rassal olarak çözüm

uzayına dağıtılması sırasında ilk iterasyonda bireysel en iyi ve sürünün en iyi değeri belirlenmezse PSO algoritması çalışamamaktadır. Bu çalışmada söz konusu olumsuz durumu ortadan kaldırmak için, parçacıkların daha daraltılmış bir çözüm uzayına rassal dağıtımını sağlayan parçacıkların başlangıç minimum değerleri için atama algoritması geliştirilmiştir. PSO algoritmasındaki başlangıç minimum değerleri geliştirilen yeni algoritma yardımıyla değiştirilerek, parçacıkların daha dar bir çözüm uzayına bırakılması sağlanır. Bu sayede, ilk iterasyonda parçacıkların en iyi değerlerinin ve sürünün en iyi değerlerinin belirlenmesi olasılığı arttırılmaktadır. Çözüm uzayının daraltılmasını sağlayan yeni atama algoritmanın kullanılması ile aynı zamanda, parçacıkların en uygun ve en iyi sonucu bulma olasılığı da artmaktadır. Bu yeni algoritmanın kullanılması ile parçacıklar problemin çözümüne daha hızlı ve daha kısa sürede ulaşmakta ve daha yüksek isabet oranına sahip olmaktadır. Daha dar çözüm uzayına dağılmış parçacıklar ile daha çok parçacık arama yapmakta ve uygun olmayan sonuçlarda arama yapılması azaltılmaktadır.

Değişken yüksekliğe sahip paletler için depo yerleşim düzenlemesi problemine ilişkin geliştirilmiş olan atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemleri için ve standart PSO algoritmasının sınırlandırma yöntemleri için hazırlanan bilgisayar programlarının 10 farklı örnek problem üzerinde çalıştırılarak çözülmesi sonucu elde edilen sonuçlar ayrı ayrı tablolaştırılmıştır. Ardından, her iki çözüme ait sonuç tabloları özet bir tablo haline getirilerek sonuçlar maliyet, isabet oranı, süre ve iterasyon sayısı kriterlerine göre karşılaştırılmıştır. Karşılaştırma sonucunda söz konusu kriterler dikkate alındığında, atama algoritmalı standart PSO algoritmasının sınırlandırma yöntemlerinden doktora tezi kapsamında geliştirilmiş olan “Görünür Yansıtma” yöntemi ile yapılan çözümlerde daha iyi sonuçlar elde edildiği görülmüştür. Son olarak ise kriterler dikkate alındığında en iyi sonucu veren yöntemlerin 10 farklı problem üzerinde depo raf sistemi konfigürasyonu (raf sayısı, raf uzunluğu, raf yüksekliği) ve depo yerleşim boyutlarına (depo eni, depo boyu) ilişkin verdiği sonuçlar sunulmuştur.

Bu çalışmanın depo yerleşim düzenlemesi konusu ile ilgili literatüre sağlayacağı katkılar aşağıda özetlenmektedir:

- Bu çalışma, literatürde depo yerleşim düzenlemesi problemine çözüm arayan çalışmalardan farklı olarak, farklı palet yüksekliğine sahip olan malzemeler için uygun bir depo yerleşim düzenlemesi probleminin modelinin oluşumunu sağlamaktadır.
- Literatürde kısıtlı optimizasyon problemlerinin çözümünde PSO algoritması ile birlikte kullanılan mevcut sınırlandırma yöntemlerine ek olarak "Görünür Yansıtma" ve "Görünür Sönümleme" isimlerinde özgün iki sınırlandırma yöntemi geliştirilmiştir.
- Literatürde kısıtlı optimizasyon problemlerinin çözümünde PSO algoritması ile birlikte kullanılacak; parçacıkların daha daraltılmış bir çözüm uzayına rassal dağıtımını sağlayan "*Parçacıkların Başlangıç Minimum Değerleri İçin Atama Algoritması*" isminde özgün bir algoritma geliştirilmiştir.

Bu çalışmanın gelecek çalışmalara ışık tutması için öneriler aşağıda yer almaktadır.

- Bu çalışmada yer alan depo yerleşim düzenlemesi probleminde depoya giriş ve çıkış için kullanılacak kapı sayıları dikkate alınmıştır. Ancak, kapı ihtiyacının nasıl belirleneceği ile ilgili bir çalışma yapılmamıştır. Kapı ihtiyacının farklı metasezgisel yöntemler veya simülasyon yöntemi kullanarak belirlenmesi ve depo yerleşim düzenlemesi probleminin modeline entegre edilmesi daha ileriki çalışmaların konusu olabilecektir.
- Bu çalışmada, depo yerleşim düzenlemesinin modellenmesi sırasında sadece ortalama elleçleme maliyeti enküçükleyen hesaplama amaçlanmaktadır. Ancak, depo düzenlemesi sırasında ortaya çıkan inşaat maliyeti, elleçleme ekipmanı maliyeti ve raf sistemi maliyeti gibi diğer maliyetler hesaplamalarda değerlendirilmemiştir. Gelecek çalışmalarda bu maliyetler de dikkate alınarak var olan çalışma geliştirilebilir.
- Bu çalışmada, depo yerleşim düzenlemesi modellenirken yalnızca paletlerin farklı yüksekliklere sahip olduğu dikkate alınmıştır. Gelecek çalışmalarda paletlerin farklı yüksekliklere sahip olmasının yanı sıra farklı ağırlıklara da sahip olduğu göz önüne alınabilir.
- Bu çalışmada önerilen atama algoritması sadece PSO algoritmasına uygulanmıştır. Önerilen algoritma, genel bir kullanım yapısına sahiptir ve depo

yerleşim düzenlemesinin farklı şekillerde ortaya çıkabilecek uygulamalarına kolaylıkla uygulanabilir. Bununla birlikte, parçacıkların başlangıç minimum değerleri için atama algoritması farklı metasezgisel yöntemlere de uygulanabilir.

KAYNAKLAR

Abbasi, M., 2011, *Storage, Warehousing and Inventory Management*, Logistics Operations and Management: Concepts and Models, In: Farahani R. Z. (ed.), Rezapour S. (ed.), Kardar L. (ed.), Chapter 10, Elsevier, London-Waltham, 181-197.

Abdullah, Z., Abdullah, A. H., Musa, R. ve Putit, L., 2012, The Effect of Inventory Management, Internal and External Relationships on Customer Service in the Retail Industry, *2012 IEEE Symposium on Humanities, Science and Engineering Research*, 24-27 June 2012 Kuala Lumpur, 1557–1562.

Acar, A. Z., 2010, *Depolama ve Depo Yönetimi*, Nobel Yayıncılık, İstanbul.

Akyol, S. ve Alataş, B., 2012, Güncel Sürü Zekâsı Optimizasyon Algoritmaları, *Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi,* 1, 36-50.

Ashayeri, J. ve Gelders, L., 1985, Warehouse Design Optimization, *European Journal of Operational Research,* 21, 285-294.

Axsater, S., 2006, *Inventory Control*, Springer, New York.

Baker, P. ve Canessa, M., 2009, Warehouse Design: A Structured Approach, *European Journal of Operational Research*, 193(2), 425-436.

Bartholdi, J. J. ve Hackman, S. T., 2011, *Warehouse & distribution science Release 0.92,* http://www2.isye.gatech.edu/~jjb/wh/book/editions/wh-sci-teachers-edition-0.92.pdf, [Ziyaret tarihi: 30 Ocak 2014].

Beasley, D., Bull, D. ve Martin, R., 1993, An Overview of Genetic Algorithms: Part I Fundamentals, *University Computing*, 15, 58-69.

Berg, J. P. v. D., 2007. *Integral Warehouse Management*, Management Outlook Publications, Utrecht.

Berry, J., 1968, Elements of Warehouse Layout, *The International Journal of Production Research*, 7(2), 105-121.

Bidgoli, H., 2010, *The Handbook of Technology Management: Supply Chain Management, Marketing and Advertising, and Global Management*, 2nd edition, John Wiley & Sons, Hoboken.

Bowersox, D. J., Closs, D. J. ve Cooper, M. B., 2002, *Supply Chain Logistics Management*, 2nd edition, McGraw Hill, New York.

Brito, A. E. S. C., 1992, *Configuring Simulation Models Using CAD Techniques: A New Approach to Warehouse Design,* Ph.D. Thesis,Centre for Logistics and Transportation School of Management, Cranfield Instıtute of Technology.

Caron, F., Marchet, G. ve Perego, A., 2000a, Layout Design In Manual Picking Systems: A Simulation Approach, *Integrated Manufacturing Systems*, 11(2), 94-104.

Caron, F., Marchet, G. ve Perego, A., 2000b, Optimal Layout in Low-level Picker-to-Part Systems, *International Journal of Production Research,* 38(1), 101-117.

Choy, K., Chow, H., Poon, T. ve Ho, G., 2012, Cross-Dock Job Assignment Problem in Space-Constrained Industrial Logistics Distribution Hubs With a Single Docking Zone, *International Journal of Production Research,* 50(9), 2439-2450.

Coath, G. ve Halgamuge, S. K., 2003, A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems, *CEC'03 Evolutionary Computation,* 8-12 Dec. 2003, 4, 2419 - 2425.

Coyle, J., Bardi, E. ve Langley, C., 2003, *The Management of Business Logistics: A Supply Chain Perspective,* 7th ed., South-Western/Thomson Learning, Mason.

Dorigo, M. ve Blum, C., 2005, Ant Colony Optimization Theory: A Survey, *Theoretical Computer Science,* 344(2-3), 243-278.

Dowlatshahi, S., 1994, A Modelling Approach to Design of Integrated Facilities, *International Journal of Production Research,* 32(6), 1313-1330.

Dowlatshahi, S., 2012, A framework for the role of warehousing in Reverse Logistics, *International Journal of Production Research,* 50(5), 1265-1277.

Dreo, J., Petrowsk, A., Siarry, P. ve Taillard, E., 2006, *Metaheuristics for Hard Optimization: Simulated Annealing, Tabu Search, Evolutionary and Genetic Algorithms, Ant Colonies,* Springer Berlin Heidelberg, Berlin.

Eberhart, R. C. ve Kennedy, J., 1995, A New Optimizer Using Particle Swarm Theory, *Proceedings of 6th Symposium Micro Machine and Human Science,*4-6 October 1995 Nagoya, 39-43.

Elmas, Ç., 2010, *Yapay Zeka Uygulamaları,* Seçkin Yayıncılık, Ankara.

Ertek, G., 2010, Çapraz Sevkiyat İçin Temel Bilgiler, *Lojistik Dergisi,* 13, 22-27.

Farahani, R. Z., Rezapour, S. ve Kardar, L., 2011, *Logistics Operations and Management: Concepts and Models,* Elsevier, London-Waltham.

Frazelle, E., 2002, *World-Class Warehousing and Material Handling,* McGraw-Hill, USA.

Frazelle, E. H., 2002, *Supply Chain Strategy: The Logistics of Supply Chain Management*, McGraw-Hill, New York.

Gagliardi, J.-P., Ruiz, A. ve Renaud, J., 2008, Space allocation and stock replenishment synchronization in a distribution center, *International Journal of Production Economics*, 115(1), 19-27.

Görçün, Ö. F., 2013, *Depo ve Envanter Yönetimi*, Beta Basım, İstanbul.

Gourdin, K. N., 2006, *Global Logistics Management: A Competitive Advantage for the 21st Century*, 2nd Edition, Blackwell Publishing, Victoria.

Gray, A. E., Karmarkar, U. S. ve Seidmann, A., 1992, Design and Operation of An Order-Consolidation Warehouse: Models and Application. *European Journal of Operational Research*, 58, 14-36.

Gue, K. ve Meller, R., 2009, Aisle configurations for unit-load warehouses, *IIE Transactions*, 41(3), 171–182.

Gue, K. R., Ivanovic, G. ve Meller, R. D., 2012, A Unit-load Warehouse with Multiple Pickup and Deposit Points and Non-traditional Aisles, *Transportation Research Part E: Logistics and Transportation Review*, 48(4), 795-806.

Gu, J., Goetschalckx, M. ve McGinnis, L. F., 2010, Research on Warehouse Design and Performance Evaluation: A Comprehensive Review, *European Journal of Operational Research*, 203(3), 539-549.

Gülsün, B., Tuzkaya, G. ve Bildik, E., 2008, Tersine Lojistikte Ağ Tasarımı : Bir Tavlama Benzetimi Yaklaşımı, *Sigma : Mühendislik ve Fen Bilimleri Dergisi*, 26(1), 68-80.

Heragu, S. S., Du, L., Mantel, R. J. ve Schuur, P. C., 2005, Mathematical Model for Warehouse Design and Product Allocation, *International Journal of Production Research*, 42(3), 327-338.

Hsieh, L.-f. ve Tsai, L., 2006, The Optimum Design of a Warehouse System On Order Picking Efficiency, *The International Journal of Advanced Manufacturing Technology*, 28(5-6), 626-637.

Huertas, J. I., Ramirez, J. D. ve Salazar, F. T., 2007, Layout Evaluation of Large Capacity Warehouses, *Facilities*, 25(7/8), 259-270.

Hugos, M., 2003, *Essentials of Supply Chain Management.* John Wiley & Sons Inc, New Jersey.

Johnson, D. S., Cecilia R. Aragon, L. A. M. ve Schevon, C., 1989, Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning, *Operations Research*, 37(6), 865-892.

Karaboğa, D., 2011, *Yapay Zekâ Optimizasyon Algoritmaları,* Nobel Yayıncılık, Ankara.

Keskintürk, T. ve Şahin, S., 2009, Doğrusal Olmayan Regresyon Analizinde Gerçek Değer Kodlamalı Genetik Algoritma. *Istanbul Ticaret Üniversitesi Sosyal Bilimler Dergisi*, 15, 167-178.

Kosner, R. D., Le-Duc, T. ve Roodbergen, K. J., 2007, Design and control of warehouse order picking: A literature review, *European Journal of Operational Research,* Issue 182, 481-501.

Lai, K., Xue, J. ve Zhang, G., 2002, Layout Design for A Paper Reel Warehouse: A two-stage Heuristic Approach, *International Journal of Production Economics,* 75(3), 231-243.

Lambert, D. M., Stock, J. R. ve Ellram, L. M., 1998, *Fundamentals of Logistics Management*, Irwin McGraw-Hill, Boston.

Larson, T., March, H. ve Kusiak, A., 1997, A Heuristic Approach to Warehouse Layout With Class-based Storage, *IIE Transactions,* 29(4), 337-348.

Le-Duc, T. ve Koster, R. D., 2005, Travel Distance Estimation and Storage Zone Optimization In a 2-block Class-based Storage Strategy Warehouse, *International Journal of Production Research,* 43(17), 3561-3581.

Lee, H., Padmanabhan, V. ve Whang, S., 1997, The Bullwhip Effect in Supply Chains, *Sloan Management Review*, 3(92), 38.

Li, H., He, X., Xie, X., Li, L., Zhou, J., Li X., 2010, A New Boundary Condition for Particle Swarm Optimization, *Journal of Converge Information Technology,* 5(9), 215–221.

Lin, C.H. ve Lu, I.Y., 1999, The Procedure of Determining the Order Picking Strategies in Distribution Center, *International Journal of Production Economics,* 60-61, 301-307.

Macro, J. ve Salmi, R., 2002. A Simulation Tool to Determine Warehouse Efficiencies and Storage Allocations. *Proceedings of the 2002 Winter Simulation Conference,* 8-11 December 2002, San Diego, 2, 1274-1281.

Mitchell, M., 1998, *An Introduction to Genetic Algorithms,* MIT Press, London.

Murphy, P. R. ve Wood, D. F., 2011, *Contemporary Logistics,* International Edition, Pearson Education, USA.

Napolitano, M., 2000, *Making The Move To Cross Docking*, Warehousing Education and Research Council, Oak Brook.

Nowicki, E. ve Smutnicki, C., 1996, A Fast Tabu Search Algorithm for the Permutation Flow-Shop Problem, *European Journal of Operational Research,* 91(1), 160-175.

Önüt, S., Tuzkaya, U. R. ve Doğan, B., 2008, A Particle Swarm Optimization Algorithm for the Multiple-level Warehouse Layout Design Problem, *Computers & Industrial Engineering,* 54(4), 783-799.

Öztürkoğlu, Ö., Gue, K. R. ve Meller, R. D., 2012, Optimal Unit-load Warehouse Designs for Single-command Operations, *IIE Transactions,* 44(6), 459-475.

Parikh, P. J. ve Meller, R. D., 2008, Selecting between batch and zone order picking strategies in a distribution center, *Transportation Research Part E 44,* 696-719.

Park, B. C., 2012, *Order Picking: Issues, Systems and Models*. In: R. Manzini, (ed.) Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems, London, Springer-Verlag London, 1-51.

Park, Y. H. ve Webster, D. B., 1989, Modelling of Three-dimensional Warehouse Systems, *International Journal of Production Research,* 27(6), 985-1003.

Petersen, C. G., 2002, Considerations in order picking zone configuration, *International Journal of Operations & Production Management,* 27(7), 793–805.

Petersen, C. G. ve Aase, G. R., 2004, Improving Order-Picking Performance Through The Implementation of Class-Based Storage, *International Journal of Physical Distribution & Logistics Management,* 34(7), 534-544.

Pohl, L. M., Meller, R. D. ve Gue, K. R., 2009, An Analysis of Dual-Command Operations in Common Warehouse Designs, *Transportation Research Part E,* 5, 367-379.

Queirolo, F., Tonelli, F., Schenone, M., Paolo, N., Zunino, I., 2002, Warehouse Layout Design: Minimizing Travel Time with A Genetic And Simulative Approach - Methodology And Case Study, *Proceedings 14th European Simulation Symposium,* September 2002, Dresden.

Richards, G., 2003, *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse,* Kogan Page, London, Philadelphia, New Delhi.

Roodbergen, K. J. ve de Koster, R., 2001, Routing Order Pickers In a Warehouse With a Middle Aisle, *European Journal of Operational Research,* 133(1), 32-43.

Roodbergen, K. J. ve Vis, I. F. A., 2006, A Model for Warehouse Layout, *IIE Transactions,* 38(10), 799-811.

Rosenblatt, M. J. ve Roll, Y., 1984, Warehouse Design With Storage Policy Considerations, *International Journal of Production Research,* 22(5), 809-821.

Rouwenhorst, B., Reuter, B., Stockrahm, V., van Houtum, G.J., Mantela, R.J., Zijm, W.H.M., 2000, Warehouse Design and Control: Framework and Literature Review, *European Journal of Operational Research,* 3(515-533), 122.

Rushton, A., Croucher, P. ve Baker, P., 2010, *The Handbook of Logistics and Distribution Management.* 4th ed., Kogan Page, London.

Salvendy, G., 2001, *Handbook of Industrial Engineering: Technology And Operations Management.* Canada, John Wiley & Sons.

Scott, C., Lundgren, H. ve Thompson, P., 2011, *Guide to Supply Chain Management,* Springer, Berlin.

Shiau, J.-Y. ve Lee, M.-C., 2010, A Warehouse Management System with Sequential Picking for Multi-Container Deliveries, *Computers & Industrial Engineering,* 58(3), 382-392.

Simchi-Levi, D., Kaminsky, P. ve Simchi-Levi, E., 2003, *Designing & Managing the Supply Chain Concepts, Strategies & Case Studies.* 2. ed. McGraw-Hill/Irwin, New York.

Sople, V. V., 2007, *Logistics Management: The Supply Chain Imperatives*, Pearson Education, Singapore.

Strack, G. ve Pochet, Y., 2010, An Integrated Model for Warehouse and Inventory Planning, *European Journal of Operational Research,* 204(1), 35-50.

Stroh, M., 2006, *A Practical Guide to Transportation and Logistics,* Logistics Network Inc., Dumont, NJ..

Tompkins, J., White, J., Bozer, Y. ve Tanchoco, J., 2010, *Facilities Planning.* 4.ed., John Wiley & Sons, Hoboken, NJ..

van den Berg, J. P., 2007, *Integral Warehouse Management,* Management Outlook Publications, Utrecht.

van den Berg, J. ve Zijm, W., 1999, Models for Warehouse Management : Classification and Examples, *International Journal of Production Economics,* 59, 519–528.

Voortman, C., 2004, *Global Logistics Management,* Juta Academic, Johannesburg.

Walter, D., 2003, *Logistics An Introduction to Supply Chain Management,* Palgrave Macmillan, New York.

Wisner, J., Tan, K. ve Leong, G., 2008, *Principles Of Supply Chain Management : A Balanced Approach,* South-Western Cengage Learning, Mason, OH.

Xu, S. ve Rahmat-Samii, Y., 2007, Boundary Conditions in Particle Swarm Optimization Revisited, *IEEE Transactions On Antennas and Propagation,* 55(3), 760-765.

Yang, L. ve Sun, Y., 2004, Expected Value Model for A Fuzzy Random Warehouse Layout Problem, *IEEE International Conference Proceedings on Fuzzy Systems,* 751-756.

Yiğit, V. ve Türkbey, O., 2003, Tesis Yerleşim Problemlerine Sezgisel Metotlarla Yaklaşım, *Gazi Üniv. Müh. Mim. Fak. Der.*, 18(4), 45-56.

Zhang, G. Q. ve Lai, K., 2006, Combining Path Relinking And Genetic Algorithms for the Multiple-level Warehouse Layout Problem, *European Journal of Operational Research,* 169(2), 413-425.

Zhang, G. Q. ve Lai, K., 2010, Tabu Search Approaches for the Multi-level Warehouse Layout Problem with Adjacency Constraints, *Engineering Optimization,* 42(8), 775-790.

Zhang, G. Q., Xue, J. ve Lai, K. K., 2002, A Class of Genetic Algorithms For Multiple-Level Warehouse Layout Problems, *International Journal of Production Research,* 40(3), 731-744.

Zhang, J.-M. ve Xie, L., 2009, Particle Swarm Optimization Algorithm For Constrained Problems, *Asia-Pacific Journal Of Chemical Engineering*, 4, 437-442.

Zhu, J. 2., 2009, A Modified Particle Swarm Optimization Algorithm, *Journal of Computers,* 4(12), 1231–1236.

EKLER

EK 1. Tez kapsamında hazırlanan bilgisayar programının kaynak kodları

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;

namespace PSO_TEZ
{
   class PSO_SINGLE
   {
      //Stok Parametreleri
      private double DockDim, Cost;
      private int[] AdetArr;
      private double[] OlasılıkArr, PaletYukArr; public double[] InertArr;
      private int ProductNum;
      //Depo Parametreleri
      int[] NyMinArr, NyMaxArr;
      int xMin, xMax, yMax, PSOType;
      /*Konuma bağlı olarak değişen depo parametreleri veri alımı*/
      private double Wx;
      private double Wy;
      private double Ax;
      private double Ay;

      private int  YillikTalep, SumDock2, DockNum, Deneme;
      //PSO Parametreleri
      private int Iteration, Part, VMax, SinirlandirmaMethodu;
      private static readonly Random RastgeleDeger = new Random();
      double MR, SocialParam, PersonalParam;

      private string[] GlobalBestArray;

      public string yazdirabi, elapsedMS;
      //
      double InertMax, InertMin, Inert;
      double OgrenmeOrani;
      bool InertControl, DebControl;
      private int[] MaxYukseklikArr, MinYukseklikArr;
```

```
public PSO_SINGLE(/*WareHouse Param*/double wx, double wy, double Ax,
double Ay, int yillikTalep, double cost, double xMax, double yMax, int dockNum,
double dockDim, double[] olasilikArr, int[] adetArr, int[] maxYukseklikArr,double[]
paletYukArr, int productNum, /*Pso Param*/int iteration, double socialParam, double
personalParam, int part, int vMax, int sinirlandirmaMethodu, double mR, int PSOType,
double InertMax, double InertMin, double Inert, bool InertControl, int deneme, bool
debControl)
    {

        #region ---/Depo Parametreler/---
        YillikTalep = yillikTalep; Cost = cost;
        DockDim = dockDim; DockNum = dockNum; Deneme = deneme;
        this.OlasılıkArr = olasilikArr; this.AdetArr = adetArr; this.PaletYukArr =
paletYukArr;
        this.ProductNum = productNum; this.GlobalBestArray = new
string[ProductNum * 2 + 3];
        this.xMax = (int)xMax; this.yMax = (int)yMax;
        this.MaxYukseklikArr = maxYukseklikArr ;
        this.SinirlandirmaMethodu = sinirlandirmaMethodu;
        #endregion

        this.PSOType = PSOType;
        this.DebControl = debControl;
        #region ---/Eğim Hesaplanması\---

        #region ---/ X Eğim Hesaplanması\---
        double[] XYi = new double[(Convert.ToInt32(xMax))]; double[] FarkXX =
new double[(Convert.ToInt32(xMax))];
        int[] XXi = new int[(Convert.ToInt32(xMax))];
        double TopXYi = 0.0; double TopUst = 0.0; double TopFarkX = 0.0;

        double Ortx = (xMax + 1) / 2;
        for (int i5 = 0; i5 < xMax; i5++)
        {
           XXi[i5] = i5 + 1;
           XYi[i5] = (((-3 * (3 + 2.7 * XXi[i5])) / (5.4 * XXi[i5])) + (((2.1 * XXi[i5]
+ 6) * (2.1 * XXi[i5] + 6)) / (2.7 * XXi[i5] * 4))) ;
           TopXYi += XYi[i5];
           FarkXX[i5] = XXi[i5] - Ortx;//Xi ile ortalama x arasındaki fark
        }

        double OrtXYi = TopXYi / xMax;//Ortalama Y
        for (int i6 = 0; i6 < xMax; i6++)
        {
           double Ust = 0.0;
           Ust = FarkXX[i6] * (XYi[i6] - OrtXYi); //Üst değerlerinin toplamı
           TopUst += Ust;
        }
```

```
            #endregion

            #region ---/ Y değerlerinin Eğimlerinin Hesaplanması---\

            double[] YiEgim = new double[productNum];
            double[] TopProbArr = new double[productNum];
            for (int i7 = 0; i7 < productNum; i7++)
            {
               TopProbArr[i7] = 0.0000;
               for (int j7 = productNum - 1; j7 >= i7; j7--)
               {
                  TopProbArr[i7] += olasilikArr[j7];
                  if (j7 == i7)
                     TopProbArr[i7] -= olasilikArr[j7] / 2;
               }
               YiEgim[i7] = Math.Round((0.9 ) * TopProbArr[i7], 4);

            }

            #endregion

            #region ---/ Z değerlerinin Eğimlerinin Hesaplanması\---
            double[] ZiEgim = new double[productNum];
            for (int i8 = 0; i8 < productNum; i8++)
            {
               double[] ZYi = new double[(Convert.ToInt32(MaxYukseklikArr[i8]))];
double[] FarkZX = new double[(Convert.ToInt32(MaxYukseklikArr[i8]))];
               int[] ZXi = new int[(Convert.ToInt32(MaxYukseklikArr[i8]))];
               double TopZYi = 0.0; double TopZUst = 0.0; double TopFarkZX = 0.0;

               double Ortz = (MaxYukseklikArr[i8] + 1) / 2.0;
               for (int i15 = 0; i15 < MaxYukseklikArr[i8]; i15++)
               {
                  ZYi[i15] = (OlasılıkArr[i8] * PaletYukArr[i8] *
Math.Exp(Math.Pow(ZXi[i15] / 2.0, 0.5))) ;
                  TopZYi += ZYi[i15];
                  FarkZX[i15] = ZXi[i15] - Ortz;//Xi ile ortalama x arasındaki fark
                  TopFarkZX += Math.Pow(FarkZX[i15], 2);//Taban değerinin bulunması
               }

               double OrtZYi = TopZYi / MaxYukseklikArr[i8];//Ortalama Y
               for (int i16 = 0; i16 < MaxYukseklikArr[i8]; i16++)
               {
                  double Ust = 0.0;
                  Ust = FarkZX[i16] * (ZYi[i16] - OrtZYi); //Üst değerlerinin toplamı
                  TopZUst += Ust;
               }

               ZiEgim[i8] = Math.Round(TopZUst / TopFarkZX, 4); //Eğim X bulunması
```

```
            }

            #endregion


            #region ---/Gecis Oranları Hesaplanması\---

            double[] GecisY = new double[productNum];
            double[] GecisZ = new double[productNum];
            double[] GecisYZ = new double[productNum];
            double[] XsonraKalan = new double[productNum]; double[] Yideger = new
double[productNum];
            double[] Zideger = new double[productNum];
            double OrtalamaGecisY = 0.0; double OrtalamaGecisZ = 0.0; double
OrtalamaUrun = 0.00;

            for (int k1 = 0; k1 < productNum; k1++)
            {
               GecisY[k1] = Math.Round(YiEgim[k1] / XXEgim, 3);
               GecisZ[k1] = Math.Round(ZiEgim[k1] / XXEgim, 3);
               GecisYZ[k1] = Math.Round(ZiEgim[k1] / YiEgim[k1], 3);
               OrtalamaGecisY += Math.Round(GecisY[k1] / productNum, 5);
               OrtalamaGecisZ += Math.Round(GecisZ[k1] / productNum, 5);

               OrtalamaUrun += adetArr[k1] / (productNum * 2);

            }
            double OrtalamaX = Math.Ceiling(Math.Pow((OrtalamaUrun /
(OrtalamaGecisY * OrtalamaGecisZ)), (1 / 3.0)) * productNum);
            if (OrtalamaX > xMax)
               OrtalamaX = xMax;

            double topyideger = 0;
            for (int j2 = 0; j2 < productNum; j2++)
            {
               XsonraKalan[j2] = adetArr[j2] / (OrtalamaX * 2);
               Yideger[j2] = Math.Ceiling(Math.Pow((XsonraKalan[j2] / GecisYZ[j2]),
0.5) * GecisYZ[j2]);
               Zideger[j2] = Math.Ceiling(XsonraKalan[j2] / Yideger[j2]);
               if (Zideger[j2] > MaxYukseklikArr[j2])
               {
                  Zideger[j2] = MaxYukseklikArr[j2];
                  //Düşürülen Z değeri için yeni y değeri belirlenir
                  Yideger[j2] = Math.Ceiling(XsonraKalan[j2] / Zideger[j2]);
               }

               topyideger += Yideger[j2];

            }
```

```
bool topyidegerkontrol;

if (topyideger > yMax)
{
   topyidegerkontrol = true;

   while (topyidegerkontrol)
   {
      for (int pnr = 0; pnr < productNum; pnr++)
      {
         {
            topyideger -= Yideger[pnr];
            Zideger[pnr] += 1;
            Yideger[pnr] = Math.Ceiling(XsonraKalan[pnr] / Zideger[pnr]);
         }
      }
      int zdegerikontrol = 0;
      for (int i3 = 0; i3 < productNum; i3++)
      {

         if (Zideger [i3] == MaxYukseklikArr[i3])
            zdegerikontrol +=1;
      }

      if (topyideger > yMax)
      {
         if (zdegerikontrol == productNum)
            topyidegerkontrol = false;
      }
      else
      {
         topyidegerkontrol = false;
      }

   }
}


if (topyideger > yMax)
{
   topyidegerkontrol = true;

   while (topyidegerkontrol)
   {
      OrtalamaX +=1;
      topyideger = 0;
      for (int pnr = 0; pnr < productNum; pnr++)
      {
```

```
                topyideger += Yideger[pnr];
            }

            if (topyideger > yMax)
                topyidegerkontrol = false;
        }
    }

    #endregion

    for (int t = 0; t < 8; t++)
    {
        sinirlandirmaMethodu = t;
        for (int e = 0; e < deneme; e++)
        {

            #region ---/Minimum ve Maksimum Kısıtların Hesaplanması\---
            NyMinArr = new int[ProductNum];
            NyMaxArr = new int[ProductNum];
            MinYukseklikArr = new int[ProductNum];
            this.xMin = 1;
            xMin = Convert.ToInt32(Math.Ceiling((OrtalamaX - XSapma)));
            for (int j22 = 0; j22 < productNum; j22++)
            {
                NyMinArr[j22] = Convert.ToInt32(Yideger[j22]);
                NyMaxArr[j22] = Convert.ToInt32(Yideger[j22] + YSapma + 2);
                MinYukseklikArr[j22] = Convert.ToInt32(Zideger[j22]);

            }

            #endregion

            #region --/Pso Parametreleri/---
            Iteration = iteration; SocialParam = socialParam; PersonalParam =
personalParam; Part = part; VMax = vMax; SinirlandirmaMethodu =
sinirlandirmaMethodu;
            this.MR = mR;
            #endregion


            getStartIterationProcess();

            int milliseconds = 2000;
            System.Threading.Thread.Sleep(milliseconds);
        }
        yazdirabi += "Sınırlandırma" + t + "\n";
    }
}
```

```
public PSO_SINGLE(/*WareHouse Param*/double wx, double wy, double Ax,
double Ay, int yillikTalep, double cost, double xMax, double yMax, int dockNum,
double dockDim, double[] olasilikArr, int[] adetArr, int[] maxYukseklikArr, double[]
paletYukArr, int productNum, /*Pso Param*/int iteration, double socialParam, double
personalParam, int part, int vMax, int sinirlandirmaMethodu, double mR, int PSOType,
double InertMax, double InertMin, double Inert, bool InertControl, int deneme, bool
debControl, double ogrenmeOrani)
    {


        #region ---/Depo Parametreler/---
        YillikTalep = yillikTalep; Cost = cost;
        DockDim = dockDim; DockNum = dockNum;
        this.OlasılıkArr = olasilikArr; this.AdetArr = adetArr; this.PaletYukArr =
paletYukArr;
        this.ProductNum = productNum; this.GlobalBestArray = new
string[ProductNum * 2 + 3];
        this.xMax = (int)xMax; this.yMax = (int)yMax;
        this.MaxYukseklikArr = maxYukseklikArr;
        this.SinirlandirmaMethodu = sinirlandirmaMethodu;
        InertArr = new double[ProductNum];
        #endregion


        #region --- PSO Type ---

        this.PSOType = PSOType;
        this.DebControl = debControl;
        this.InertMax = InertMax; this.InertMin = InertMin; this.Inert = Inert;
        this.InertControl = InertControl;
        this.OgrenmeOrani = ogrenmeOrani;
        #endregion

        #region ---/Eğim Hesaplanması\---

        #region ---/ X Eğim Hesaplanması\---
        double[] XYi = new double[(Convert.ToInt32(xMax))]; double[] FarkXX = new
double[(Convert.ToInt32(xMax))];
        int[] XXi = new int[(Convert.ToInt32(xMax))];
        double TopXYi = 0.0; double TopUst = 0.0; double TopFarkX = 0.0;

        double Ortx = (xMax + 1) / 2;
        for (int i5 = 0; i5 < xMax; i5++)
        {
           XXi[i5] = i5 + 1;
           TopXYi += XYi[i5];
           FarkXX[i5] = XXi[i5] - Ortx;//Xi ile ortalama x arasındaki fark
           TopFarkX += Math.Pow(FarkXX[i5], 2);//Taban değerinin bulunması
        }
```

```
            double OrtXYi = TopXYi / xMax;//Ortalama Y
            for (int i6 = 0; i6 < xMax; i6++)
            {
               double Ust = 0.0;
               Ust = FarkXX[i6] * (XYi[i6] - OrtXYi); //Üst değerlerinin toplamı
               TopUst += Ust;
            }

            #endregion

            #region ---/ Y değerlerinin Eğimlerinin Hesaplanması---\

            double[] YiEgim = new double[productNum];
            double[] TopProbArr = new double[productNum];
            for (int i7 = 0; i7 < productNum; i7++)
            {
               TopProbArr[i7] = 0.0000;
               for (int j7 = productNum - 1; j7 >= i7; j7--)
               {
                  TopProbArr[i7] += olasilikArr[j7];
                  if (j7 == i7)
                     TopProbArr[i7] -= olasilikArr[j7] / 2;
               }
               YiEgim[i7] = Math.Round((0.9) * TopProbArr[i7], 4);

            }

            #endregion

            #region ---/ Z değerlerinin Eğimlerinin Hesaplanması\---
            double[] ZiEgim = new double[productNum];
            for (int i8 = 0; i8 < productNum; i8++)
            {
               double[] ZYi = new double[(Convert.ToInt32(MaxYukseklikArr[i8]))];
double[] FarkZX = new double[(Convert.ToInt32(MaxYukseklikArr[i8]))];
               int[] ZXi = new int[(Convert.ToInt32(MaxYukseklikArr[i8]))];
               double TopZYi = 0.0; double TopZUst = 0.0; double TopFarkZX = 0.0;

               double Ortz = (MaxYukseklikArr[i8] + 1) / 2.0;
               for (int i15 = 0; i15 < MaxYukseklikArr[i8]; i15++)
               {
                  ZXi[i15] = i15 + 1;
                  ZYi[i15] = (OlasılıkArr[i8] * Math.Exp(Math.Pow(ZXi[i15] / 2.0, 0.5))) ;
                  TopZYi += ZYi[i15];
                  FarkZX[i15] = ZXi[i15] - Ortz;//Xi ile ortalama x arasındaki fark
                  TopFarkZX += Math.Pow(FarkZX[i15], 2);//Taban değerinin bulunması
               }
```

```
            double OrtZYi = TopZYi / MaxYukseklikArr[i8];//Ortalama Y
            for (int i16 = 0; i16 < MaxYukseklikArr[i8]; i16++)
            {
               double Ust = 0.0;
               Ust = FarkZX[i16] * (ZYi[i16] - OrtZYi); //Üst değerlerinin toplamı
               TopZUst += Ust;
            }

         }

         #endregion


         #endregion

         #region --/Pso Parametreleri/---
         Iteration = iteration; SocialParam = socialParam; PersonalParam =
personalParam; Part = part; VMax = vMax; SinirlandirmaMethodu =
sinirlandirmaMethodu;
         this.MR = mR;
         #endregion

         #region ---/Gecis Oranları Hesaplanması\---

         double[] GecisY = new double[productNum];
         double[] GecisZ = new double[productNum];
         double[] GecisYZ = new double[productNum];
         double[] XsonraKalan = new double[productNum]; double[] Yideger = new
double[productNum];
         double[] Zideger = new double[productNum];
         double OrtalamaGecisY = 0.000; double OrtalamaGecisZ = 0.000; double
OrtalamaUrun = 0.00;

         for (int k1 = 0; k1 < productNum; k1++)
         {
            GecisY[k1] = Math.Round(YiEgim[k1] / XXEgim, 3);
            GecisZ[k1] = Math.Round(ZiEgim[k1] / XXEgim, 3);
            GecisYZ[k1] = Math.Round(ZiEgim[k1] / YiEgim[k1], 3);
            OrtalamaGecisY += Math.Round(GecisY[k1] / productNum, 5);
            OrtalamaGecisZ += Math.Round(GecisZ[k1] / productNum, 5);


         }
         //X miktarının hesaplanması ve kontrolü
         double OrtalamaX = Math.Ceiling(Math.Pow((OrtalamaUrun /
(OrtalamaGecisY * OrtalamaGecisZ)), (1 / 3.0)) * productNum);
         if (OrtalamaX > xMax)
            OrtalamaX = xMax;
```

```
        //Xden sonra kalan ürün sayılarının hesaplanması ve y-z değerlerinin
belirlenmesi
        double  topyideger = 0;
        for (int j2 = 0; j2 < productNum; j2++)
        {
           XsonraKalan[j2] = adetArr[j2] / OrtalamaX;
           Yideger[j2] = Math.Ceiling(Math.Pow((XsonraKalan[j2] / GecisYZ[j2]), 0.5)
* GecisYZ[j2]);
           Zideger[j2] = Math.Ceiling(XsonraKalan[j2] / Yideger[j2]);
           if (Zideger[j2] > MaxYukseklikArr[j2])
           {
              //Z değeri maksimum yükseklikten fazla ise maksimum yüksekliğe indirilir.
              Zideger[j2] = MaxYukseklikArr[j2];
              //Düşürülen Z değeri için yeni y değeri belirlenir
              Yideger[j2] = Math.Ceiling(XsonraKalan[j2] / Zideger[j2]);
           }
           topyideger += Yideger[j2];
        }

        if (topyideger > yMax)
        {
           for (int pnr=0; pnr < productNum; pnr++)
           {

           }

        if (topyideger > yMax)
           for (int pnr = 0; pnr < productNum; pnr++)
           {
              if ((Zideger[pnr] + 1 <= MaxYukseklikArr[pnr]) && (topyideger > yMax))
              {
                 topyideger -= Yideger[pnr];
                 Yideger[pnr] = Math.Ceiling(XsonraKalan[pnr] / Zideger[pnr]);
                 topyideger += Yideger[pnr];
              }

           }

        }


        #endregion


              for (int e = 0; e < deneme; e++)
              {

                 #region ---/Minimum ve Maksimum Kısıtların Hesaplanması\---
```

```
                NyMinArr = new int[ProductNum];
                NyMaxArr = new int[ProductNum];
                MinYukseklikArr = new int[ProductNum];
                this.xMin = 0;
                xMin = Convert.ToInt32(Math.Ceiling((OrtalamaX - XSapma)));
                for (int j22 = 0; j22 < productNum; j22++)
                {
                   NyMinArr[j22] = Convert.ToInt32(1);
                   MinYukseklikArr[j22] = Convert.ToInt32(1);

                }

                #endregion

                getStartIterationProcess();
                int milliseconds = 2000;
                System.Threading.Thread.Sleep(milliseconds);
            }

        }

        private void getStartIterationProcess()
        {
            int x = 0; int[] y = new int[ProductNum]; int[] z = new int[ProductNum]; int[] v
= new int[ProductNum * 2 + 1];
            /*
            int x1 = 0, x2 = 0, x3 = 0, x4 = 0, x5 = 0, x6 = 0, x7 = 0, x8 = 0, x9 = 0, x10 = 0,
x11 = 0, x12 = 0, x13 = 0, x14 = 0, x15 = 0;
            int Z1 = 0, Z2 = 0, Z3 = 0, Z4 = 0, Z5 = 0, Z6 = 0, Z7 = 0, Z8 = 0, Z9 = 0, Z10 =
0, Z11 = 0, Z12 = 0, Z13 = 0, Z14 = 0, Z15 = 0;
            int Y1 = 0;
            int v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16, v17,
v18, v19, v20, v21, v22, v23, v24, v25, v26, v27, v28, v29, v30, v31;
            */
            double[] uygunluksonuc = new double[Part];
            int[,] ajan = new int[Part, 2 * ((ProductNum * 2) + 1)];/*Konum, Parçacık
sayısı, değişken sayısı */
            int[,] localBest = new int[Part, ProductNum * 2 + 1];/*Konum, Parçacık sayısı,
lokal sayısı */
            double[] localBestUygunluk = new double[Part]; /*Konum, Parçacık sayısı */
            double[] ajanViolation = new double[Part]; double[] localBestViolation = new
double[Part];

            //Timer
            Stopwatch sw = new Stopwatch();
            sw.Start();
            #region --- İlk İterasyon İşlemleri ---
            //İlk itarasyon gerçekleştiriliyor
```

```
        //Tüm parçacıklar x1, x2, x3, x4, x5, v1, v2, v3, v4, v5 için rastgele değerler
atanıyor

        //Uygun olmayan en kötü sonuçlu Global Best atanması
        for (int g = 0; g < ProductNum; g++)
        {
           GlobalBestArray[g] = Convert.ToString(NyMinArr[g]);
        }

        for (int h = 0; h < ProductNum; h++)
        {
           GlobalBestArray[ProductNum + h] =
Convert.ToString(MaxYukseklikArr[h]);
        }

        GlobalBestArray[ProductNum * 2] = Convert.ToString(xMax);

        GlobalBestArray[ProductNum * 2 + 1] = Convert.ToString(0);
        GlobalBestArray[ProductNum * 2 + 2] =
Convert.ToString(getAmacFonksiyonu(xMax, NyMinArr, MaxYukseklikArr) - 1);


        for (int i = 0; i < Part; i++)
        {
           /*Minimum ve Maksimum hesaplamaları kullanılarak rastgele değerleri
üretimi */
           for (int j = 0; j < ProductNum; j++)
           {
              z[j] = RastgeleDeger.Next(MinYukseklikArr[j], MaxYukseklikArr[j] + 1);

              if (j == ProductNum - 1)
                 x = RastgeleDeger.Next(xMin, xMax + 1);
           }
           // Hızların ilk değerlerinin atanması
           for (int r = 0; r < ProductNum * 2 + 1; r++)
              v[r] = RastgeleDeger.Next(-VMax, VMax + 1);

           //Uygunluk sonucu hesaplanıyor
           uygunluksonuc[i] = getAmacFonksiyonu(x, y, z);
           //Parçacık başına uygunluklar kontrol edliyor böylece global parçacığımızın
indexi alınıyor

           if ((uygunluksonuc[i] < Convert.ToDouble(GlobalBestArray[ProductNum * 2
+ 2])))// && (i != 0))
           {
              for (int g = 0; g < ProductNum; g++)
              {
                 GlobalBestArray[g] = Convert.ToString(y[g]);
              }
```

```
                for (int h = 0; h < ProductNum; h++)
                {
                    GlobalBestArray[ProductNum + h] = Convert.ToString(z[h]);
                }

                GlobalBestArray[ProductNum * 2] = Convert.ToString(x);
                GlobalBestArray[ProductNum * 2 + 1] = Convert.ToString(1);
                GlobalBestArray[ProductNum * 2 + 2] =
Convert.ToString(uygunluksonuc[i]);
            }


            //Parçacıkların x1, x2, x3, x4, x5, v1, v2, v3, v4, v5
            for (int d = 0; d < ProductNum; d++)
            {
                //Updated For y
                ajan[i, d] = y[d];
                //Updated For z
                ajan[i, ProductNum + d] = z[d];
                //Updated For x
                if (d == 0)
                {
                    ajan[i, ProductNum * 2] = x;

                }
            }

            //Updated For v
            for (int t = 0; t < ProductNum * 2 + 1; t++)
                ajan[i, ProductNum * 2 + 1 + t] = v[t];

            //Updated For Local Best
            for (int u = 0; u < ProductNum; u++)
            {
                y[u] = 0;
                localBest[i, ProductNum + u] = z[u];
                z[u] = 0;
                if (u == 0)
                {
                    localBest[i, ProductNum * 2] = x;
                    x = 0;
                }
            }
            /*
             //Yazdırma
             if (i == 0)
                 yazdirabi += "1. İterasyon \n";
             for (int w = 0; w < ProductNum * 2 + 1; ++w)
```

```
            {
                yazdirabi += ajan[i, w];
                if (ProductNum * 2 == w)
                    yazdirabi += "\n";
            }

            if (i == Part - 1)
            {
                yazdirabi += "GlobalBest : ";
                for (int w = 0; w < ProductNum * 2 + 3; ++w)
                {
                    if (w == ProductNum * 2 + 2)
                    yazdirabi += GlobalBestArray[w];
                    yazdirabi += ",";
                    if (ProductNum * 2 + 2 == w)
                        yazdirabi += "\n";
                }
            } */

        }

        for (int h = 0; h < Part; h++)
            localBestUygunluk[h] = uygunluksonuc[h];
        #endregion

        #region --Sınır Düzenlemeleri---
        int topNyMin = 0;
        // Yatay için ayrılan min stok gözü sayısı hesaplanır.
        for (int pn = 0; pn < ProductNum; pn++)
        {
            NyMinArr[pn] = Convert.ToInt32(Math.Ceiling((AdetArr[pn] / (xMax *
MaxYukseklikArr[pn] * 2.00))));
            MinYukseklikArr[pn] = 1;
            topNyMin += NyMinArr[pn];
        }

        //X yönündeki sınır belirlenir.
        xMin = 1;

        // Y yönündeki max sınırların belirlenmesi

        topNyMin = yMax - topNyMin;

        for (int pe = 0; pe < ProductNum; pe++)
            NyMaxArr[pe] = NyMinArr[pe] + topNyMin;

        #endregion

        #region --- İterasyon İşlemleri ---
```

```
        //İterasyon
        for (int j = 1; j < Iteration; j++)
        {
           for (int k = 0; k < Part; ++k)
           {

              //yeni x1, x2, x3, x4, x5, v1, v2, v3, v4, v5
              for (int l = 0; l < ProductNum * 2 + 1; ++l)
              {
                 #region --- Updated VELOCITY ---
                 switch (PSOType)
                 {
                    // Update VeloCity for PSO Single
                    case 0: ajan[k, l + ProductNum * 2 + 1] =
getVeloCityValueSingle(ajan, k, GlobalBestArray, localBest, l); break;
                    // Update VeloCity for PSOIN
                    case 1:
                       //Update VeleCity for PSOIN - Static
                       if (InertControl)
                          ajan[k, l + ProductNum * 2 + 1] =
getVeloCityValueINStatic(ajan, k, GlobalBestArray, localBest, l);
                       //Update VeloCity for PSOIN - Range
                       else
                          ajan[k, l + ProductNum * 2 + 1] =
getVeloCityValueINRange(ajan, k, GlobalBestArray, localBest, l, j );
                       break;
                 }

                 #endregion

                 #region --- Sınır Aşım Yöntemleri ---
                 //ajan = getPositionCityValue(k, l, ajan);
                 if (RastgeleDeger.NextDouble() < MR)
                 {
                    switch (PSOType)
                    {
                       case 0:
                          switch (SinirlandirmaMethodu)
                          {
                             case 0: ajan = getPositionCityValueGorunmez(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                             case 1: ajan = getPositionCityValueEmme(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                             case 2: ajan = getPositionCityValueYansitma(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                             case 3: ajan = getPositionCityValueSonumleme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
```

```
                        case 4: ajan = getPositionCityValueGorunmezYansitma(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 5: ajan = getPositionCityValueGorunmezSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 6: ajan = getPositionCityValueGorunurEmme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 7: ajan = getPositionCityValueGorunurSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 8: ajan = getPositionCityValueHibrit(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                    }
                    break;
                case 1:
                    switch (SinirlandirmaMethodu)
                    {
                        case 0: ajan = getPositionCityValueGorunmez(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 1: ajan = getPositionCityValueEmme(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 2: ajan = getPositionCityValueYansitma(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 3: ajan = getPositionCityValueSonumleme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 4: ajan = getPositionCityValueGorunmezYansitma(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 5: ajan = getPositionCityValueGorunmezSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 6: ajan = getPositionCityValueGorunurEmme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 7: ajan = getPositionCityValueGorunurSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 8: ajan = getPositionCityValueHibrit(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                    }
                    break;
                case 2:
                    switch (SinirlandirmaMethodu)
                    {
                        case 0: ajan = getPositionCityValueGorunmez(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 1: ajan = getPositionCityValueEmme(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
```

```
                        case 2: ajan = getPositionCityValueYansitma(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 3: ajan = getPositionCityValueSonumleme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 4: ajan = getPositionCityValueGorunmezYansitma(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 5: ajan = getPositionCityValueGorunmezSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 6: ajan = getPositionCityValueGorunurEmme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 7: ajan = getPositionCityValueGorunurSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 8: ajan = getPositionCityValueHibrit(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                    }
                    break;

                case 3:
                    switch (SinirlandirmaMethodu)
                    {
                        case 0: ajan = getPositionCityValueGorunmez(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 1: ajan = getPositionCityValueEmme(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 2: ajan = getPositionCityValueYansitma(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 3: ajan = getPositionCityValueSonumleme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 4: ajan = getPositionCityValueGorunmezYansitma(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 5: ajan = getPositionCityValueGorunmezSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 6: ajan = getPositionCityValueGorunurEmme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 7: ajan = getPositionCityValueGorunurSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 8: ajan = getPositionCityValueHibrit(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                    }
                    break;
                case 4:
                    switch (SinirlandirmaMethodu)
                    {
```

```
                        case 0: ajan = getPositionCityValueGorunmez(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 1: ajan = getPositionCityValueEmme(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 2: ajan = getPositionCityValueYansitma(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 3: ajan = getPositionCityValueSonumleme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 4: ajan = getPositionCityValueGorunmezYansitma(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 5: ajan = getPositionCityValueGorunmezSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 6: ajan = getPositionCityValueGorunurEmme(k, l, xMin,
xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                        case 7: ajan = getPositionCityValueGorunurSonumleme(k, l,
xMin, xMax, NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan);
break;
                        case 8: ajan = getPositionCityValueHibrit(k, l, xMin, xMax,
NyMinArr, NyMaxArr, MinYukseklikArr, MaxYukseklikArr, ajan); break;
                    }
                    break;

                }
            }

            #endregion
        }

        #region --- Local Part Updated ---

        //LocalBest (PBest) Update
        int x1 = 0; int[] y1 = new int[ProductNum]; int[] z1 = new
int[ProductNum];
        for (int j1 = 0; j1 < ProductNum; j1++)
        {
            y1[j1] = ajan[k, j1];
            z1[j1] = ajan[k, ProductNum + j1];
            if (j1 == ProductNum - 1)
                x1 = ajan[k, 2 * ProductNum];
        }

        uygunluksonuc[k] = getAmacFonksiyonu(x1, y1, z1);

        if ((PSOType == 1) || (DebControl))
        {
            #region --- Deb Kuralları ---
            //Uygunluk sonuc 0 ise parçacığın violation hesapla
```

```
                if (uygunluksonuc[k] == 0)
                {
                    for (int i = 0; i < ProductNum; i++)
                    {
                        if (ajan[k, i] * ajan[k, ProductNum + i] * ajan[k, 2 * ProductNum] *
2 < AdetArr[i])
                            ajanViolation[k] += AdetArr[i] - (ajan[k, i] * ajan[k, ProductNum
+ i] * ajan[k, 2 * ProductNum] * 2);
                    }
                    int tempy = 0;
                    for (int i = 0; i < ProductNum; i++)
                    {
                        tempy += ajan[k, i];
                    }
                    if (tempy > yMax)
                    {
                        ajanViolation[k] += (tempy - yMax) * ajan[k, 2 * ProductNum] *
2* MaxYukseklikArr [0];
                    }

                    for (int i = 0; i < ProductNum; i++)
                    {
                    if (ajan[k, ProductNum + i] > MaxYukseklikArr [i])
                        ajanViolation[k] += 2* (ajan[k, i] * (ajan[k, ProductNum + i] -
MaxYukseklikArr [i]) * ajan[k, 2 * ProductNum]);
                    }

                    if (ajanViolation[k] == 0)
                        ajanViolation[k] = 1;
                }

                //Lokal Best Array Violation Hesaplama
                localBestViolation[k] = 0;
                for (int i1 = 0; i1 < ProductNum; i1++)
                {
                    if (localBest[k, i1] * localBest[k, ProductNum + i1] * 2 * localBest[k,
ProductNum * 2] < AdetArr[i1])
                    {
                        int fark = AdetArr[i1] - (localBest[k, i1] * localBest[k, ProductNum
+ i1] * localBest[k, ProductNum * 2] * 2);
                        localBestViolation[k] += AdetArr[i1] - (localBest[k, i1] *
localBest[k, ProductNum + i1] * localBest[k, ProductNum * 2] * 2);
                    }

                    if (localBest[k, ProductNum + i1] > MaxYukseklikArr[i1])
                        localBestViolation[k] += 2 * (localBest[k, i1] * (localBest[k,
ProductNum + i1] - MaxYukseklikArr[i1]) * localBest[k, 2 * ProductNum]);

                }
```

```
                int templocaly = 0;
                for (int i = 0; i < ProductNum; i++)
                {
                    templocaly += localBest [k, i];
                }
                if (templocaly > yMax)
                {
                    localBestViolation[k] += (templocaly - yMax) * localBest [k, 2 *
ProductNum] * 2;
                }

                #region --- Normal Global Aktarma ---
                if ((uygunluksonuc[k] <
Convert.ToDouble(GlobalBestArray[ProductNum * 2 + 2])))
                {
                    //Timer - EOF -
                    //sw.Stop();
                    TimeSpan elapsedTime = sw.Elapsed;
                    elapsedMS = elapsedTime.TotalMilliseconds.ToString();

                    for (int g = 0; g < ProductNum; g++)
                    {
                        GlobalBestArray[g] = Convert.ToString(y1[g]);
                    }

                    for (int h = 0; h < ProductNum; h++)
                    {
                        GlobalBestArray[ProductNum + h] = Convert.ToString(z1[h]);
                    }

                    GlobalBestArray[ProductNum * 2] = Convert.ToString(x1);
                    GlobalBestArray[ProductNum * 2 + 1] = Convert.ToString(j);
                    GlobalBestArray[ProductNum * 2 + 2] =
Convert.ToString(uygunluksonuc[k]);
                }
                #endregion
            }
            #endregion

            #region --- Yazdır ---

            #endregion

        }
    }

    #endregion
    //Timer - EOF -
```

```
            sw.Stop();

            if (DebControl == false)
            {
                yazdirabi += "GlobalBest : ";
                for (int w = 0; w < ProductNum * 2 + 3; w++)
                {
                    if (w == ProductNum * 2 + 2)
                        yazdirabi += "  //Uygunluk :";
                    yazdirabi += GlobalBestArray[w];
                    yazdirabi += ",";

                    if (ProductNum * 2 + 2 == w)
                    {
                        yazdirabi += "  Süre: " + elapsedMS + "\n";
                    }
                }
            }
            if (DebControl == true)
            {
                yazdirabi += "GlobalBest : ";
                for (int w = 0; w < ProductNum * 2 + 3; w++)
                {
                    if (w == ProductNum * 2 + 2)
                        yazdirabi += "  //Uygunluk :";
                    yazdirabi += GlobalBestArray[w];
                    yazdirabi += ",";

                    if (ProductNum * 2 + 2 == w)
                    {
                        yazdirabi += "  Süre: " + elapsedMS + "\n";
                    }
                }
            }
        }



 private double getAmacFonksiyonu(int x11, int[] y11, int[] z11 )
        {

            double xDist = (((-2 * (2 + 4.2 * x11)) / (4.2 * x11)) + (((4.2 * x11 + 4) * (4.2 *
x11 + 4)) / ((4.2 * x11) * 4)));

            double yataysonuc = 0;
            double dikeysonuc = 0;
            double sonuc = 0;

            bool control = true;
```

```
for (int i = 0; i < ProductNum ; i++)
{
   if (y11[i] * z11[i] * 2* x11 >= AdetArr[i])
      control = true;
   else
   {
      control = false;
      break;
   }
}

   if (x11 > xMax)
      control = false;
   if (x11 < 1)
      control = false;
   //Yükseklik sınır kontrolü
   for (int prr = 0; prr < ProductNum; prr++)
   {
      if (z11[prr] > MaxYukseklikArr[prr])
         control = false;
   }

   for (int jas = 0; jas < ProductNum; jas++)
   {
      if (z11[jas] < 1)
         control = false;
   }

//Yatay Mesafe Kısıtı
int toplam = 0;
for (int j = 0; j < AdetArr.Length; j++)
   toplam += y11[j];
if (toplam > yMax)
   control = false;

   //Yatay Sonuç
   double[] yataysonucArr = new double[ProductNum - 2];
   for (int i = 0; i < ProductNum - 2; i++)
   {
      yataysonuc = 0;
      //double kontrol
      yataysonuc += 0.9 * (y11[i + 1] + y11[i + 2]);

      double yataysonucolasılık = 0.0;
      for (int i1 = i; i1 < ProductNum - 2; i1++)
      {
         yataysonucolasılık += Math.Round(OlasılıkArr[i1 + 2] / 2, 4);
      }
      yataysonuc *= yataysonucolasılık;
```

```
            yataysonucArr[i] = yataysonuc;
        }
        yataysonuc = 0;
        yataysonuc = 0.9 * ((y11[0] + y11[1]) * (1 - OlasılıkArr[0] / 2) - y11[1]/2.0);
        for (int j = 0; j < ProductNum - 2; j++)
            yataysonuc += yataysonucArr[j];
        yataysonuc += 2;

        //Dikey Sonuç
        for (int i = 0; i < ProductNum; i++)
        {
            dikeysonuc += PaletYukArr[i] * OlasılıkArr[i] * z11[i] / 2.0 ;
        }
       sonuc = Math.Floor(542.4 * (yataysonuc + dikeysonuc + xDist));

    return sonuc;
}

private int getVeloCityValueSingle(int[,] ajan, int part, string[] GlobalBestArray,
int[,] localBest, int boyut)
{
    int VeloCity;

    VeloCity = Convert.ToInt32(ajan[part, boyut + ProductNum * 2 + 1] +
SocialParam * RastgeleDeger.NextDouble() * (localBest[part, boyut] - ajan[part,
boyut]) + PersonalParam * RastgeleDeger.NextDouble() *
(Convert.ToInt32(GlobalBestArray[boyut]) - ajan[part, boyut]));
    if ((VeloCity < 0) && (VeloCity < -VMax))
        VeloCity = -VMax;
    else if ((VeloCity > 0) && (VeloCity > VMax))
        VeloCity = VMax;

    return VeloCity;
}

private int[,] getPositionCityValueGorunmez(int part, int boyut, int NxMin, int
NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
{
    int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];
    ajan[part, boyut] = newPosition;

    return ajan;
}

private int[,] getPositionCityValueEmme(int part, int boyut, int NxMin, int NxMax,
int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[] MaxYukseklikArr, int[,]
ajan)
{
```

```
        int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

        if (newPosition < 1)
        {
           newPosition = 1;
           ajan[part, ProductNum * 2 + 1 + boyut] = 0;
        }
        if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
        {
           newPosition = MaxYukseklikArr[boyut - ProductNum];
           ajan[part, ProductNum * 2 + 1 + boyut] = 0;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
           newPosition = NyMinArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] = 0;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
           newPosition = NyMaxArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] = 0;
        }

        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
        {
           newPosition = NxMax;
           ajan[part, ProductNum * 2 + 1 + boyut] = 0;
        }
        ajan[part, boyut] = newPosition;

       return ajan;

     }

     private int[,] getPositionCityValueYansitma(int part, int boyut, int NxMin, int
NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
      {
        int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

        if (newPosition < 1)
        {
           newPosition = 1;
           ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }
```

```
        if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
        {
           newPosition = MaxYukseklikArr[boyut - ProductNum];
           ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
           newPosition = NyMinArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
           newPosition = NyMaxArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
        {
           newPosition = NxMax;
           ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }
        ajan[part, boyut] = newPosition;

      return ajan;
    }

    private int[,] getPositionCityValueSonumleme(int part, int boyut, int NxMin, int
NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
    {
       int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

       if (newPosition < 1)
       {
          newPosition = 1;
          ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
       }
       if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
       {
```

```
            newPosition = MaxYukseklikArr[boyut - ProductNum];
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble() ));
         }

         if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
         {
            newPosition = NyMinArr[boyut];
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
         }

         if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
         {
            newPosition = NyMaxArr[boyut];
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
         }

         if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
         {
            newPosition = NxMax;
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
         }
         ajan[part, boyut] = newPosition;

        return ajan;
      }

      private int[,] getPositionCityValueGorunmezYansitma(int part, int boyut, int
NxMin, int NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
      {
         int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

         if (newPosition < 1)
         {
            ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
         }
         if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
         {
```

```
            ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }

        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] = ajan[part, ProductNum * 2 + 1 +
boyut] * -1;
        }
        ajan[part, boyut] = newPosition;

        return ajan;
    }

    private int[,] getPositionCityValueGorunmezSonumleme(int part, int boyut, int
NxMin, int NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
    {
        int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

        if (newPosition < 1)
        {
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
        }
        if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
```

```
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
        }

        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
        {
            ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] * -1.0 *
RastgeleDeger.NextDouble()));
        }
        ajan[part, boyut] = newPosition;

        return ajan;
    }

    private int[,] getPositionCityValueGorunurEmme(int part, int boyut, int NxMin, int
NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
    {
        int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

        if (newPosition < 1)
        {
            newPosition = 1;
        }
        if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
        {
            newPosition = MaxYukseklikArr[boyut - ProductNum];
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
            newPosition = NyMinArr[boyut];
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
            newPosition = NyMaxArr[boyut];
        }
```

```
        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
        {
           newPosition = NxMax;
        }
        ajan[part, boyut] = newPosition;

       return ajan;

    }

    private int[,] getPositionCityValueGorunurSonumleme(int part, int boyut, int
NxMin, int NxMax, int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[]
MaxYukseklikArr, int[,] ajan)
    {
        int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

        if (newPosition < 1)
        {
           newPosition = 1;
           ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
        }
        if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
        {
           newPosition = MaxYukseklikArr[boyut - ProductNum];
           ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble() ));
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
        {
           newPosition = NyMinArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
        }

        if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
        {
           newPosition = NyMaxArr[boyut];
           ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
        }

        if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
```

```
            {
                newPosition = NxMax;
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }
            ajan[part, boyut] = newPosition;

           return ajan;
        }

        private int[,] getPositionCityValueHibrit(int part, int boyut, int NxMin, int NxMax,
int[] NyMinArr, int[] NyMaxArr, int[] MinYukseklikArr, int[] MaxYukseklikArr, int[,]
ajan)
        {
            int newPosition = ajan[part, boyut] + ajan[part, boyut + ProductNum * 2 + 1];

            if (newPosition < 1)
            {
                newPosition = 1;
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }
            if ((boyut > ProductNum) && (boyut < 2 * ProductNum) && (ajan[part, boyut]
> MaxYukseklikArr[boyut - ProductNum]))
            {
                newPosition = MaxYukseklikArr[boyut - ProductNum];
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }

            if ((boyut < ProductNum) && (ajan[part, boyut] < NyMinArr[boyut]))
            {
                newPosition = NyMinArr[boyut];
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }

            if ((boyut < ProductNum) && (ajan[part, boyut] > NyMaxArr[boyut]))
            {
                newPosition = NyMaxArr[boyut];
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }
```

```
            if ((ajan[part, ProductNum * 2] > NxMax) && (boyut == ProductNum * 2))
            {
                newPosition = NxMax;
                ajan[part, ProductNum * 2 + 1 + boyut] =
Convert.ToInt32(Math.Round(ajan[part, ProductNum * 2 + 1 + boyut] *
RastgeleDeger.NextDouble()));
            }
            ajan[part, boyut] = newPosition;

            return ajan;
        }

        #endregion

        private int[,] getSharkPhase(int[,] ajan, int part, string[] GlobalBestArray, int
boyut)
        {
            for (int i = 0; i < boyut; i++)
            {
                ajan[part, i] = RastgeleDeger.Next(NyMinArr[i], NyMaxArr[i] + 1);
                ajan[part, i + ProductNum] = RastgeleDeger.Next(MinYukseklikArr[i],
MaxYukseklikArr[i] + 1);
            }

            ajan[part, ProductNum * 2] = RastgeleDeger.Next(xMin, xMax + 1);
            return ajan;

        }

    }
}
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı	Emre ÇAKMAK
Uyruğu	TC
Doğum tarihi, Yeri	01/03/1982, İzmir
Telefon	533 730 65 81
E-mail	emre.cakmakemre@gmail.com
Web adres	

Eğitim

Derece	Kurum/Anabilim Dalı/Programı	Yılı
Doktora	İ.Ü. Fen Bilimleri Enstitüsü/ /	2014
Yüksek Lisans	İ.T.Ü. Fen Bilimleri Enstitüsü/ Mühendislik Yönetimi	2007
Lisans	İ.Ü. Kimya Mühendisliği	2006
Lise	Isparta Anadolu Lisesi	2000

Makaleler / Bildiriler

Uluslararası Bildiriler

Çakmak, E. & Tanyas, M. 2008. "Determination of Safety Stock Levels for Subassemblies and Components of a Product". *Proceedings of the VI. International Logistics & Supply Chain Congress, Istanbul Bilgi University, Istanbul, 6-7 November, 201-209.*

Çakmak, E., Günay, N.S. & Tanyas, M. 2009. "Determination of Warehouse Storage Area Size with Pareto Analysis". *Proceedings of the VII. International Logistics & Supply Chain Congress, Yildiz Technical University, Istanbul, 5-6 November, 376-383.*

Çakmak, E., Tanyas, M. & Aba B. 2009. "Determination of Capital Investment to Build a Warehouse by Using a Simulation Program". *Proceedings of the VII. International Logistics & Supply Chain Congress, Yildiz Technical University, Istanbul, 5-6 November, 383-390.*

Uluslararası Bildiri Özetleri

Çakmak, E. Gulkac, H. & Tanyas, M. 2010, "Managing Subassemblies and Components for Safety Stock of Two Products". *Euro XXIV- 24th European Conference on Operational Research, Universidade de Lisboa, Portugal, 11-14 July.*

Günay, S.B., **Çakmak, E.** & Tanyas, M. 2010, "Modeling with Simulation for the Synchronization of Manufacturing and Warehousing Activities in a Paint Production Company". *Euro XXIV- 24th European Conference on Operational Research, Universidade de Lisboa, Lisbon, Portugal, 11-14 July.*

Ulusal Bildiriler

Çakmak, E. Günay, S.B. & Tanyas, M. 2009, "Akış Tipi Depo için Kapı Sayısı Gözönünde Bulundurarak Depo Eni, Boyu ve Raf Sayısı Belirleme". *9. Ulusal Üretim Araştırmaları Sempozyumu, Osmangazi Üniversitesi, Eskişehir, 15-17 Ekim, 679-686.*

Çakmak, E. Günay, S.B. & Tanyas, M. 2010, "Üretimde ve Depo Faaliyetleri Yönetiminde Simülasyon ile Modelleme". *10. Ulusal Üretim Araştırmaları Sempozyumu, Girne Amerikan Üniversitesi, KKTC, 16-18 Eylül.*