T.C.
HASAN KALYONCU ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
İŞLETME ANABİLİM DALI
DOKTORA PROGRAMI

SIRA-BAĞIMLI HAZIRLIK ZAMANLI GENEL MONTAJ HATTI DENGELEME PROBLEMLERİNİN ÇÖZÜMÜ İÇİN BİR HİBRİT ALGORİTMA ÖNERİSİ

DOKTORA TEZİ

HAZIRLAYAN
Şehmus ASLAN

GAZİANTEP -- 2020

T.C.
HASAN KALYONCU ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
İŞLETME ANABİLİM DALI
DOKTORA PROGRAMI

SIRA-BAĞIMLI HAZIRLIK ZAMANLI GENEL MONTAJ HATTI DENGELEME PROBLEMLERİNİN ÇÖZÜMÜ İÇİN BİR HİBRİT ALGORİTMA ÖNERİSİ

DOKTORA TEZİ

HAZIRLAYAN
Şehmus ASLAN

DANIŞMAN
Doç. Dr. Mehmet AYTEKİN

GAZİANTEP -- 2020

TEZ ETİK VE BİLDİRİM SAYFASI

Doktora Tezi olarak sunduğum "**Sıra-Bağımlı Hazırlık Zamanlı Genel Montaj Hattı Dengeleme Problemlerinin Çözümü İçin Bir Hibrit Algoritma Önerisi**" başlıklı çalışmanın tarafımca, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden oluştuğunu ve bunlara atıf yaparak yararlanmış olduğumu belirtir ve onurumla doğrularım. 16.06.2020

Şehmus ASLAN

ÖNSÖZ

Çalışmalarım sırasında yardımlarını her daim hissettiren ve yol gösteren değerli danışman hocam Doç. Dr. Mehmet AYTEKİN'e teşekkürü bir borç bilirim.

Doktora tez izleme komitesinde yer alarak, tezin daha özgün hale gelmesine katkıda bulundukları için değerli Hocalarım, Prof. Dr. Mazlum ÇELİK ve Dr. Öğr. Üyesi Zeynep ÖZGÜNER'e çok teşekkür ederim. Ayrıca doktora tez savunma komitesinde yer alarak, tezin son haline gelmesinde katkıda bulundukları için değerli Hocalarım, Doç. Dr. Süleyman METE ve Doç. Dr. Eren ÖZCEYLAN'a çok teşekkür ederim.

Doktora eğitimim boyunca bana destek veren TÜBİTAK-2211 Yurt İçi Lisansüstü Burs Programı'na teşekkürlerimi arz ederim.

Çalışmalarım sırasında sabır, sevgi ve anlayışla beni dinleyen ve destekleyen sevgili eşim Nevin ASLAN'a ve biricik kızım Mira'ya sonsuz teşekkürlerimi sunarım.

Gaziantep, 2020 Şehmus ASLAN

ÖZET

Basit Montaj Hattı Dengeleme Problemleri (BMHDP) ile ilgili literatürde bir çok çalışma yapılmıştır. Ancak BMHDP'de bulunan kısıtlardan dolayı yapılan akademik çalışmalar ve endüstrideki uygulamalar arasında büyük bir boşluk bulunmaktaydı. Bu boşluğun kapatılması için Genel Montaj Hattı Dengeleme Problemleri (GMHDP) adı altında daha çok endüstrinin pratik sorunlarını çözmeye yönelik çalışmalar başlamıştır. Otomotiv ve elektronik sektöründe sıkça rastlanan sıra-bağımlı hazırlık zamanları, daha önce yapılan Montaj Hattı Dengeleme (MHD) çalışmalarında ele alınmamıştır. Daha önceleri, MHD çalışmalarında hazırlık zamanları, istasyon zamanlarına eklenerek problemler çözülmekteydi. Bu yaklaşım sorunu çözmede yetersiz kaldığı için sıra-bağımlı hazırlık zamanlı GMHDP çalışmaları ortaya çıkmıştır. Sıra-bağımlı hazırlık zamanlı GMHDP, NP-zor yapıda ve çok karmaşık problemler olduğundan lineer programlama ve dal-sınır algoritması gibi belirli (deterministik) yöntemler, makul zamanlarda çözüm üretememektedir. Bu çalışmada problemlerin çözümünde metasezgisel bir yöntem olan yeni bir Diferansiyel Gelişim Algoritması (DGA) ve Diferansiyel Gelişim Algoritması (DGA) - Parçacık Sürü Optimizasyonu (PSO) metasezgisellerinden oluşan bir Hibrit Algoritma (HA) geliştirilmiştir. Geliştirilen DGA ve HA'nın performansı literatürdeki test problemleri üzerinde denenmiş ve bu algoritmaların literatürde daha önce geliştirilmiş sezgisel yöntemlerden daha iyi sonuçlar verdiği tespit edilmiştir.

Anahtar kelimeler: Montaj Hattı Dengeleme, Sıra-Bağımlı Hazırlık Zamanlı, Diferansiyel Gelişim Algoritması, Parçacık Sürü Optimizasyonu, Hibrit Algoritma

ABSTRACT

Many studies have been conducted in the literature on Simple Assembly Line Balancing Problems (SALBP). However, there was a huge gap between academic studies and industry practices due to the limitations found in SALBP. In order to close this gap, studies under the field of the General Assembly Line Balancing Problems (GALBP) have been started to solve the practical problems of the industry. The sequence-dependent setup times, which are common in the automotive and electronics sectors, have not been addressed in previous Assembly Line Balancing (ALB) studies. Setup times were added to station times to solve the problems in previous ALB studies. As this approach is insufficient to solve the problem, the sequence-dependent setup times GALBP studies have emerged. Because the sequence-dependent setup times GALBP is NP-hard and very complex problems, they can not be solved in reasonable time by deterministic methods such as linear programming and branch and bound algorithm. In this study, a new Differential Evolution Algorithm (DEA) and Differential Evolution Algorithm (DEA) - Particle Swarm Optimization (PSO) Hybrid Algorithm (HA) which are metaheuristics, was developed to solve these problems. The performance of the developed DGA and HA was tested on the test problems in the literature and they gave better results than the previously developed heuristic methods in the literature.

Keywords: Assembly Line Balancing, Sequence-Dependent Setup Times, Differential Evolution Algorithm, Particle Swarm Optimization, Hybrid Algorithm

İÇİNDEKİLER

TABLOLAR LİSTESİ

Sayfa No

ŞEKİLLER LİSTESİ

Sayfa No

KISALTMALAR

MHD : Montaj Hattı Dengeleme

MHDP : Montaj Hattı Dengeleme Problemleri

BMHDP : Basit Montaj Hattı Dengeleme Problemleri

GMHDP : Genel Montaj Hattı Dengeleme Problemleri

DGA : Diferansiyel Gelişim Algoritması

GDGA : Geliştirilmiş Diferansiyel Gelişim Algoritması

PSO : Parçacık Sürü Optimizasyonu

HA : Hibrit Algoritma

TMD : Tek Modelli Deterministik

TMS : Tek Modelli Stokastik

ÇKMD : Çoklu/Karışık Modelli Deterministik

ÇKMS : Çoklu/Karışık Modelli Stokastik

KMMHDP : Karışık Modelli Montaj Hattı Dengeleme Problemi

KMSP : Karışık Modelli Sıralama Problemi

UMHDP : U-tipi Montaj Hattı Dengeleme Problemi

SİMGELER

$\tau_{i,j}$: i ve j görevleri arasındaki ileri hazırlık zamanı

$\mu_{p,i}$: p ve i görevleri arasındaki geri hazırlık zamanı

t_i : i görevinin işlem zamanı

c : Çevrim zamanı

X_i : i indisli vektör

G : Nesil sayısı

S : Başlangıç popülasyonu

N_p : Populasyon sayısı

v_i : i indisli mutant vektör

u_i : i indisli aday vektör

$f(X_i)$: X_i vektörü uygunluk değeri

x_i : *i* parçacığının pozisyonu

v_i : *i* parçacığının hızı

P_i : i indisli parçacık

c_1, c_2 : ivme katsayıları

ω : Atalet ağırlığı

χ : Daralma katsayısı

G_{best} : Global en iyi

P_{best} : Yerel en iyi

Pb_i : i parçacığı yerel en iyisi

S : Yer değiştirme vektörü

S_n : Başlangıç sürüsü

CR : Çaprazlama oranı

BİRİNCİ BÖLÜM

GİRİŞ

Montaj, değiştirilebilir parçaların daha kompleks yapıda nihai ürün veya ara montajların oluşturulması için kesin tanımlı bir şekilde birbirine bağlandığı endüstriyel olarak üretilen malların nihai üretim aşamasını oluşturmaktadır. Bu, ara montaj ürünlerinin farklı yerlerde üretilip sonrasında tam ürün ortaya çıkarılması için başka bir üretim bandında monte edilebilmesine olanak sağlamaktadır (Nof vd., 1997). Montaj hatları, elektrikli aletlerden (kahve makinası, ekmek kızartma makinası, kurutucu, bulaşık makinası, buzdolabı), elektronik aletlere (radyo, kamera, televizyon, yazıcı, bilgisayar) ve otomobillere (araba, kamyon) kadar birçok ürünün seri üretiminde sıklıkla kullanılan sistemlerdir (Uğurdağ vd., 1997; Amen, 2001; Cheldelin ve Ishii, 2004). Giderek çeşitlenen müşteri ihtiyaçlarını karşılayabilmek adına ana ürün çeşitlerine ek parçalar eklenerek, parça değiştirilerek ve/veya parça çıkartılarak, ürün aileleri çeşitlendirilmektedir (Zülch vd. 1997).

Müşteri taleplerinin hızla değişmesi ve piyasa belirsizliği nedeniyle üreticiler rekabet güçlerini koruyabilmek ve artırabilmek için teknolojik ve kurumsal düzeyde yeni çözümler aramak zorundadırlar. Ürün çeşitliliğine yönelik müşteri taleplerini karşılayabilmek için montaj hatları, standart malların yüksek hacimlerde üretiminden daha çok isteğe göre uyarlanmış ürünlerin düşük hacimde üretimine doğru kaymıştır (Becker ve Scholl, 2006; Scholl ve Becker 2006; Boysen vd., 2007). Müşteri taleplerinin seviye ve yapısının sürekli değişmesi ve yeni üretim teknolojilerinin ortaya çıkması, montaj hatlarının düzenli olarak yeniden tasarlanması gerektirmektedir.

Montaj hatları, ölçek ekonomilerinde düşük birim fiyatlı üretim imkânı sağlamaktadır. Bu nedenle de bir ürünün üretilmesi için gereken işin, hat boyunca çalışma istasyonlarına verimli bir şekilde yayılması büyük önem arz etmektedir. Kurulumun gerektirdiği yüksek sermaye ve sonrasında üretim verimliliği ve maliyetlerine olan etkisi göz önünde bulundurulduğunda, bir montaj hattının dizaynının belirlenmesi, uygulayıcılar için çok önemlidir (Boysen vd., 2007).

Montaj Hattı Dengeleme (MHD), montaj işleminin yapılabilmesi için gerekli işlerin, bu işlerin süreleri ve işler arasındaki öncelik ilişkileri verildiğinde, bir performans ölçütü en iyilenecek şekilde, sıralı iş istasyonlarına atanması şeklinde tanımlanabilir (Ağpak vd., 2002). MHD'nin ana amaçlardan biri, her iş istasyonuna eşit miktarda iş dağıtımı yapabilmektir. Başka bir deyişle, toplam iş yükünü iş istasyonları arasında mümkün olduğu kadar eşit bir şekilde bölebilmektir. Bir Basit Montaj Hattı Dengeleme Problemi'nde (BMHDP) temel olarak aşağıdaki kısıtlar kullanılmaktadır:

- Bütün işler iş istasyonlarına atanmalıdır,
- Bir görev sadece bir iş istasyonuna atanabilir,
- İş istasyonuna atanan görevlerin toplam süreleri çevrim süresini geçemez,
- Görevler arasındaki teknolojik ilişkilerden kaynaklanan öncül-ardıl ilişkileri bozulamaz.

Amaç fonksiyonu göz önüne alındığında Montaj Hattı Dengeleme Problemleri (MHDP), 4 farklı gruba ayrılabilir (Becker ve Scholl, 2006):

1- MHDP tip-1: Çevrim zamanı verilir, istasyon sayısı minimize edilmeye çalışılır.
2- MHDP tip-2: İstasyon sayısı verilir, çevrim zamanı minimize edilmeye çalışılır.
3- MHDP tip-E: Hat verimliliğini maksimize etmek için, istasyon sayısı ve çevrim zamanı aynı anda minimize edilmeye çalışılır.
4- MHDP tip-F: Verilen istasyon sayısı ve çevrim zamanı için problemin uygun (feasible) olup olmadığına bakılır. Uygun çözümler bulunmaya çalışılır.

BMHDP, montaj hattı dengeleme problemlerinin en basit ve orijinal halidir. Bu problem tipine paralel istasyonlar, bölgeleme kısıtları, kaynak kısıtları gibi bazı kısıtlamalar veya faktörler eklenirse problem Genel Montaj Hattı Dengeleme Problemleri (GMHDP) olarak adlandırılır. Literatürde GMHDP'nin U-tipi, paralel istasyonlar, karışık modelli vb. çeşitleri mevcuttur.

Bu çalışmada, GMHDP grubunda yer alan sıra-bağımlı hazırlık zamanlı montaj hattı dengeleme tip-1 problemlerinin çözümü için bir Diferansiyel Gelişim Algoritması (DGA) ve Diferansiyel Gelişim Algoritması (DGA) - Parçacık Sürü Optimizasyonu (PSO)

metasezgisellerinin hibritize edilmesiyle oluşan bir Hibrit Algoritma (HA) geliştirilmiştir. Geliştirilen algoritmalar literatürde daha önce bu tip problemlerin çözümü için geliştirilen sezgisellerle karşılaştırılmıştır.

Bu çalışma beş bölümden oluşmaktadır. Çalışmanın ilk bölümünde; araştırmanın amacı, araştırmanın önemi, araştırmanın kapsamı ve sınırları, araştırmanın yöntemi ele alınmıştır.

İkinci bölümde, çalışmanın kavramsal çerçeve kısmında, öncelikle tez çalışmasında ele alınan montaj hatlarının tanımları ve montaj hatları ile ilgili temel kavramlar verilmiştir. Daha sonra montaj hatlarının sınıflandırılması hakkında bilgi verilmiştir. Ardından, MHDP'nin tanımı yapılmış olup ve onunla ilgili literatürde yapılmış sınıflandırmalar açıklanmıştır. Sonrasında MHDP'nin çözümünde kullanılan çözüm yöntemleri ele alınmıştır. Montaj hatları ile ilgili son bölümde ise bu çalışmanın esas problemi olan sıra-bağımlı hazırlık zamanlı GMHDP sayısal örneklerle detaylı açıklanmıştır. MHD ile ilgili detaylı bilgiler verildikten sonra yöntem kısmında kullanılan diferansiyel gelişim algoritması (DGA) ile ilgili temel bilgiler verilmiştir. Daha sonra yöntem kısmında kullanılan diğer bir metasezgisel olan parçacık sürü optimizasyonu (PSO) açıklanmıştır. Kavramsal çerçeve kısmının son bölümünde ise DGA, PSO ve sıra-bağımlı hazırlık zamanlı GMHDP ile ilgili literatür taraması detaylı bir şekilde verilmiştir.

Üçüncü bölümde, çalışmanın yöntem kısmında, öncelikle sıra-bağımlı hazırlık zamanlı GMHDP için geliştirilmiş DGA (GDGA) metasezgisel yöntemi detaylı açıklanmıştır. Daha sonra GDGA'nın daha iyi anlaşılabilmesi için bir sayısal örnek verilmiştir. Ardından sıra-bağımlı hazırlık zamanlı GMHDP için geliştirilmiş diğer bir yöntem olan GDGA-PSO metasezisellerinin hibritize edilmesinden oluşan hibrit algortima (HA) metasezgiseli açıklanmıştır. HA'nın daha iyi anlaşılabilmesi için de bir sayısal örnek verilmiştir.

Dördüncü bölümde, çalışmanın bulgular ve değerlendirme kısmında, GDGA ve HA metasezgiselleri literatürde daha önce geliştirilmiş olan sezgisellerle kıyaslanmıştır. Bu karşılaştırma, literatürde yer alan test problemleri üzerinden yapılmıştır.

Beşinci bölümde, çalışmanın sonuç ve öneriler kısmında, çalışmanın sonuçları değerlendirilmiş olup, çalışmada kullanılan yöntemlerin başarılı olmasındaki etkenler üzerinde durulmuştur. Son olarak da bundan sonra yapılacak çalışmalar için araştırmacılara önerilerde bulunulmuştur.

1.1. Araştırmanın Amacı

MHDP literatürüne bakıldığında BMHDP ile ilgili çok sayıda çalışma yapıldığı görülmektedir. BMHDP'de kullanılan kısıtlar gerçek hayattaki montaj hatlarını yansıtmaktan uzak kalmasına sebep olmaktadır. Bu nedenle de akademik çalışmalar ile pratikteki uygulamalar arasındaki boşluğun doldurulması için bu çalışmalar da, daha gerçekçi yaklaşımlar sunan GMHDP altında yapılmaya başlandı (Becker ve Scholl, 2006). Bu doğrultuda Andres vd. (2008) literatürdeki bir boşluğu doldurmak için sıra-bağımlı hazırlık zamanlı GMHDP kavramını ortaya atmışlardır. Bu çalışmada sıra-bağımlı hazırlık zamanlı tip-1 GMHDP, Diferansiyel Gelişim Algoritması (DGA) ve Diferansiyel Gelişim Algoritması (DGA)- Parçacık Sürü Optimizasyonu (PSO) metasezgisellerinden oluşan hibrit bir algoritma (HA) ile çözülmüştür. Sıra-bağımlı hazırlık zamanlı tip-1 GMHDP daha önce sezgisel yöntemlerle (Andres vd., 2008; Martino ve Pastor, 2010; Scholl vd., 2013) çözüldüğü için, literatürde ilk defa bu çalışmayla hibrit metasezgisel yöntemler ile çözülmeye çalışılmıştır. Bulunan sonuçlar daha önceki çalışmalarda çıkan sonuçlarla karşılaştırılmıştır.

1.2. Araştırmanın Önemi

Bir çok endüstriyel montaj hattında hazırlık zamanları olmasına rağmen, işlem (proses) zamanına göre az oranda olduğu için gözardı edilmektedir. Ancak; otomotiv, elektronik ve diğer bazı robotik üretim hatlarında işlem zamanları kısa olduğu için hazırlık zamanlarının işlem zamanına göre oranı yüksek olabilmektedir. Dolayısıyla daha verimli bir montaj hattı dengelemesi için bu hazırlık zamanlarının göz önünde bulundurulması gerekmektedir. Gerçek hayatta sıkça karşılaşılan bu durum montaj hattı dengeleme literatüründe çok fazla yer edinememiştir. Literatürde ilk defa bahsedilen bu kavram aslında pratikte sık rastlanan bir durumdur. Dolayısıyla literatürdeki bu açığı kapatmak için bu konu üzerinde çalışılmıştır.

1.3. Araştırmanın Kapsamı ve Sınırları

Bu çalışmada MHDP'nin alt grubunda bulunan GMHDP'nin sıra-bağımlı hazırlık zamanlı olan modeli çözülmeye çalışılmıştır. Bu çalışmada ele alınan GMHDP düz, tek modelli ve deterministik olarak varsayılmıştır. Ayrıca amaç fonksiyonu göz önüne alındığında bu problem tip-1 GMHDP kategorisine girmektedir.

1.4. Araştırmanın Yöntemi

Bu çalışmada sıra-bağımlı hazırlık zamanlı tip-1 GMHDP, bir metasezgisel olan Diferansiyel Gelişim Algoritması (DGA) ve Diferansiyel Gelişim Algoritması (DGA)-Parçacık Sürü Optimizasyonu (PSO) metasezgisellerinden oluşan hibrit bir algoritma (HA) ile çözülmüştür. Algoritmalar MATLAB programlama dili kullanılarak kodlanmış, probleme ait veriler ve temel parametreler metin dosyasından okutulmuştur. Programın çıktıları metin formatında dosyalara yazdırılmış daha sonra üzerinde düzenlemeler yapılarak Microsoft Excel ile işlenebilecek hale getirilmiştir. Elde edilen veriler Microsoft Excel ile karşılaştırmalı tablolara dönüştürülmüştür.

İKİNCİ BÖLÜM

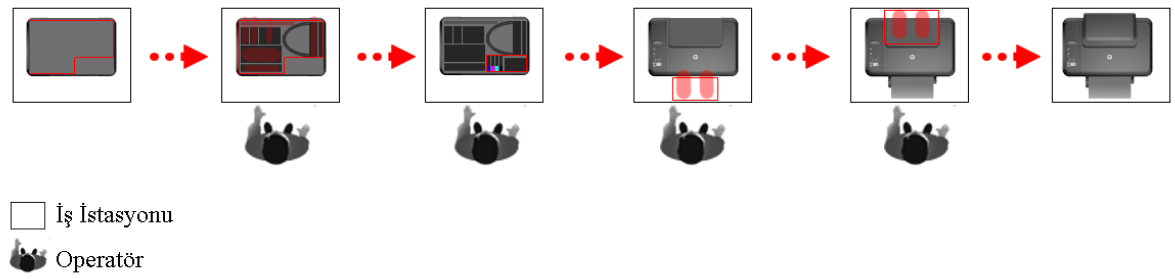
KAVRAMSAL ÇERÇEVE

Çalışmanın bu bölümünde, montaj hattı dengeleme problemleri, diferansiyel gelişim algoritması ve parçacık sürü optimizasyonu ile ilgili önemli olduğu düşünülen bazı temel kavramlar açıklanmıştır.

2.1. Montaj Hattı Dengeleme

2.1.1. Montaj Hatları

Montaj hattı, nihai ürünlerin istasyon olarak adlandırılan seri hizalı üretim birimlerinden geçerken monte edildiği, akış yönlü bir üretim sistemidir. Üretim, üzerinde çalışılacak parçanın hat üzerine bırakılması ile başlar. Bu parça, konveyör bandı gibi bir taşıma sistemi vasıtasıyla istasyonlar arasında hareket eder. Her bir istasyonda parça üzerinde görev olarak adlandırılan belirli sayıda operasyon gerçekleştirilir. Her bir görevin gerçekleştirilmesi için belirli bir süre ve bir takım araçlar gerekir. Teknolojik kısıtlamalar veya ürünün fiziksel yapısı nedeniyle bazı görevler başlamadan önce bazılarının bitmesi gerekmektedir. Bu kısıtlamalar, her görevin bir nokta ve her öncelik ilişkisinin ise bir yay ile temsil edildiği öncelik grafiğinde gösterilir. Basit bir montaj hattının yerleşim planı aşağıda Şekil 1'de verilmiştir.



Şekil 1. Montaj Hattı Yerleşimi

Montaj hattı üretimi, Sanayi Devrimi'nin son döneminde geliştirilmiş önemli bir buluş olmakla birlikte bir ürün veya malın, otomatik bir hat boyunca çalışan işçiler tarafından üretilmesi ana fikrine dayanır. Ürün hat boyunca hareket ederken her bir işçi, ürünü oluşturmak için farklı bir görev gerçekleştirir. Montaj hattının ana fikri, her bir işçinin yalnızca bir veya bir kaç görevden sorumlu olması ve farklı ürünler hat boyunca hareket ettikçe aynı görevi sürekli tekrar etmesidir. Montaj hatları geliştirilmeden önce ürünler genelde tek bir zanaatkar veya küçük bir zanaatkar ekibi tarafından elde üretilmekteydi. Zanaatkarın birçok görevi gerçekleştirmesi gerektiği için bu yöntem yavaş ve verimsizdi. Bu nedenle de Sanayi Devrimi döneminde montaj hatlarının geliştirilmesi sayesinde üretim hızlanmış ve kolaylaşmıştır.

Tarihçiler, montaj hattı yöntemi örneklerine ilk defa 1700'lerin sonunda ve 1800'lerin başında İngiltere'de rastlamışlardır. İngiltere'nin Salford bölgesinde bulunan Bridgewater dökümhane'sinde 1830'larda takım tezgâhı ve lokomotif üretiminde montaj hattı yöntemlerinden yararlanılmaktaydı. Fakat, montaj hattı kullanımının en bilinen örneği, 1913'te Ford Motor Company'de görülmüştür. Otomobilin icat edilmesi, İkinci Sanayi Devrimi'nin en önemli icatlarından biri olmuştur. Tarihte ilk otomobili üreten kişi, Henry Ford değildir. Fakat Henry Ford'un önemi, daha geniş bir kitleye yönelik ilk uygun maliyetli otomobili montaj hatlarını kullanarak seri üretme geçirmiş olmasıdır.

2.1.2. Montaj Hatları ile ilgili Temel Kavramlar

MHDP'de sık kullanılan kavramlar aşağıda verilmiştir (Benzer, 2005):

Görev, bir montaj prosesindeki toplam iş içeriğinin bölünemeyen en küçük parçasıdır. Montaj hattının verimsiz olmaması için görevlerin çok küçük de olmaması gerekmektedir. Bir problemdeki toplam görev sayısı, *N* ile ifade edilmektedir.

Görev zamanı, bir görevin gerçekleştirilmesi için gereken süredir. *i* görevinin süresi, t_i ile ifade edilmektedir. Montaj hattı dengeleme problemlerinin çözülmesi için geliştirilmiş tekniklerin büyük bir kısmı, bu değeri belirli (deterministik) bir sabit olarak kabul etmektedir. Diğer yandan, endüstride kullanılan montaj hatlarının çoğunluğunda işçilerin değişken (stokastik) sürelerle çalıştığı görülmektedir. Görevlerin karmaşık olduğu ve yüksek düzeyde

beceri ve konsantrasyon gerektirdiği zamanlarda görev sürelerin standart sapmaları da daha yüksek olmaktadır. Değişken görev sürelerinin μ_i ve σ_i^2 ifadeleri, i görevinin sırasıyla ortalama ve varyans görev sürelerini ifade etmektedir.

İstasyon, hatta yapılması öngörülen toplam iş miktarının bir kısmının yerine getirildiği yerdir. Her bir istasyonda ürünlerin üretilmesi için gereken *N* operasyonun bir alt kümesi gerçekleştirilir ve *k* ile gösterilir. Toplam istasyon sayısı ise *K* ile ifade edilir.

İstasyon zamanı, montaj hattı üzerindeki belirli bir istasyona verilen görevlerin toplam süresidir. *k* istasyonunun istasyon zamanı, S_k ile ifade edilir.

Çevrim zamanı, her bir operatörün bir çevrimde çalışabileceği maksimum zamanı göstermektedir. C ile ifade edilir ve belirli bir hızda akan konveyör üzerinde çalışan tüm operatörler için sabit olduğu kabul edilir. Standart bir akış temposuyla bir üretim hattında tamamlanıp çıkan, ardı ardına iki ürün arasındaki fark olarak tanımlanır.

$$\underset{i=1,\ldots,N}{\text{Max}}\ t_i \leq \underset{j=1,\ldots,K}{\text{Max}}\ S_j \leq C \qquad (2.1)$$

İstasyon boş zamanı, çevrim zamanı ile istasyon zamanı arasındaki farkı ifade eder. Tüm istasyonların boş zamanlarının toplanması ile bir hattın verimli olup olmadığı ölçülebilir. Bu toplama, **toplam boş zaman** adı verilir. İlgili bir diğer verimlilik ölçüsü de toplam boş zamanın, ürünün hattın başından sonuna kadar gelmesi sırasında geçen toplam zamana oranını veren **denge gecikmesidir (d)**. Şu şekilde ifade edilebilir:

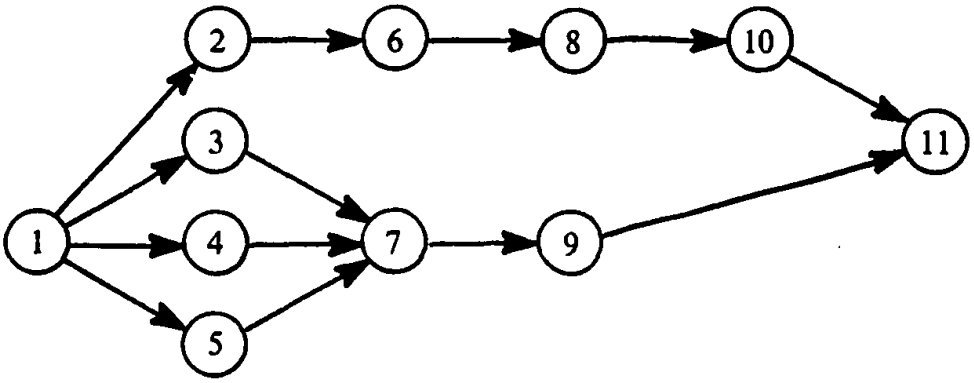
$$d = \frac{100[K.C - \sum_{i=1}^{N} t_i]}{K.C} \qquad (2.2)$$

Mükemmel dengede olan bir hatta denge gecikmesi sıfırdır. Bir diğer verimlilik ölçüsü de **düzgünlük endeksidir (DE)**. Şu şekilde ifade edilebilir:

$$DE = \sqrt{\sum_{j=1}^{K}(S_{max} - S_j)^2} \qquad (2.3)$$

Öncelik diyagramı, ürün montajı için hangi görevlerin hangi sırada gerçekleştirileceğinin grafik şeklinde açıklamasıdır. 11 görevli bir montaj hattının öncelik

diyagramı, Şekil 2'de örnek olarak verilmiştir. Dairelerin içindeki sayılar, görevleri ifade etmektedir. Diyagram, montajın soldan sağa gittiğini gösterir şekilde çizilmiştir.



Şekil 2. 11 Görevli Bir Hattın Öncelik Diyagramı

Öncelik matrisi, öncelik ilişkilerini gösteren bir teknik öncelik matrisidir. Öncelik diyagramındaki *i* görevini *j* görevi takip ediyorsa matriste *i.* satır *j.* sütun 1, aksi halde 0 değerini alır. Şekil 2'de gösterilen örneğin öncelik matrisi, Tablo 1'de verilmiştir.

Tablo 1. 11 Görevli Hattın Öncelik Matrisi

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1		1	1	1	1	1	1	1	1	1	1
2			0	0	0	1	0	1	0	1	1
3				0	0	0	1	0	1	0	1
4					0	0	1	0	1	0	1
5						0	1	0	1	0	1
6							0	1	0	1	1
7								0	1	0	1
8									0	1	1
9										0	1
10											1

Bir montaj işleminin öncelik yapısı, **Esneklik Oranı (E-oranı)** ile ifade edilir. Esneklik oranı, *N* görevli bir montaj işleminin ortaya çıkabilecek uygulanabilir sıra sayısını

ölçen bir orandır. Esneklik, problem çözümünü kolaylaştıran ya da zorlaştıran bir faktördür. Şu şekilde ifade edilir:

$$\text{E-oranı} = \frac{2Y}{N(N-1)} \tag{2.4}$$

Eşitlikteki Y değişkeni öncelik matrisindeki sıfır sayısını, N ise görev sayısını ifade etmektedir.

Sıralama gücü (SG), öncül ilişkilerin sayısının olası öncül ilişkileri sayısına oranı olarak tanımlanır ve şu şekilde ifade edilir:

$$SG = \frac{\text{Öncül ilişkileri sayısı}}{\frac{N(N-1)}{2}} \tag{2.5}$$

Görevlerin paralellenmesi, görevlerin birden fazla istasyonda gerçekleştirilmesinin sağlanmasıdır.

İstasyonların paralellenmesi, hat üzerinde belli noktalarda eşdeğer çalışma istasyonlarına imkân verilmesidir. Paralelleme yöntemleri, hat tasarımını farklı şekillerde modifiye etmek amacıyla kullanılmaktadır. Böylece hattın denge verimliliği geliştirilebilir ve en uzun görev süresinin çevrim zamanı üzerinde yarattığı baskı gevşetilebilir.

İstasyon tıkanması (Blocking), istasyondaki yetersiz bekleme yeri kapasitesi veya istasyondaki görevlerin zamanında tamamlanamaması nedeniyle görevin istasyonda beklemesidir.

İstasyonun açlığı (Starving), İstasyon (ilk istasyon hariç) uygun fakat işlem için herhangi bir görev olmaması durumudur.

Bölgeleme kısıtlamaları, bazı görevlerin aynı istasyonda gerçekleştirilmesinin imkânsız olması veya bazı görevlerin yalnızca aynı istasyonda gerçekleştirilebilmesi gibi montaj hattı dengeleme problemlerindeki ek kısıtlamalardır.

2.1.3. Montaj Hatlarının Sınıflandırılması

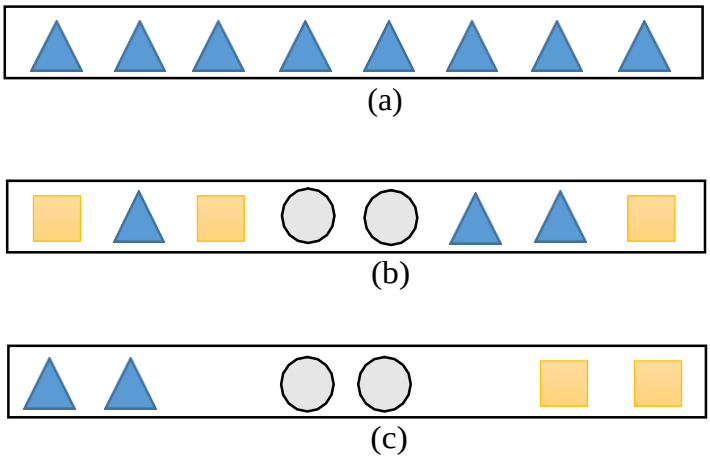
Bir montaj hattı sistemi seçerken uygulama maliyeti, üretim hacmi, yer imkânı, ürün çeşitliliği ve mevcut montaj teknolojisi gibi çeşitli ve birbiriyle çatışabilen kriterleri göz önünde bulundurmak gerekmektedir. Montaj hatlarının dengelenmesi için genel bir çözüm yöntemi olmadığı için montaj hattının özelliklerinin belirlenmesi, MHD probleminin çözümünde kritik öneme sahiptir. Bu bölümde üretim modelleri, hat düzeni, otomasyon seviyesi ve operasyon modu gibi yaygın şekilde kullanılan kategorizasyon şemalarına göre alternatif montaj hattı uygulamaları gösterilmektedir.

2.1.3.1. Ürün (Model) Çeşidi veya Sayısına Göre Montaj Hatları

Tek modelli hatlar: Tek bir ürün türünün yüksek hacimli montajı için kullanılan geleneksel hatlar, bu montaj sistemi sınıfında yer almaktadır (Şekil 3-a). Tek modelli hatlar, büyük ölçekle düşük birim maliyetli üretim için uygundur. Tek modelli hatlar

Karışık modelli hatlar: Renk, boyut, kullanılan malzeme veya ekipman gibi özelleştirilebilir özelliklere göre farklılık gösteren aynı baz ürünün çeşitleri (modelleri), aynı hat üzerinde farklı sıralarda monte edilebilir (Boysen vd., 2008). Karışık modelli montaj hatları, her bir ürün çeşidi için ayrı bir montaj hattı kurulumunun ekonomik olmadığı durumlarda yatırım maliyetlerinin amorti edilmesi için farklı modellerin tek bir montaj hattını kullanması durumunda kullanılır (Becker ve Scholl, 2006). Hazırlık zamanlarının ve maliyetlerin karışık modelli üretim için uygun olan montaj hatlarında tüm ürünler homojen bir sırayla üretime alınırlar (Şekil 3-b).

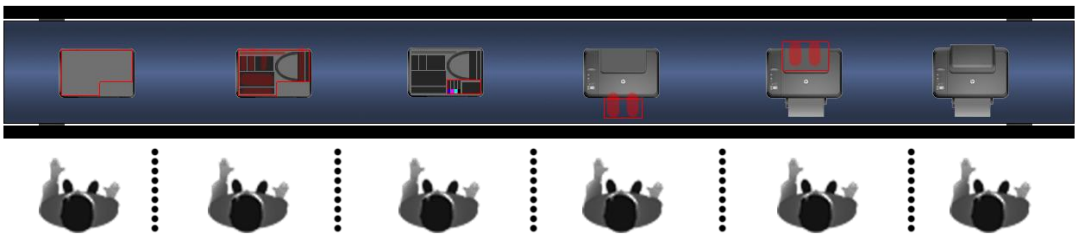
Çok modelli hatlar: Bu hat sistemleri, ürün çeşitleri arasındaki benzerlik az olduğu için birbirinden farklı ürünlerin homojen bir sırada montaj işleminden geçirilmesi uygun olmayabilir. Fakat her bir ürün çeşidinin montajından önce bir hazırlık işlemi yapıldıktan sonra montaj işlemine geçilir (Boysen vd., 2008). Bunun sonucu olarak da üretim, partiler halinde gerçekleşir (Şekil 3-c). Gereksiz kurulum süresi ve maliyetlerinden kaçınmak için her partide yalnızca tek bir çeşit ürün bulunur (Becker ve Scholl, 2006).



Şekil 3. Tek modelli (a), Karışık modelli (b), Çok modelli (c) montaj hatları

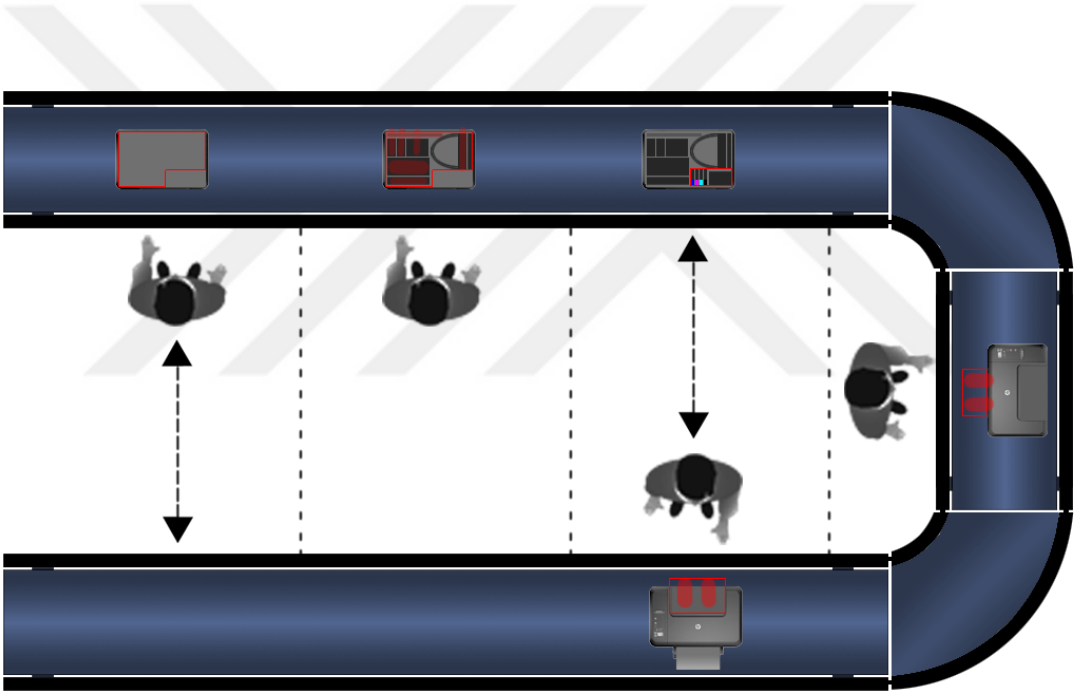
2.1.3.2. Hat Şekline ya da Yerleşime Göre Montaj Hatları

Seri montaj hatları: Geleneksel montaj hatlarında çalışma istasyonları fabrika zemini üzerinde tek bir hat üzerinde sıralanır (Şekil 4). Ürünler, birbiri ardına sıralanan çalışma istasyonları arasında konveyör bandı gibi bir malzeme taşıma mekanizması ile taşınır (Boysen, 2007).



Şekil 4. Düz Montaj Hattı Yapısı

U şekilli montaj hatları: U şekilli konfigürasyonda bir ürünün montajı için gereken görevler, fabrika zemini üzerinde kavisli bir hat üzerinde gerçekleştirilir (Şekil 5). Bu model, modern montaj hatlarında daha çok kullanılmakta olup hattın iki ucu, dar bir "U" harfi oluşturacak şekilde birbirine yakındır (Scholl ve Klein, 1999). Operatörler, hattın iki bölümü arasında hareket edebilecek şekilde U'nun içerisinde yer alırken bir operatör de hattın hem baş hem de son kısmını kontrol eder (Miltenburg, 2001). Bunun sonucu olarak, çalışma istasyonlarına hattın her iki bölümünden de görev verilebilir ve tek bir operatör, çalışma döngüsü sırasında iki farklı ürün üzerinde çalışabilir. Bu şekildeki montaj hatları, operatörler arasındaki iletişim ve ekip çalışmasını kolaylaştırmakta olup bu sayede motivasyonlarını artırmakta, üretim hacminde esneklik sağlamakta ve ürün kalitesini artırmaktadır (Scholl ve Klein, 1999).

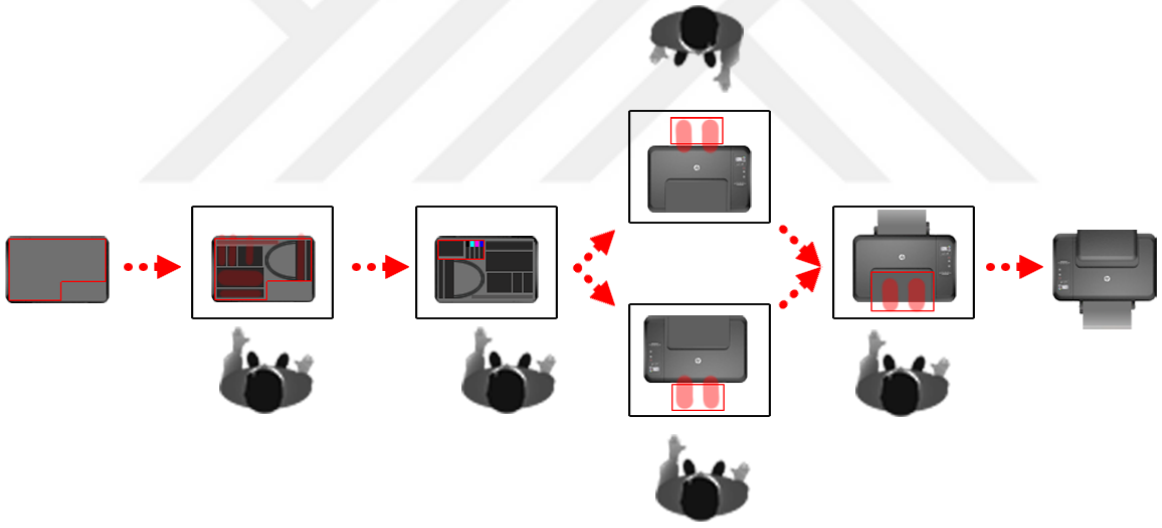


Şekil 5. U-Tipi Montaj Hattı Yapısı

Paralel yapı elemanları olan montaj hatları: Bir montaj prosesinin görevleri, temel işlemler olarak kabul edilir ve bu nedenle daha basit iş unsurlarına bölünemez. Bunun

sonucunda da en uzun işlem süresine sahip görev, montaj hattının çevrim zamanını yükseltir. Bu engeli aşmak amacıyla üretim hızının talep edilen hacmi karşılaması için çevrim zamanının kısalması gerektiğinde, montaj hattı tasarımına paralellik getirmek; yani çalışma istasyonlarını veya hattın kendisini paralellemek yaygın bir uygulamadır. Buna ek olarak; bu uygulama sayesinde esneklik artmakta ve montaj sisteminin arıza duyarlılığı azalmaktadır. Fakat buna karşın sabit maliyetler (kurulum) artmaktadır (Becker ve Scholl, 2006). En yaygın iki kullanım şekli bulunmaktadır:

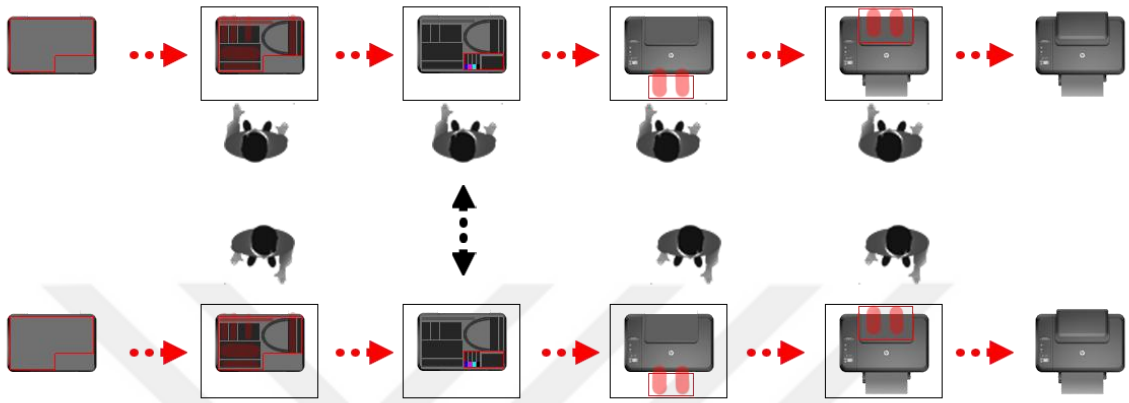
Paralel çalışma istasyonları: Bu konfigürasyonda montaj hattının aynı bölümünde kurulmuş ve paralel olarak aynı görevleri gerçekleştiren 2 ya da daha fazla aynı iş istasyonu bulunur. Paralel çalışma istasyonları, değişimli bir şekilde çalışma parçaları ile beslenir, bu sayede montaj hattının toplam çevrim zamanı çarpımı kadar çalışma süreleri artar (Becker ve Scholl, 2006; Boysen vd., 2008). Tek bir çalışma istasyonunun çoğaltıldığı basit bir paralel uygulama örneği, Şekil 6'da gösterilmiştir.



Şekil 6. Paralel Çalışma İstasyonlu Montaj Hattı Yapısı

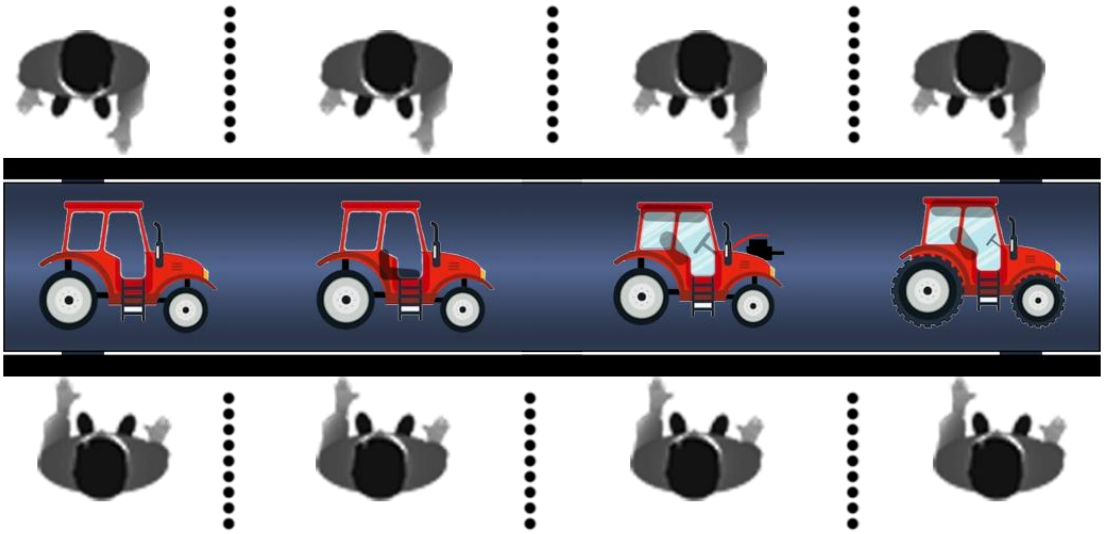
Paralel hatlar: Bu model, tek bir ürün modeline talebin tüm bir montaj hattının sayısının ikiye çıkarılmasını gerektirdiği veya birden çok ürünün bağımsız paralel hatlarda

monte edildiği durumlarda kullanılmaktadır. İlk durum, daha fazla alet ve ekipmana rağmen montaj hattının kısaltılmasına ve döngü sürelerinin artırılmasına imkan vermektedir. İkinci durumda ise, birden fazla montaj hattı arasında kaynak paylaşımının mümkün olması halinde iş yükü dengelenebilir ve bu sayede verimlilik artırılabilir (Şekil 7) (Becker ve Scholl 2006; Gökçen vd., 2006).



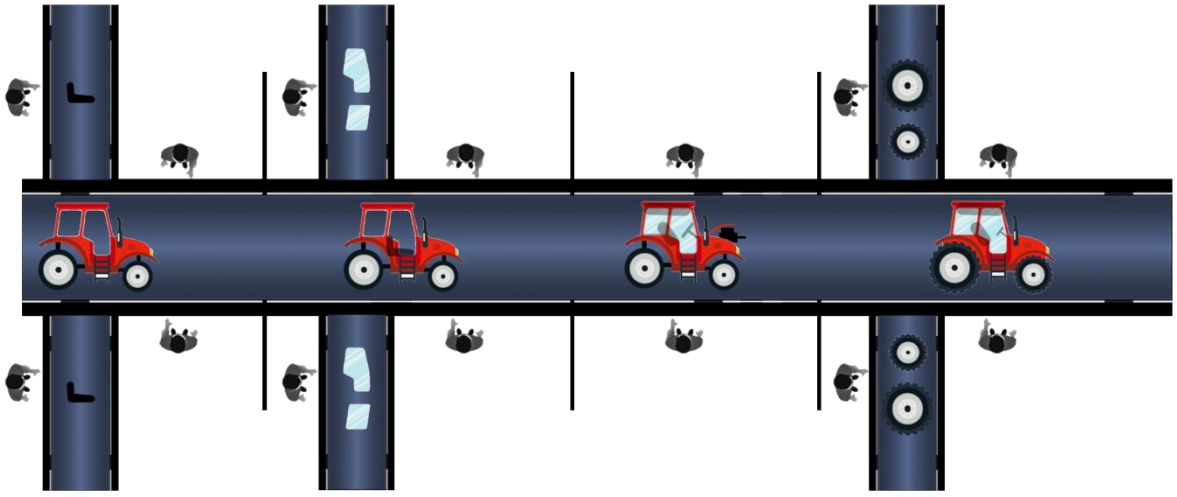
Şekil 7. Paralel Hatlı Montaj Hattı Yapısı

Çift taraflı montaj hatları: Bunlar, otomotiv gibi büyük boyutlu ürünlerin yüksek hacimde montajını sağlayan sistemler olup bir taşıma sistemi boyunca yerleştirilmiş karşılıklı çalışma istasyonu eşlerinden oluşmaktadır (Baykaşoğlu ve Dereli, 2008). Eşli çalışma istasyonları, birbirine bakacak şekilde hattın her iki tarafına yerleştirilir (Şekil 8). Bu sayede kendi başlarına veya eş çalışma istasyonuyla iş birliği içerisinde çalışma imkanı olur. Bu konfigürasyon, bir yandan aynı ürün üzerinde farklı görevlerin hattın her iki tarafında aynı zamanda gerçekleştirilmesine izin verirken diğer yandan da eş çalışma istasyonları aynı görevi gerçekleştirirken aynı anda çalışabilmektedir. Örneğin, ilk durum, araç montajında hattın belli bir tarafında veya herhangi bir tarafında gerçekleştirilmesi gereken görevleri (örn; sol tekerleğin takılması) ifade ederken ikinci durum, karşıt çalışma istasyonlarının koordinasyon içerisinde çalışmasını gerektirmektedir (örn; arka koltuğun montajı) (Bartholdi, 1993).



Şekil 8. Çift Taraflı Montaj Hattı Yapısı

Besleme veya alt montaj hatları: Montaj hatlarının gerçek uygulamalarında alt montaj hatları yoluyla ana montaj hattını besleyen bir veya daha fazla ek montaj hattından oluşan "balık kılçığı" yapısında karmaşık bir sistem bulunmaktadır (Şekil 9) (Tempelmeier, 2003; Lapierre ve Ruiz, 2004; Boysen vd., 2007).



Şekil 9. Besleme Montaj Hattı Yapısı

2.1.3.3. Otomasyon Seviyesine Göre Montaj Hatları

Otomatik montaj hatları: Bu hatlar genelde otomotiv endüstrisinde kullanılmakta olup belirli bir hassasiyet ile tekrarlayan görevleri yerine getiren özel amaçlı makinalar ve/veya robotlardan oluşmaktadır (Tempelmeier, 2003; Boysen vd., 2008). Böyle bir montaj sisteminin uygulanması ve işletilmesi için gereken büyük miktardaki yatırım maliyetinin gerekçelendirilmesi için de yüksek hacimli bir talep olması gerekmektedir. Otomatik hatlar, genelde 5 saniyelik minimum döngü süresi ile çalışmakta olup üretim hedefinin daha da kısa bir döngü süresi gerektirmesi halinde paralel hat düzeni önerilmektedir (Nof vd., 1997; Khan ve Day, 2001).

Manuel montaj hatları: Bu montaj hattı modelinde üretim birimlerinin montajı için gerekli görevleri gerçekleştirmek üzere insan operatörler çalıştırılmaktadır. Manuel hatlardan genelde hassas veya özel el yapımı ürünlerin (örn; Harley Davidson motorsikletleri) orta hacimli montajında yararlanılmakta olup iş gücünün ucuz olduğu ülkelerde yararlanılabilecek ekonomik bir sistemdir (Boysen vd., 2008). Manuel montaj hatlarında iş zenginleştirmesine imkan vermek adına döngü süreleri 30 saniyeyi aşmaktadır (Helander, 1995). Çünkü insan operatörleri tarafından tekrarlayan görevlerin yapılması düşük iş memnuniyetine ve motivasyon azlığına yol açtığı görülmüştür (Shtub ve Dar-El, 1989). Bunun için de düşük döngü süreleri için paralel manuel montaj hatlarından yararlanılması önerilmektedir (Khan ve Day, 2001).

2.1.3.4. İş Parçasının Akışına (Hat Kontrolüne) Göre Montaj Hatları

Zaman kısıtı olan montaj hatları: Bu çalışma modunda üzerinde çalışılan parçalar, konveyör bandı gibi otomatik bir malzeme taşıma ekipmanı vasıtasıyla eş zamanlı olarak çalışma istasyonlarının önüne getirilir (Boysen vd., 2008). Sonrasında her bir çalışma istasyonu kendi parçası üzerinde aynı anda çalışmaya başlar. Belirli bir süre (çevrim zamanı) bittikten sonra taşıma sistemi, bir sonraki pozisyona ilerler; bu şekilde üzerinde çalışılan parçalar sırayla işlenir ve sabit zaman aralıklarında bir sonraki çalışma istasyonlarına taşınır. Her bir iş döngüsünde, son çalışma istasyonunda işlenen parçanın montajı bitip hattan

ayrılırken ilk çalışma istasyonu yeni bir parça ile beslenir. Böyle senkronize çalışan bir hatta görev süresi farklılıklarından dolayı gerekli görevlerin çevrim zamanı içerisinde tamamlanamaması halinde ya tamamlanmamış görevler hat dışarısında tamamlanır ve üzerinde çalışılan parça daha sonraki bir aşamada hatta eklenir ya da gerekli tüm görevler gerçekleştirilene kadar hat durdurulur.

Zaman kısıtı olmayan montaj hatları: Bu operasyon modunda her bir çalışma istasyonu kendi hızında çalışır ve sabit zaman aralıklarında değil, gerekli operasyonlar ne zaman tamamlanırsa parçayı o zaman bir sonraki çalışma istasyonuna aktarır (Boysen vd., 2008). Bu modda iki alternatif bulunabilir (Buzacott ve Shanthikumar, 1993): a) üzerinde çalışılan parçaların çalışma istasyonlarına eş zamanlı şekilde iletildiği senkronize mod (montaj hattı üzerindeki gerekli tüm görevler ne zaman tamamlanırsa) ve b) bir çalışma istasyonunun üzerinde çalıştığı parçayı kendi görevi tamamlandıktan sonra ve bir sonrakinin almaya hazır olduğu zaman bir sonraki çalışma istasyonuna aktardığı asenkronize mod. Asenkronize bir hatta, geçici görev süresi farklılıklarından doğabilecek istasyon aç kalması veya tıkanması etkilerini azaltmak için çalışma istasyonları arasına tamponlar kurulabilir. Açlık, yukarı yöndeki tamponun boş kalmasından dolayı bir çalışma istasyonunun boşta çalışması nedeniyle meydana gelirken tıkanma, aşağı yöndeki tamponun dolu olması nedeniyle yukarı yöndeki çalışma istasyonlarından işlemiş parçayı alamaması nedeniyle meydana gelir.

2.1.4. Montaj Hattı Dengeleme Problemleri ve Sınıflandırılması

Montaj hattı konfigürasyonları, üretimin en önemli bileşenleri arasındadır. Endüstriyel sorunlarının çoğu, montaj hatlarından kaynaklanmaktadır. Montaj hattı konfigürasyonunda verimlilik elde etmek için çözülmesi gereken en önemli sorun, hat dengelemesidir. Montaj hattı dengeleme problemi (MHDP), uygulanabilir bir hat dengesinin bulunmasına yönelik bir problem olup her bir görevi, öncelik kısıtlamaları ve diğer sınırlamalar yerine getirilecek şekilde bir istasyona verebilmektir. "Dengeleme" kelimesi, çalışma istasyonu sayısının en aza indirilmesi ile tüm istasyonlarda elde edilecek dengeli bir iş yükününün sağlanmasına ve tüm istasyonlardaki toplam boş zamanın en aza indirilmesi ile ulaşılmasından gelmektedir.

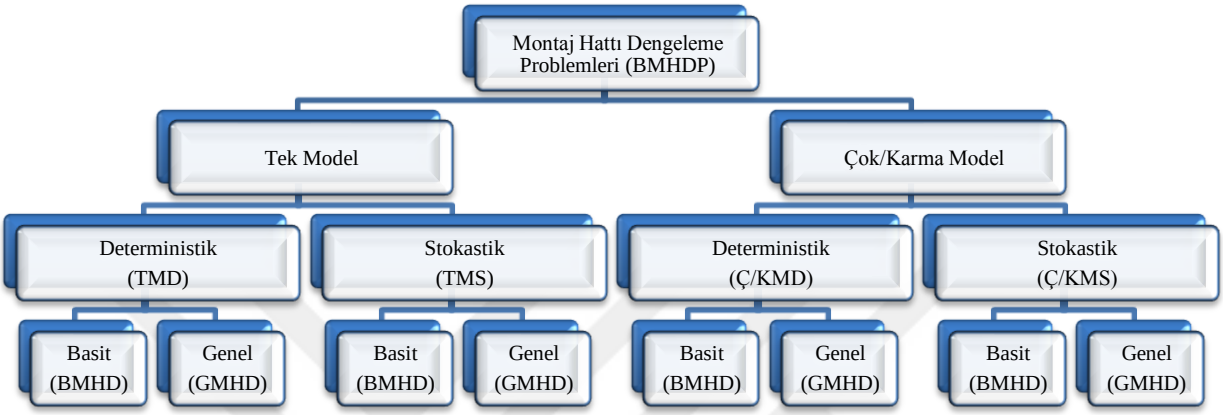
Montaj hattı üzerindeki istasyonlara görev verilmesine ilişkin bir üretim planlama problemi olan montaj hattı dengeleme problemi (MHDP), Salveson tarafından 1955 yılında matematiksel programlama formülasyonunu yapılana kadar geçici olarak yönetilmiştir. 1956 ile 1961 yılları arasında bazı yazarların da katkısıyla (Jackson, 1956; Bowman, 1960; Supnick ve Solinger, 1960; White, 1961; Hu, 1961) montaj hattı dengeleme, bir araştırma alanı olarak ortaya çıkmıştır. Son 65 yıl içerisinde hem teknolojik destek ve fiziksel montaj hatlarının karmaşıklığı olsun hem de MHDP'nin çözülmesinde kullanılan yöntemler olsun bu alanda çok çeşitli uyarlamalar ve yenilikler ortaya çıkmıştır.

Montaj hattı dengeleme problemlerine yönelik olarak dal ve sınır algoritmaları (Glover, 1989) ve dinamik programlama (Nearchou, 2008; Becker ve Scholl, 2006) gibi çeşitli optimal çözüm teknikleri geliştirilmiştir. Bu teknikler, Ali ve Törn (2004) tarafından detaylı incelenmiştir. Sorunun kombinatoryal doğası nedeniyle optimal çözüm algoritmalarının büyük boyutlu problemlere uygulanabilirliği kısıtlıdır. Geleneksel (düz) montaj hattı dengeleme problemlerinin NP-zor olduğu bilinmektedir (Kaelo ve Ali, 2006). M görev ve r sıra kısıtlaması varsa $m!/2^r$ kadar olası görev dizisi vardır (Nearchou, 2006). Böylesine geniş bir çeşitlilik içerisinde kesin algoritmaları kullanarak optimal bir çözüm elde etmek neredeyse imkansızdır. Bu nedenle, bazı araştırmacılar dikkatlerini MHDP çözümü için sezgisel (Rachamadugu ve Talbot, 1991) ve metasezgisel yöntemlere yöneltmiştir.

Montaj hattı dengeleme problemleri, üretim planlama ve operasyon yönetiminin birbiriyle bağlantılı sayısız sorunlarından biridir. Bu sorunlar genellikle hiyerarşik olarak çözülür. Yani, en uzun vadeli planlama sorunları önce ele alınır. Tesis konumu, yerleşim düzeni ve kapasite planlama gibi problemler uzun vadeli sorunlardan kabul edilmekte olup çözümleri 1 ile 10 yıl içerisinde tekrar gözden geçirilir. İş gücü yönetimi (işe alım, işten çıkarma, eğitim) problemleri ile MHDP de orta vadeli planlama problemleridir. Orta vadeli problemler ile genellikle bir kaç haftada veya ayda bir ilgilenilir. Kısa vadeli problemlere satın alım, nakliye, güzergah problemleri sayılabilir ve günlük olarak ilgilenilir.

Montaj hattı dengeleme problemlerinin sınıflandırılması için geçmişte farklı kategoriler ortaya atılmıştır. Örneğin, Kriengkorakot ve Pianthong (2007) montaj hattı dengeleme problemlerini literatüre göre iki kategoriye ayırmıştır. İlk kategori, montaj hattı

dengeleme problemlerini tek model ve çoklu/karışık model olarak iki ana modele ayıran Ghosh ve Gagnon (1989) tarafından tanımlanmıştır. Sonrasında iki tür görev süresi olduğunu öne sürerek her modeli deterministik ve stokastik zamana göre ayırmışlardır. Son olarak, iki görev süresini de basit ve genel montaj hattı olarak gruplandırmışlardır. Bu kategoriler, Şekil 10'da gösterilmiştir.



Şekil 10. Montaj Hattı Dengeleme Problemlerinin Sınıflandırılması 1

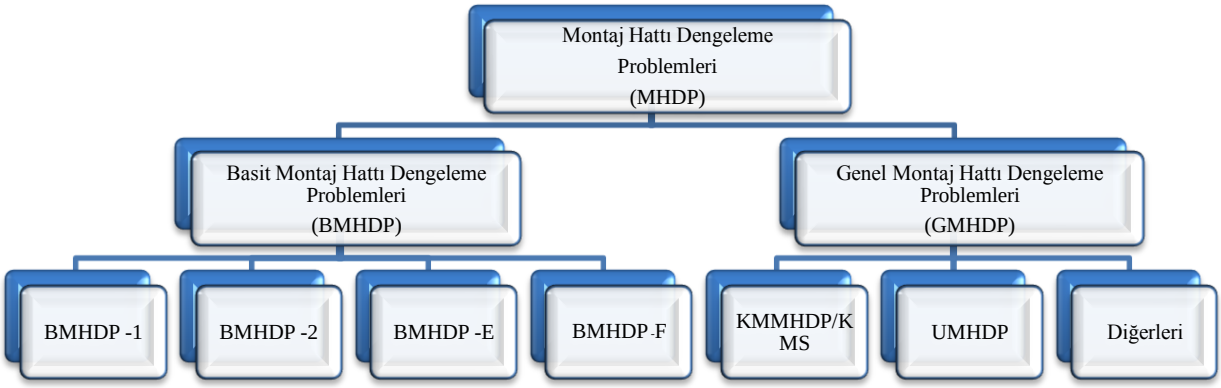
Kaynak: Ghosh ve Gagnon, 1989

Tek Modelli Deterministik (TMD), montaj hattı üzerinde deterministik operasyon süresi olan tek bir modeli ifade etmektedir. En basit montaj hattı dengeleme (BMHD) türüdür. Kısıtlamaların eklenmesi veya varsayımların değişmesi halinde bu tür, genel montaj hattı dengeleme (GMHD) türüne dönüşür. BMHD'nin GMHD'ye dönüşmesine iki örnek olarak bölge kısıtlamalarının getirilmesi veya montaj hattında paralel çalışma istasyonlarının kullanılması verilebilir. Tek Modelli Stokastik (TMS), montaj hattı üzerinde olasılıksal operasyon süresi olan tek bir ürün modelini ifade etmektedir. Bu model türü, manuel montaj hatlarında kullanılır. Çoklu/Karışık Modelli Deterministik (ÇKMD), çoklu veya karışık modelli montaj hattındaki deterministik görev süresini ifade etmektedir. Çoklu/Karışık modelde tekli modelde olmayan başlatma hızı ve model sıralaması gibi çeşitli faktörler bulunmaktadır. Çoklu/Karışık Modelli Stokastik (ÇKMS), çoklu/Karışık modelli montaj

hattındaki değişken operasyon süresini ifade etmektedir. TMS modeli ile ilişkilendirilen tüm nesneler, ÇKMS modelinde de bulunmakta fakat TMS modeline kıyasla daha karışıktır.

İkinci kategori Scholl ve Becker (2006) ve Becker ve Scholl (2006) tarafından tanımlanmıştır. Ghosh ve Gagnon'a (1989) göre farklı sınıflandırmalar önermişler ve montaj hattı dengeleme problemlerini, basit montaj hattı dengeleme problemi (BMHDP) ve genel montaj hattı dengeleme problemi (GMHDP) olmak üzere iki ana gruba ayırmışlardır. Sonrasında çeşitli hedefleri göz önünde bulundurarak sınıflandırmalarını basit montaj hattı dengeleme problemi için BMHDP-1, BMHDP-2, BMHDP-E ve BMHDP-F; genel montaj hattı dengeleme problemi için ise KMHDP/KSP ve UMHDP ile genişletmişlerdir. İlgili sınıflandırma, Şekil 11'de gösterilmiştir. Basit Montaj Hattı Dengelemenin (BMHD), düz bir konveyör bantlı bir seri hattaki tek bir ürün için kullanılması uygundur. BMHDP için tek kısıtlama, öncelik kısıtlamasıdır. BMHDP-1'in hedefi, çevrim zamanı sabitken montaj hattındaki çalışma istasyonlarının sayısını en aza indirmekken; BMHDP-2'nin hedefi, çalışma istasyonu sayısının bilindiği durumlarda çevrim zamanını en aza indirmektir. BMHDP-E'nin hedefi, çevrim zamanını ve çalışma istasyonu sayısını aynı anda azaltmak ve hat verimliliğini maksimuma çıkarmak iken BMHDP-F'nin hedefi ise çalışma istasyonu ve çevrim zamanı bilinen hat dengeleme problemine uygun bir çözüm bulmaktır.

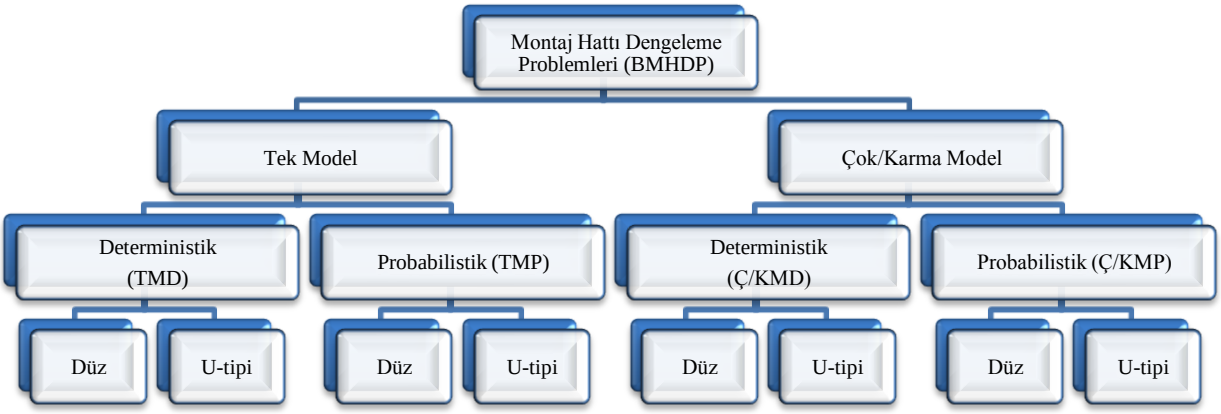
Genel montaj hattı dengeleme problemlerinde (GMHDP), basit montaj hattı dengeleme problemlerinin (BMHDP) bazı özellik ve varsayımları çıkarılmış veya değiştirilmiştir. Bu kategoride Karışık Modelli Montaj Hattı Dengeleme Problemi'nin (KMMHDP) hedefi, maliyet ve kapasitenin optimize edecek çalışma istasyonu sayısı, çevrim zamanı ve hat dengeleme tekniklerini bulmaktır (Scholl, 1999). Diğer yandan, Karışık Modelli Sıralama Problemi'nin (KMSP) hedefi ise hat verimsizliğinin en aza indirilmesi gibi çeşitli hedeflere ulaşılması için bir ürünün farklı modelleri için sıralamayı tespit edebilmektir (Scholl vd., 1998). U-tipi Montaj Hattı Dengeleme Problemi (UMHDP), U-tipi montaj hattındaki tek bir model için uygundur. Bu durumda operatörler, U'nun her iki tarafında da bulunabilen birkaç çalışma istasyonunda çalışabilirler (Miltenburg ve Wijngaard, 1994; Erel vd., 2001).



Şekil 11. Montaj Hattı Dengeleme Problemlerinin Sınıflandırılması 2

Kaynak: Scholl ve Becker, 2006; Becker ve Scholl, 2006

Sivasankaran ve Shahabudeen (2014), Ghosh ve Gagnon'un (1989) sınıflandırmasına benzer başka bir montaj hattı dengeleme sınıflandırması tanımlamıştır. Öne sürdükleri sınıflandırma, Şekil 12'de gösterilmiştir. Ghosh ve Gagnon'un (1989) sınıflandırması ile bu sınıflandırma arasındaki en büyük fark, Ghosh ve Gagnon'un (1989) sınıflandırmasında göz önünde bulundurulmayan çalışma istasyonu düzenidir. Sivasankaran ve Shahabudeen (2014) sınıflandırmasında düzen, düz hat ve U-tipi hat olmak üzere iki gruba ayrılmıştır. Düz hatta çalışma istasyonları, montaj işlemi düz konveyör bandı kullanılarak düz bir şekilde adım adım ilerlemektedir. U-tipi hatta ise çalışma istasyonları U şeklinde yerleştirilmiştir. Bu montaj hattında operatörler birden fazla istasyonda çalışabilmektedir.



Şekil 12. Montaj Hattı Dengeleme Problemlerinin Sınıflandırılması 3

Kaynak: Sivasankaran ve Shahabudeen, 2014

2.1.5. Montaj Hattı Dengeleme Problemlerine Çözüm Yaklaşımları

Montaj hattı dengeleme problemlerine yönelik çözüm prosedürleri genellikle kesin ve yaklaşık olmak üzere iki kategoriye ayrılır. En basit hat dengeleme problemi bile NP-zor olduğu için; hat dengeleme problemlerinin büyük bir kısmında problem boyutu arttıkça kesin yöntemlerle optimum sonuca ulaşmak için geçen hesaplama süresi, büyük oranda artmaktadır. Bu nedenle daha büyük boyutlu problemler için yaklaşık yöntemlere ihtiyaç vardır. Buradaki amaç; kabul edilebilir bir hesaplama süresi içerisinde iyi ve uygulanabilir bir çözüme ulaşmaktır (Adham vd., 2013; Junsong, 2014).

2.1.5.1. Kesin Yöntemler

Hat dengeleme problemleri genelde şu iki yaklaşımdan biri ile çözülmektedir: standart bir çözücü veya özgün bir çözüm yöntemi. İkinci durumda amaç, problemi olabildiğince hızlı çözebilmek için uygun matematiksel programlama modellerini tanımlamak ve çözücü parametrelerini ayarlamaktır. Hat dengeleme problemlerini açıklamak için literatürde öne sürülen matematiksel modellerin büyük bir kısmı, karma tam sayılı programlama modellerini içine almaktadır (Miralles vd, 2007; Corominas vd, 2008; Pastor, 2011; Delorme vd., 2012). Kullanılan diğer matematiksel modeller arasında tamsayılı doğrusal programlama (Bowman,

1960), doğrusal olmayan tamsayılı programlama (Hamta vd., 2011) ve hedef ve bulanık hedef programlama (Özcan ve Toklu, 2009) bulunmaktadır.

Çözücüler, genel optimizasyon problemlerini çözmek için tasarlandıkları için, belirli hat dengeleme problemleri için yeterince verimli olmayabilir. Bu durumda dinamik programlama (Güngör ve Gupta, 2001; Dolgui vd., 2008) ve dal ve sınır (Miralles vd, 2008; Hu vd., 2010; Borisovsky vd., 2012; Sewell ve Jacobson, 2012) gibi özgün yöntemler geliştirilebilir.

2.1.5.2. Yaklaşık Yöntemler

Literatürde montaj hattı dengeleme problemlerini çözmek için çok çeşitli yaklaşık yöntem öne sürülmüştür (Scholl ve Voβ, 1996; Amen, 2001). Bu yaklaşık yöntemler; sınırlı kesin yöntemler, basit sezgisel ve metasezgisel olmak üzere üç kategoriye ayrılabilir.

1.Sınırlı kesin yöntemlerde, çözüm uzayının eksik bir sayımı yapılmaktadır. Bu da, araştırılan çözüm uzayını kısıtlayarak veya mevcut hesaplama zamanını azaltarak var olan kesin yöntemleri sınırlamayla elde edilebilir (Blum ve Miralles, 2011; Bautista ve Pereira, 2011).

2.Basit sezgisel yöntemler, genelde konuya özel ve probleme bağlı tekniklerdir. Çoğu durumda görevlendirme yapılırken öncelik kurallarından yararlanılır. Bu kuralların temelinde, görev zamanı veya takipçi sayısı gibi özellikler yer almaktadır (Capacho ve Pastor 2006; Scholl ve Becker, 2006; Pastor vd., 2012). Basit sezgisel yöntemler, iki sınıfa ayrılabilir:

- *Tek geçişli sezgisel yöntemler:* Görev dağılımı, açgözlü fonksiyon veya öncelik kuralından yararlanılarak tek bir tekrarla gerçekleştirilir (Toksarı vd., 2008). Bu yöntemler kullanılarak büyük çaplı problemler söz konusu olduğunda bile kısa sürede uygun çözümler elde edilmektedir.
- *Çok geçişli sezgisel yöntemler*: Bu algoritmaların rassal niteliğe sahip olması nedeniyle farklı sonuçlar elde edilebilmekte ve bulunan sonuç, birkaç tekrardan sonra en iyi çözüm olarak tanımlanabilmektedir (Andrés vd., 2008). Verilecek bir görevin seçiminde de rastgelelikten faydalanılabilir. Belli bir görev, aday

listesinden (Toksarı vd., 2010) veya açgözlü fonksiyonu en yüksek değerde olanlar arasından (Guschinskaya vd., 2011) ve rasgele bir öncelik kuralına göre (Gamberini vd., 2009) seçilebilir. Önceden belirlenmiş bir iterasyon sayısına ulaşıldığında veya elde edilen en iyi çözümden belli bir iterasyon sayısını geçtikten sonra daha iyi bir çözüm bulunmaması halinde program durdurulur.

Kesin bir yöntem için üst sınır belirlemek üzere (Baldacci vd., 2004) veya ara çözümlerin lokal olarak geliştirilmesi adına metasezgisel yöntemlere entegre edilmek üzere (Essafi vd., 2012) basit sezgisel yöntemlerden yararlanılabilir.

3.Metasezgisel ve hibrit metasezgisel yöntemler: Metasezgisel yöntemler, her bir problem için derinlemesine uyarlama yapmak zorunda kalmadan pek çok zor optimizasyon problemini çözmek üzere tasarlanmış genel nitelikte yöntemlerdir. Metasezgisel yöntemler ile ilgili detaylı bilgi için Boussaid vd.'lerinin (2013) çalışmasına bakılabilir. Bu tür yöntemler, sayıca fazla ve çeşitli olsa da kabaca aşağıda gösterilen sınıflara ayrılabilir:

- *Komşuluk yaklaşımları:* Tabu araması (Glover ve Laguna, 1997; Özcan ve Toklu, 2009), GRASP (Chica vd., 2010), benzetimli tavlama (Kirkpatrick vd., 1983; Jayaswal ve Argawal, 2014), değişken komşu araması (Hansen ve Mladenovic, 1999), vs.
- *Evrimsel yaklaşımlar:* Diferansiyel gelişim yöntemleri (Mozdgir vd., 2013), genetik algoritmalar (Kazemi vd., 2011), emperyalist rekabetçi algoritmalar (Bagher vd., 2011) veya memetik algoritmalar (Gamberini vd., 2009).
- *Sürü zekası temelli metasezgisel yöntemler:* Parçacık sürü optimizasyon algoritmaları (Nearchou, 2011), arı algoritmaları (Tapkan vd., 2011) ve karınca koloni optimizasyonu (Bautista ve Pereira, 2007).

Son yıllarda araştırmaların odak noktası, diğer optimizasyon teknikleri ile birlikte metasezgisel yöntemlerin hibritleştirilmesine doğru kaymıştır. Metasezgisel yöntemleri hibritleştirmenin ardındaki temel etken, farklı optimizasyon stratejilerinin birbirini tamamlar nitelikteki özelliklerden yararlanmaktır. Raidl (2006) ve Blum vd. (2011) tarafından yapılan

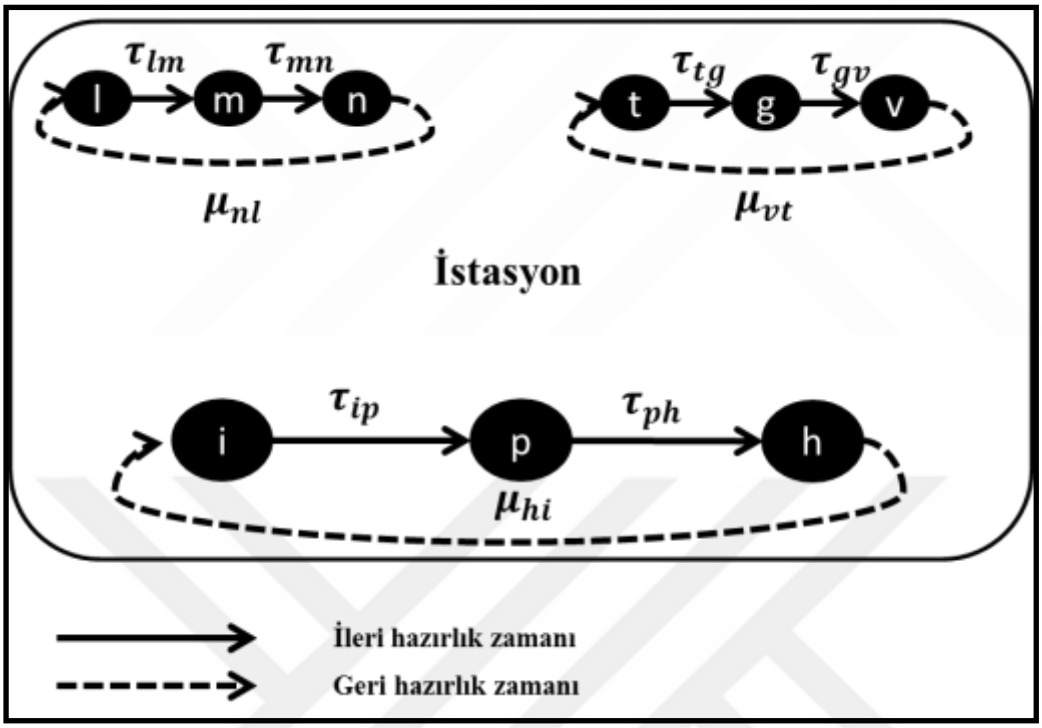
çalışmalarda, kombinatoryel optimizasyonda hibrit metasezgisel yöntemler ile ilgili detaylı literatür taramaları verilmiştir.

Yeni ve gelecek vadeden bir eğilim de, matsezgisel yöntemlerdir (Maniezzo vd., 2009). Metasezgisel ve kesin yöntemlerin birbiriyle etkileşimi, bu yöntemlerin temelini oluşturmaktadır. Ancak yeni olmaları nedeniyle konsolide bir sınıflandırma şekli değildir ve bu nedenle bu yöntemlerin kendilerine has tanımlarına rastlamak zordur. Matsezgisel yöntemler ile ilgili daha ayrıntılı bilgi için Maniezzo vd. (2009) tarafından gerçekleştirilen çalışmaya bakılabilir.

2.1.6. Sıra-Bağımlı Hazırlık Zamanlı Genel Montaj Hattı Dengeleme Problemleri

BMHDP'de, görevler arasında hazırlık zamanları varsa ve görev sırasına bağımlı olarak değişiyorsa bu tarz problemlere sıra-bağımlı hazırlık zamanlı GMHDP denmektedir. Andres vd.'den (2008) önce hazırlık zamanları dikkate alınmamıştır. Fakat robotik hatlar, otomotiv ve elektronik montaj hatları gibi hazırlık zamanlarının yüksek oranda olduğu hatlarda hazırlık zamanlarınının gözardı edilmesi verimli bir dengeleme yapılmasına engel olmaktadır. Andres vd.'ne (2008) göre robotik hatlarda bir görevden diğerine geçildiği zaman bazı aletlerin değiştirilmesi için bir hazırlık zamanına ihtiyaç duyulmaktadır. Becker ve Scholl (2009), otomotiv montaj hatlarında araç gövdesi üzerinde çalışan işçiler için yürüme zamanlarının hesaba katılması gerektiğini savunmuşlardır. Ayrıca yürüme zamanlarının öncül-ardıl olan işe göre değişebildiğini söylemişlerdir.

Bu problem şu şekilde tanımlanabilir: Bir i görevi herhangi bir j görevinin öncülü ise ve aynı istasyona atandıysa görevler arasında $\tau_{i,j}$ kadar bir hazırlık zamanı gerekebilir. Bu süreye ileri (forward) hazırlık zamanı denilmekte ve istasyonun toplam zamanına eklenmesi gerekmektedir. Scholl vd. (2013), buna ek olarak geri (backward) hazırlık zamanını eklemiştir. Eğer bir istasyonda p son görev olarak yer alıyor ve i de birinci görev olarak yer alıyorsa, iki görev arasında $\mu_{p,i}$, kadar bir geri (backward) hazırlık zamanı gerekebilir. Dolayısıyla bu geri hazırlık zamanının da istasyon süresine eklenmesi gerekir. Aşağıdaki Şekil 13'te ileri ve geri hazırlık zamanları gösterilmiştir.



Şekil 13. İleri ve Geri Hazırlık Zamanları

Kaynak: Güner**,** 2019

Problemin daha iyi anlaşılması açısından şöyle bir örnek verilebilir: 3 görevden oluşan ve aralarında öncül-ardıl ilişkisi olmayan A,B,C görevleri aynı istasyonda yer almaktadır. Görev zamanları: τ_A=15, τ_B=12, τ_C=10 olsun. İleri ve geri hazırlık zamanları: $\tau_{A,B}$=3, $\tau_{A,C}$=4, $\tau_{B,C}$=5, $\tau_{B,A}$=2, $\mu_{C,A}$=2, $\mu_{B,A}$=1, $\mu_{B,C}$=2, $\mu_{C,B}$=1 olduğunu varsayalım. Tablo 2'de iki farklı şekilde sıralanan görevlerin toplam istasyon zamanları verilmiştir.

Tablo 2. A, B ve C görevlerinin İki Farklı Şekilde Sıralanışı

Sıralama	Dikkate Alınacak Zamanlar	Toplam İstasyon Zamanı
A-B-C	15+3+12+5+10+2	47
B-A-C	12+2+15+4+10+1	44

Aynı istasyonda yer alan bu üç görevin iki farklı dizilişi ile aralarında 3 birimlik zaman farkı oluşmuştur. Yukarıdaki örnekte çevrim zamanının 45 olduğunu varsayalım. Bu durumda A-B-C sıralaması uygun bir çözüm olmaz; ancak B-A-C uygun bir çözüm olmaktadır. Andres vd.'ne (2008) montaj hatları dengelenirken göre hem görevlerin istasyonlara atanması gerekir hem de atanan görevlerin istasyon içinde de çizelgelenmesi gerekmektedir. Dolayısıyla sıra-bağımlı hazırlık zamanlı GMHDP'nin hem bir dengeleme, hem de bir çizelgeleme problemi olduğu söylenebilir. Bu durum da problem çözümünü çok daha fazla zorlaştırmaktadır.

2.2. Diferansiyel Gelişim Algoritması (DGA)

Diferansiyel Gelişim Algoritması (DGA), Storn ve Price (1997) tarafından geliştirilen popülasyon temelli sezgisel bir eniyileme yöntemidir. DGA, her ne kadar sürekli ve küresel eniyileme problemlerinin çözümü için ortaya çıkmış olsa da Nearchou (2006) tarafından ilk defa permütasyonel ve kesikli olan makina yerleşim probleminde başarıyla uygulanmıştır.

DGA işleyiş yapısı ve kullandığı operatörler bakımından genetik algoritmaya (GA) benzemektedir. Aralarındaki temel farklardan biri, GA arama işlemini ağırlıklı olarak çaprazlama işlemi üzerinden yaparken, DGA ise arama işlemini daha çok mutasyon işlemi üzerinden yapmaktadır. Dolayısıyla DGA'da iyi bir arama yapabilmek için mutasyon işleminin etkin çalışması gerekir.

DGA diğer metasezgisellerden farklı olarak seçilim aşamasında en iyi bireylerin yaşamasına izin verir. Ayrıca her bir iterasyonda yeni nesiller oluşturmak yerine, ilk nesilde oluşturulan bireyler iterasyonlar boyunca varlıklarına devam etmekte ancak kendinden daha iyi bireyler oluştuğu zaman onunla yer değiştirirler. Böylece mevcut durumda kötü bir sonuç

verse bile, küresel en iyiye ulaşabilecek çözümlerin devamı ve bireylerin çözüm uzayında dağıtıklığı sağlanarak erken yakınsamanın önüne geçmektedir.

Klasik DGA, temel olarak başlangıç popülasyonunun oluşturulması, mutasyon, çaprazlama ve seçilim olmak üzere dört temel aşamadan oluşmaktadır.

2.2.1. Başlangıç Popülasyonunun Oluşturulması

DGA'nın ilk aşamasında N_p adet D boyutlu parametreleri gerçel sayı olan vektör $x_{i,G}$ oluşturulur.

$$X_{i,G}\ ,\ \ i = 1,2,3,\dots\dots,N_p \tag{2.6}$$

$$G = 0,1,2,\dots\dots,G_{max} \tag{2.7}$$

$$S = (X_{1G}, X_{2G}, X_{3G}, \dots\dots, X_{N_pG}) \tag{2.8}$$

Burada i popülasyon indisini; G, nesil sayısını ve S ise başlangıç popülasyonunu göstermektedir.

2.2.2. Mutasyon

Her bir hedef vektör $X_{i,G}\ ,\ \ i = 1,2,3,\dots\dots,N_p$ için bir mutant vektör v_{iG+1} aşağıdaki gibi oluşturulur.

$$v_{iG+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \tag{2.9}$$

Burada $r_1, r_{2,} r_3\, \epsilon(1,2,3,\dots\dots,N_p)$ birbirinden farklı tamsayılardır. Burada $r_1, r_{2,} r_3$ tamsayıları hedef vektör olan $X_{i,G}$'nin i indisinden farklı olduğu için popülasyon sayısı N_p, dört veya daha büyük olması gerekir. $F\epsilon[0,2]$ ise $\left(x_{r2,G} - x_{r3,G}\right)$ vektörlerin farkını artıran veya azaltan kullanıcı tarafından belirlenen gerçel bir katsayıdır.

2.2.3. Çaprazlama

Mutasyona uğramış vektörlerin çeşitlendirmesini arttırmak için çaprazlama operatörü kullanılır. Bu işlem sonunda aday vektör $u_{k,iG+1}$ aşağıdaki gibi oluşturulur.

$$u_{k,iG+1}=\begin{cases} v_{k,iG+1}\ eğer\ rasg \leq CR\ ya\ da \\ x_{k,iG}\ eğer\ rasg > CR \end{cases} \tag{2.10}$$

$$k = RasgTams(1, D), i \in [1, N_p]$$

Burada rasg, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı;, $CR \in [0,1]$, kullanıcı tarafından belirlenen çaprazlama olasılığı ve RasgTams(1,D) ise [1,D] aralığında rasgele tamsayı üreten bir fonksiyondur. Bu fonksiyon en azından bir parametrenin mutant vektörden alınmasını garantilemektedir.

2.2.4. Seçilim

Seçilim operatörünün amacı bir sonraki nesil olan G+1 neslini oluşturmaktır. Bir sonraki nesile aktarılan vektöre x_{iG+1} aşağıda Eş. 2.11'deki gibi karar verilir.

$$x_{iG+1}=\begin{cases} x_{iG} & eğer\ uyg(x_{iG}) \leq uyg(u_{iG+1}) \\ u_{iG+1} & aksi\ takdirde \\ & i \in [1, N_p] \end{cases} \tag{2.11}$$

Seçilim işleminde hedef ve aday çözüm vektörlerinin uygunluk değerlerine bakılır hangisi daha küçükse o vektör bir sonraki nesle aktarılır.

2.3. Parçacık Sürü Optimizasyonu (PSO)

Parçacık sürü optimizasyonu, Kennedy ve Eberhart (1995) tarafından ortaya atılmış popülasyon temelli bir metasezgisel yöntemdir. PSO'nun ilham kaynağı, sürü halinde hareket eden kuş ve balıkların sosyal davranışlarıdır. PSO, her bir parçacığın bir çözüme karşılık geldiği ve parçacıkların global optimuma ulaşmak adına çözüm uzayında arama yaptığı popülasyon temelli bir algoritmadır. Popülasyonun tüm üyeleri (parçacıkları), bu arama işlemi sırasında varlıklarını sürdürür ve çözüm uzayındaki aramayı en iyi konuma yönlendirmek

amacıyla bu üyelerin sahip olduğu bilgiler sosyal olarak paylaşılır. Parçacık adı verilen potansiyel çözüm, belli bir hız doğrultusunda çok boyutlu problem boşluğunda seyrine devam eder ve o anda en iyi bilinen parçacıkları takip eder. Bu arama sırasında her bir parçacık, konumunu kendi deneyimlerine ve komşu parçacıkların deneyimlerine göre ayarlar. Komşuluk kavramı, lokal bir komşuluk ya da global bir komşuluk anlamına gelebilir ve bu durumda temel PSO algoritması ile iki varyasyon ortaya çıkar (Kennedy vd., 2001).

PSO algoritmasının amacı, amaç fonksiyonun $f: S \subset \mathbb{R}^n \rightarrow \mathbb{R}$ optimumunu bulmaktır. $S \subset \mathbb{R}^n$ arama uzayı ve $f: S \rightarrow Y \subseteq \mathbb{R}$ ise amaç fonksiyonu olsun. Burada *n* arama uzay boyutu ve *S,* kabul edilebilir problem uzayı olsun. Sürü $A = \{x_1, x_2, \ldots, x_N\}$ ile tanımlanır ve *N* ise tanımlanmış ve önceden belirlenmiş parçacık sayısını temsil eder**.** Parçacıklar, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in S$, $i=1, 2,\ldots,N$ ile tanımlanır. Dolayısıyla x_i arama uzayındaki *i* parçacığının pozisyonunu , $f_i = f(x_i) \in Y$ ise çözüm uzayındaki uygunluk değerini temsil eder. Her bir parçacığın pozisyonu, hız değerlerine göre tanımlanır ve $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, $i=1, 2,\ldots,N$ ile temsil edilir. PSO, hız denklemi (Eş. 2.12) ve pozisyon denklemi (Eş. 2.13) gibi iki ana denkleme bağlıdır.

$$v_i^{t+1} = v_i^t + r_1 c_1 (Pbest_i^t - x_i^t) + r_2 c_2 (Gbest^t - x_i^t) \quad (2.12)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (2.13)$$

r_1 ve r_2, [0, 1] arasında aynı şekilde düzgün dağılmış değerler olup birbirinden bağımsızdır. c_1 ve c_2 değerleri, pozitif ivme katsayılarıdır. Hız, algoritmada elde edilen önceki bilgilere göre değişmektedir. Her bir parçacık, ulaşılan en iyi konumu ezberler. Bu nedenle PSO algoritmasının, parçacıkların konumunu koruyacak bir A sürüsü ve her bir parçacığın ulaştığı en iyi konum ile birlikte Pb = {$Pbest_1$, $Pbest_2$, . . . , $Pbest_N$} şeklinde ifade edilebilecek bir hafıza seti bulunmaktadır. Bu parçacıklar, daha önce bulundukları en iyi konum ile global optimuma ulaşma eğilimindedir. Tüm sürü parçacıkları arasında en iyi konum, $Gbest = min\ f(Pbest_i)$ şeklinde ifade edilebilir.

2.3.1. Başlangıç Popülasyonunun Oluşturulması

PSO algoritmasında ilk adım, başlangıç popülasyonun (sürünün) oluşturulmasıdır. Parçacık konumlarının oluşturulması, genelde rastgele gerçekleştirilir. Sürünün oluşturulması, parçacıkların arama uzayında eşit şekilde dağılması sağlanarak yapılmalıdır (Engelbrecht, 2007). Büyük boyutlu problemlerde başlangıç popülasyonu, rastgele yapılır ve bazı durumlarda parçacıklar eşit olmayan şekilde dağılabilir (Gutierrez vd., 2011). Arama uzayındaki alanın, her bir parçacık için sınırlı vektörler olan L ve H ile tanımlandığı düşünülürse parçacık konumunun etkin şekilde oluşturulması şu şekilde olacaktır:

$$x_i(0) = L_i + r_i(H_i - L_i), \forall i = 1, 2, 3, \ldots, n \quad (2.14)$$

$r_i \sim U(0, 1)$ ve n, x karar vektörünün boyutu anlamına gelirken; i ise x'in i. boyutudur.

Parçacık hızı, genelde sıfır ile başlatılır. Ayrıca rastgele oluşturulmuş başka değerler de kullanılabilir. Ancak, hız ile ilgili ilk değerlerin büyük olmaması gerekir. Zira bu durum, parçacıkların ilk tekrarda arama uzayının dışına çıkmasına neden olabilir. Pbest, parçacığın ilk uyum değeri ile başlatılır. c_1 ve c_2 gibi PSO parametreleri, bu bölümün devamında açıklanmaktadır. Farklı hız başlatma yöntemleri ile ilgili olarak Engelbrecht (2012), en iyi yaklaşımın parçacık hızını sıfıra veya sıfıra yakın rasgele bir değer ile başlatmak olduğunu savunmuştur.

2.3.2. Hız

Hız, Eş. 2.12'den yararlanılarak değerlendirilmektedir. $\mathbf{v}^t$ ve $\mathbf{x}^t$, sırasıyla hız ve t tekrardaki konumdur.

Eş. 2.12'den yararlanılarak hız değerlendirmesi yapma konusunda asıl sorun, hız değerinin bir kaç iterasyon ile çok büyük değerler alabilmesidir. Bu durum, özellikle parçacığın arama uzayında *Pbest* veya *Gbest*'den uzak olduğu zaman olabilmektedir. Büyük bir hız değeri, büyük bir yer değişimine ve böylece parçacığın uygun (feasible) çözüm alanın dışına çıkmasına neden olur. Bu durumu engellemek adına

çeşitli değişiklikler önerilmektedir. Bunlardan ikisi, parçacık hızlarının klemplenmesi ve hız hesaplamasında atalet ağırlığından yararlanılması şeklindedir.

2.3.2.1 Hız Klemplemesi

Parçacığın hızını klemplemek, parçacık hızının artarak çözüm uzayının dışına çıkmasını engelleyen bir yoldur. Hızın, önceden belirlenmiş maksimum değeri aşması durumunda söz konusu maksimum hız değerine göre ayarlanır (Eberhart vd., 1996). Eş. 2.15, parçacığın her bir boyutuna ilişkin hız değerinin nasıl değişebileceğini göstermektedir.

$$v_i(t+1)=\begin{cases} v_{imaks} & eğer\ v_i'(t+1) > v_{imax} \\ -v_{imaks} & eğer v_i'(t+1) < -v_{imax} \end{cases} \quad (2.15)$$

V_{imaks}, her bir boyuttaki maksimum hız olup v_i^I ise Eş. 2.12'den yararlanarak hesaplanmış *i* boyutundaki parçacığın hızıdır.

Hız klemplemesi, parçacığın kabul edilebilir alanın dışına çıkmasını engellemez. Ancak, parçacıkların adım büyüklüğünü kısıtlar. Büyük bir V_{imaks} değeri, ile global arama kapasitesi artar. Küçük bir V_{imaks} değeri ise sürünün lokal aramasını arttırır ancak parçacıklar, global aramada zayıf kalırlar. Sonuç olarak, hız klemplemesi ile birlikte parçacıkların, global optimuma yaklaşması biraz daha zorlaşır (Poli vd., 2007).

2.3.2.2 Atalet Ağırlığı

Aramayı ve hızı daha iyi kontrol etmek adına Shi ve Eberhart (1998), ω parametresini önermiştir. Yeni hız denklemi (Eş. 2.16) şu şekildedir:

$$v_i^{t+1}=\omega v_i^t+r_1c_1(Pbest_i^t-x_i^t)+ r_2c_2(Gbest^t-x_i^t) \quad (2.16)$$

Eş 2.12 ile karşılaştırıldığında tek değişiklik, ω parametresinin eklenmesidir. Atalet ağırlığı ω ve ivme katsayıları olan c_1 ve c_2, hız tanımlarında önemli bir rol oynamaktadır. Örneğin, $c_1 = c_2 = 0$ ise hız sadece ω faktörüne bağımlı kalır. Eğer $\omega > 1$ ise, hızın pozisyon değerlendirmesinde limitleri aşmasına neden olabilir. Diğer yandan ω değeri, parçacıkların arama uzayında daha geniş alanları taramasına sağlamaktadır.

ω<1 olması durumunda hız sıfır olma eğiliminde olur ve parçacıklar daha az hareket eder ancak lokal bir bölgeyi daha iyi bir şekilde tararlar. ω parametresine değer vermek için çeşitli stratejiler benimsenmektedir. Bansal vd. (2011), ω parametresi için sabit bir değer kullanmaktan tekrar sayısına bağlı olarak azalacak veya artacak şekilde belli bir aralıkta değer değişimine kadar farklılık gösteren stratejiler ile ilgili bir analiz yapmıştır.

PSO gelişiminin erken bir safhasında, parçacık hızı kısıtlanmadığı zaman bu değerlerin birkaç tekrar sonrasında kabul edilemez değerlere ulaştığı sonucuna varılmıştır. Bu olguyu anlamak üzere parçacık ekseninde pek çok çalışma yapılmıştır. Kennedy (1998), tek boyutlu stokastik ve rasgele parçacıklardan oluşan eksenin, ivme katsayıları (c_1 ve c_2) toplamının [0,4] arasında olduğunda düzenli bir davranış sergilediğini belirtmiştir. Özcan ve Mohan (1999) tarafından yapılan çalışmada ise parçacıkların arama uzayında sinüs biçimli eğrileri takip ederek hareket ettiği, rastgeleliğin ise sıklıklarını ve genliklerini değiştirdiği gösterilmiştir. Algoritmanın güçlü teorik bileşenlerinden biriyle gerçekleştirilen bir analiz de Clerc ve Kennedy (2002) tarafından yapılmış ve bunun sonucunda Eş. 2.17, PSO algoritmasına formüle edilmiştir.

$$v_i^{t+1}=\chi[v_i^t+r_1c_1(Pbest_i^t-x_i^t)+ r_2c_2(Gbest^t-x_i^t)] \quad (2.17)$$

i = 1,2, ...,N. Burada, χ parametresi, daralma katsayısı olup geri kalan parametreler ise yukarıda gösterilen modellerin aynı anlama gelen karşılıklarıdır. PSO'nun bu değişkeni, Eş. 2.16'da tanımlanan atalet katsayısı ile cebirsel olarak eşdeğerdir. Daralma katsayısı, Eş. 2.18 ile tanımlanmaktadır.

$$\chi=\frac{2}{2-\phi-\sqrt{\phi^2-4\phi}} \quad (2.18)$$

Burada $\phi = c_1 + c_2$ ve $\phi > 4$. Buna göre $\chi = 0.7298$ ve $c_1 = c_2 = 2.05$ eşitsizlikleri, daralma katsayılı PSO için varsayılan değer olarak kabul edilir. Daralma katsayılı PSO, atalet katsayılı PSO ile cebirsel olarak eşit olduğundan varsayılan ω = 0.7298 ve $c_1 = c_2 = 1.49618$ değerleri, atalet katsayılı bir PSO'da yaygın olarak görülür.

2.3.2.3. Lokal En İyi ve Global En İyi

Başlangıçta *Pbest*, parçacığın ilk uygunluk değeridir ve *Gbest* ise, sürüde bulunan en iyi uygunluk değerine sahip parçacık olarak değerlendirilir. Parçacığın *Pbest* değeri; parçacık uygunluk değerinin, parçacığın *Pbest* değerinden daha iyi olduğu durumlarda güncellenir.

Gbest değer güncellemesi, çeşitli algoritma özelliklerine bağlı olabilir. Senkronize bir güncellemede *Gbest* değeri, bütün parçacıkların konumları güncellendiği zaman güncellenir. Bir parçacığın uygunluk degeri, *Gbest* değerinden daha iyi olması durumunda *Gbest* bu parçacık ile değiştirilir. Bu da, tek bir tekrar ile gerçekleştirilir. Senkronize olmayan güncellemelerde yeni *Gbest* konumu, her bir parçacığın konumu ve *Pbest* güncellemesi yapıldıktan sonra hesaplanır. Bu da her bir iterasyonda birden fazla tekrarla gerçekleştirilir.

Gbest, sürüde kullanılan komşu topolojisine bağlıdır. Komşu topolojisi, parçacıklar arası iletişimi sağlama yoludur. PSO için farklı topolojiler geliştirilmiştir (Kennedy ve Mendes, 2002; Poli vd., 2007).

Tam bağlantılı topoloji, tüm parçacıkları birbirini bağlar. Bu durumda her parçacık, birbiriyle iletişim kurar. Bu durumda her bir parçacığın komşusu bütün sürüdür. Böylece *Gbest* ile ilgili bilgiler, tüm parçacıklara hızlı bir şekilde iletilir ve yakınsama hızı daha yüksek olur. Ancak global en iyinin, *Gbest* konumuna yakın olmaması durumunda PSO, lokal bir optimum noktada hapsolmuş olabilir (Kennedy ve Mendes, 2002).

Halka topolojisi, her bir parçacığın doğrudan belli sayıda komşusuyla iletişim kurduğu tam bağlantılı topolojiye göre daha az bağlantılıdır. Bu durumda sadece tek bir *Gbest* yoktur, birden fazla komşunun en iyisi olan *nbest* vardır. Halka topolojisinde, tam bağlantılı topolojiye oranla daha geniş bir arama uzayı söz konusudur ve daha çok çekim noktası *nbest* olduğu için lokal bir optimum noktada hapsolmaya daha az meyillidir (Kennedy ve Mendes, 2002).

2.3.3. Konum

Hız hesaplamasından sonra parçacık konumu, Eş. 2.13'ten yararlanılarak güncellenir. Bu durumda parçacık arama uzayının dışında çıkabilir. Pek çok optimizasyon probleminde kısıtlama söz konusu olduğu için parçacığın arama uzayının dışına çıkmasını engellemek önemlidir. Kısıtlama ihlallerinin önüne geçmek amacıyla farklı yaklaşımlardan yararlanılmaktadır. Robinson ve Rahmat-Samii (2004), bu tür ihlallerin önüne geçmek için geniş çapta yararlanılan üç yaklaşım önermiştir:

1. Emilim: Bir parçacığın bir boyutta arama uzayının dışına çıkması durumunda konumu o boyutta sınır değere göre ayarlanır ve o boyuttaki hız bileşeni sıfıra ayarlanır.

2. Yansıtma: Bir parçacığın bir boyutta arama uzayının dışına çıkması durumunda konumu o boyutta sınır değere göre ayarlanır ve o boyuttaki hız bileşeninin işareti ters çevrilir.

3. Sönüm: Bir parçacığın bir boyutta arama uzayının dışına çıkması durumunda, konumu o boyutta sınır değere göre ayarlanır ve o boyuttaki hız bileşeninin işareti ters çevrilir ve $[0, 1]$ arasında düzgün dağılımlı rassal bir sayı ile çarpılır.

Engelbrecht'e (2007) göre iki başka seçenek daha bulunmaktadır:

1. Onarım: Bir parçacığın bir boyutta arama uzayının dışına çıkması durumunda konumu, o boyutun değeri *Pbest* veya *Gbest* ile eşitlenir ve o boyuttaki hız bileşeni sıfıra ayarlanır.

2. Pbest seçimi: Parçacıkların, sınırları aşmasına izin verilir ancak parçacığın konumu arama uzayının içinde değilse konumu parçacığın *Pbest*'i olamaz.

2.4. Literatür Araştırması

Bu bölümde, tez çalışmasında ele alınan sıra bağımlı hazırlık zamanlı GMHDP ve çözüm yöntemi olarak kullanılan diferansiyel gelişim algoritması (DGA) ve parçacık sürü optimizasyonu (PSO) konuları ile ilgili literatür araştırmalarına ayrı alt başlıklarda yer verilmiştir.

2.4.1. Sıra Bağımlı Hazırlık Zamanlı GMHDP ile ilgili Literatür Araştırması

Andres vd. (2008), çalışmalarında ilk defa sıra-bağımlı hazırlık zamanlı tip-1 GMHDP için bir tamsayılı doğrusal programlama modeli sunmuşlar ve çözüm için de 8 farklı sezgisel ve GRASP algoritması geliştirmişlerdir.

Özcan ve Toklu (2010), sıra-bağımlı hazırlık zamanlı çift taraflı GMHDP için karma tamsayılı bir doğrusal programlama modeli sunmuşlardır. Ayrıca modelin çözümü için de COMSOAL tabanlı bir sezgisel geliştirmişlerdir.

Seyed vd. (2011), sıra-bağımlı hazırlık zamanlı tip-2 GMDHP'nin çözümü için bir tavlama benzetimi algoritması geliştirmişler ve parametre optimizasyonu için de Taguchi yöntemini kullanmışlardır.

Yolmeh ve Kianfar (2012), sıra-bağımlı hazırlık zamanlı tip-2 GMHDP için hibrit bir genetik algoritma geliştirmişlerdir. Buldukları hibrit algoritmayı literatürdeki diğer algoritmalar ile karşılaştırmışlar ve daha iyi sonuçlar bulmuşlardır.

Scholl vd. (2013) çalışmalarında, Andres vd. (2008) yaptıkları çalışmaya ek olarak hem geri (backward), hem de ileri (forward) hazırlık zamanlarını kullanmışlardır. Ayrıca yeni bir tamsayılı model kurmuşlar ve model çözümü için de çeşitli sezgisel algoritmalar önermişlerdir. Bu sezgisel algoritmalar geçmiş sezgisel çözümlerden daha iyi sonuçlar elde etmesine rağmen hazırlık zamanlarının oransal olarak büyük olduğu test problemlerinde çok iyi sonuçlar bulamamışlardır.

Hamta vd. (2013), sıra-bağımlı hazırlık zamanlı tip-2 GMDHP için çok amaçlı bir matematiksel model geliştirmişlerdir. Bu modelin amaçları çevrim süresini, malzeme maliyetlerini ve istasyonlar arasındaki düzgünlük indeksini minimize etmektir. Problem NP-zor olduğu için büyük problemler için matematiksel modellerle makul çözümler bulunması çok zor olmaktadır. Bu nedenle, çözüm için değişken komşu arama sezgiseli ile parçacık sürü metasezgiseliyle hibritize edilen bir algoritma sunmuşlardır. Literatürdeki diğer yöntemlerle karşılaştırıldığında geliştirilen algoritma hem optimal çözüme yakın çözümler bulma kabiliyeti, hem de çözüme ulaşma süresi bakımından daha iyi sonuçlar vermiştir.

Akpınar vd. (2014), karışık modelli hazırlık zamanlı GMHDP için karma tamsayılı doğrusal programlama modeli geliştirmiş ve problemin karmaşık olmasından dolayı çözüm için karınca algoritması ve genetik algoritmayı hibritize ederek bir çözüm algoritması geliştirmişlerdir.

Diri vd. (2015), stokastik sıra-bağımlı hazırlık zamanlı GMHDP için doğrusal olmayan karma tamsayılı programlama modeli sunmuşlardır.

Janardhanan vd. (2018), sıra-bağımlı hazırlık zamanlı robotik tip-2 GMHDP için karma tamsayılı doğrusal programlama modeli geliştirmişlerdir. Problem yapısı NP-zor olduğundan dolayı çözüm için göç eden kuşlar meta-sezgiselini kullanmışlardır. Söz konusu meta-sezgiseli literatürdeki diğer meta-sezgiseller ile karşılaştırmışlardır. Bulunan çözümlerde göç eden kuşlar algoritmasının diğerlerine göre daha iyi sonuçlar verdiği gözlenmiştir.

Güner (2019), görevler arasında sıra bağımlı hazırlık zamanlarının bulunduğu, birincil amacın hatta kullanılan işçi sayısının, ikincil amacın ise hatta kullanılan istasyon sayısının en küçüklendiği paralel çok işçili montaj hatlarının dengelenmesi için çözüm yaklaşımları sunmuştur. Bu çalışmada Pİ-MHDP/SB için kesin çözüm metodu olarak iki farklı karma tam sayılı matematiksel model (ÖMAT-1 ve ÖMAT-2), kısıt programlama modelleri (KP1 ve KP2) ve mantık tabanlı Benders ayrıştırması algoritması (ÖMTBA) geliştirilmiştir. Büyük boyutlu problem setleri için meta sezgisel algoritmalardan yığın tabanlı arama yapan bir Genetik Algoritma (GA) ve tek noktadan arama yapan bir Tavlama Benzetimi Algoritması (TB) geliştirmiştir.

Gutjahr ve Nemhauser (1964), basit montaj hattı dengeleme problemlerinin bile NP-zor yapıda olduğunu ispatlamışlardır. Dolayısıyla çok daha karmaşık olan sıra-bağımlı hazırlık zamanlı GMHDP'nin de NP-zor olduğunu söyleyebiliriz. NP-zor yapıdaki problemlerin doğrusal programlama, dinamik programlama ve dal sınır algoritmaları gibi kesin yöntemlerle makul zamanlarda en iyi çözümlerinin bulunması büyük problemler için çok zordur. Bu nedenle bu tip problemlerin çözümü için sezgisel veya metasezgisel algoritmalar kullanılmaktadır. Bu çalışmada da çözüm için yeni bir diferansiyel gelişim algoritması ve diferansiyel gelişim algoritması-parçacık sürü optimizasyonu yöntemlerinin hibritize edilmesiyle yeni bir hibrit algoritma geliştirilmiştir. Aşağıda diferansiyel gelişim

algoritması ve parçacık sürü optimizasyonu yöntemlerinin MHDP üzerinde daha önce yapılan çalışmaları ile ilgili bir literatür taraması yapılmıştır.

2.4.2. Diferansiyel Gelişim Algoritması (DGA) ile ilgili Literatür Araştırması

Diferansiyel gelişim algoritması (DGA) ilk defa Storn ve Price (1997) tarafından geliştirilmiş ve sürekli en iyileme problemlerin çözümünde kullanılmıştır. DGA günümüze gelinceye kadar birçok en iyileme problemlerin çözümünde kullanılmıştır (Cheng ve Hwang, 2001; Ali ve Torn, 2004; Kaelo ve Ali, 2005; Sun vd., 2005; Montes vd., 2010). DGA aynı zamanda birçok MHDP'nin çözümünde de kullanılmıştır.

Nearchou (2007), ilk defa BMHDP için DGA yöntemini kullanmıştır. Bu çalışmada, hattın çevrim zamanını en aza indirmek amacıyla BMHDP'yi çözmek için DGA kullanılmıştır. Çözümlerinin kodlanmasına yönelik iki farklı yöntemle (rassal anahtar ve öncelik temelli kodlama) birlikte açık literatürden test problemleri ile deneyler yapılmıştır. Daha önce yayımlanmış diğer evrimsel algoritmaları ile yapılan karşılaştırmalar, DGA'nın daha üstün bir performansa sahip olduğunu göstermiştir.

Nearchou (2008), tek modelli deterministik MHDP'yi çözmek üzere çok amaçlı DGA modelini önermiştir. Ayrıca çok amaçlı DGA'yı MHDP'ye uyarlayan özel temsil ve kodlama modelleri geliştirilmiştir. Çok amaçlı DGA, ağırlıklı toplam Pareto genetik algoritması (GA) ve Pareto-nişli GA yöntemleri ile literatürde bilinen test problemleri üzerinden karşılaştırılmıştır. Deneysel karşılaştırmalar, çok amaçlı DGA yaklaşımının yüksek nitelikli bir performansı olduğunu göstermiştir.

Nourmohammadi ve Zandieh (2011) ise çok amaçlı BMHDP-2'ye çözüm olarak çok amaçlı bir DGA önermiştir. Mevcut algoritmaların aksine Pareto baskınlık konseptine dayalı yeni bir kabul modeli ve TOPSIS'e dayalı yeni bir gelişim modeli önererek popülasyonun gelecekteki üyelerinin seçiminde hedefleri ayrı ayrı ele alan bir model geliştirmişlerdir. Ayrıca Taguchi yönteminden yararlanarak, geliştirilen algoritmanın parametrelerini optimize etmişlerdir. Hesaplamalı deneyler, geliştirilen algoritmanın pek çok ölçüt bakımından mevcut metasezgisel yöntemlerden daha iyi performansı olduğunu göstermiştir.

Mozdgir vd. (2013), BMHDP-2'de iş yükü düzgünlük endeksini en aza indirmek için bir DGA geliştirmiştir. Bu DGA'nın ana avantajı, arama çeşitliliğini sağlamak ve aynı zamanda doğrudan amaç fonksiyonundan gelen bilgilerle arama etkinliğini arttırmak için basit ama etkili mutasyon sürecinden kaynaklanmaktadır. Ayrıca, algoritma parametreleri Taguchi yöntemi kullanılarak optimize edilmiştir. Önerilen algoritmanın etkinliğinin kanıtlanması için bir sezgisel yöntemle karşılaştırılmıştır. Bu karşılaştırma önerilen algoritmanın daha etkin olduğunu göstermiştir.

Vincent ve Ponnambalam (2013), Esnek Montaj Hattı (EMH) çizelgeleme ve dengeleme problemini çözmek için iki seviyeli DGA önermiştir. İki seviyeli DGA, EMH'nin performansını iki kritere göre optimize eder: Erken/Gecikme cezalarının ağırlıklı toplamı ve EMH'nin dengesi. İki seviyeli DGA'nın performansı, literatürde mevcut olan veri setleri ve daha önce yayınlanan iki seviyeli genetik algoritma kullanılarak değerlendirilmiştir. Deneysel sonuçlar iki seviyeli DGA'nın EMH problemini etkili bir şekilde çözebileceğini ve iki seviyeli genetik algoritma'ya göre üstün bir performans sergilediğini göstermektedir.

Pitakaso ve Sethanan (2015), BMHDP-1 ve BMHDP-1'in bir iş istasyonundaki maksimum sayıda makine tipi dikkate alındığı hali olan BMHDP-1M çözümü için DGA ve modifiye diferansiyel gelişim algoritmasını (MDGA) önermiştir. Önerilen algoritmalar BMHDP-1 için standart test örnekleri ve BMHDP-1M için ise vaka çalışmaları kullanılarak değerlendirilmiştir. DGA ve MDGA'nın BMHDP-1 için en iyi çözüm yöntemleri arasında olduğu ve BMHDP-1M'nin çözülmesinde uygulanabilir olduğu sonucuna varılabilir. MDGA, BMHDP-1 için daha az hesaplama süresi optimal sonuçlar bulabilmekte ve BMHDP-1M için ise daha iyi çözümler üretebilmektedir.

Zhang vd. (2016), BMHDP-2'yi çözmek için yeni bir mutasyon operatörü ve çaprazlama operatörü içeren tamsayı kodlamalı yeni bir DGA (TDGA) önermişlerdir. Önerilen çaprazlama operatörü, popülasyondaki tüm bireylerin BMHDP-2 için uygun ve tamsayı değerli çözümler üretmektedir. TDGA'nın performansı literatürdeki diğer çalışmalarla, yani Nearchou (2007), Kim ve ark. (1996), Goncalves ve Almeida (2002) tarafından geliştirilen gerçek kodlu DGA ile karşılaştırılmıştır. Deneysel süreç, TDGA'nın diğer birçok kombinatoryal problemlerin çözümünde kullanılabileceğini göstermiştir.

Janardhanan vd. (2016), montaj hattı maliyetini ve çevrim zamanını en aza indirgemek amacıyla maliyete dayalı robotik montaj hattı dengeleme (RMHD) problemi tanumlamıştır. Düz ve U-tipi RMHD problemini çözmek için DGA kullanılmıştır. 32 problem setinin performansının deneysel değerlendirmesinden, U-tipi montaj hatlarının düz robotik montaj hatlarına kıyasla hem maliyet hem de zaman açısından daha verimli olduğu görülmüştür.

Nearchou ve Omirou (2017), hem tip-1 hem de tip-2 BMHDP için yeni bir DGA geliştirmişlerdir. Çözüm vektörleri öncelik tabanlı kodlama ve rastgele kodlama şeklinde kodlanmıştır. Literatürdeki test problemleri üzerinden farklı metasezgisellerle karşılaştırılan DGA'nın rastgele kodlamalı olan çözüm yöntemi üstünlük sağlamıştır.

Gansterer ve Hartl (2017), tek ve çift taraflı montaj hattı dengeleme problemlerinin çeşitli kısıtlar altında çözülmesi için genetik algoritma, tabu arama ve DGA metasezgisellerini kullanmışlardır. Tekstil sektöründen gerçek verilerin kullanıldığı uygulamada 52 görevli örneklere kadar her üç metasezgisel de en iyi çözümlere ulaşabilmiştir. Ancak daha büyük problemlerde genetik algoritmanın, hem tabu arama metasezgiselinden hem de DGA'dan daha iyi sonuçlar verdiği gözlenmiştir.

Özçelik (2018), düz ve U-tipi montaj hatları için bir DGA geliştirmiştir. Önerilen algoritma literatürdeki diğer algoritmalar ile karşılaştılmış ve daha iyi sonuçlar verdiği gözlenmiştir. Önerilen algoritmanın üstünlüğü kromozom yapısından kaynaklanmaktadır. Kromozom yapısı öncelik ilişkilerine öncelik vermekte olup, herhangi bir tamir operasyonuna gerek duymamaktadır.

Literatür incelendiğinde DGA'nın, kesikli ve permütasyonel olan MHDP'de çok fazla bir kullanımı olmadığı görülmektedir. DGA'nın kullanıldığı çalışmalarda, öncelikle kodlama kesikli problemlere uygun bir yapıya göre yapılmıştır. Problemin öncül-ardıl kısıtlarından dolayı yeni mutasyon ve çaprazlama operatörleri geliştirilmiştir. Bazı çalışmalarda öncül-ardıl ilişkisini sağlamak için tamir operatörleri kullanılmıştır. Çalışmaların çoğunda DGA başarılı bir şekilde probleme uyarlanmış ve etkin sonuçlara ulaşılmıştır.

2.4.3. Parçacık Sürü Optimizasyonu (PSO) ile ilgili Literatür Araştırması

PSO, Kennedy ve Eberhart (1995) tarafından önerilen popülasyon tabanlı bir stokastik optimizasyon tekniğidir. Kennedy ve Eberhart (1997) kombinatoryal optimizasyon için PSO'nun alternatif bir versiyonunu ortaya koymuştur. Daha sonra bu durum, Shi ve Eberhart (1998, 1999) ve Kennedy vd. (2001) tarafından ele alınmıştır. Yakın zamanda, PSO araç rotalama (Belmecheri vd., 2010), görev ataması (Salman vd., 2002), akış hattı çizelgeleme (Chakaravarthy vd., 2013), en kısa yol problemleri (Mohemmed vd., 2008) ve atölye çizelgeleme (Aitzai vd., 2014; Qiu ve Lau, 2014) gibi farklı uygulamalara başarıyla uygulanmıştır. PSO aynı zamanda birçok MHDP'nin çözümünde de kullanılmıştır.

Nearchou (2011), BMHDP için PSO yöntemini ilk kez kullanmıştır. Bu makalede, çok amaçlı BMHDP'nin çözümü için PSO yaklaşımının kullanımı incelenmiştir. Bunun için iki optimizasyon kriteri göz önünde bulundurulmuştur: hattın çevrim zamanını en aza indirmek ve iş istasyonları arasındaki iş yükünün düzgünlüğünü maksimize etmek. PSO algoritmalarının farklı versiyonları ve mevcut iki karakteristik çok amaçlı genetik algoritma ile karşılaştırılmıştır. Elde edilen sonuçlar, üretilen PSO algoritmasının MHDP çözümleri açısından oldukça umut verici olduğunu ve yüksek bir performans sergilediğini göstermiştir.

Petropoulos ve Nearchou (2011), tek modelli montaj hatlarını dengelemek için PSO algoritması geliştirmiştir. Hattın çevrim zamanı, iş istasyonları arasındaki iş yükü dengesi ve hattın denge gecikme süresi kullanılarak iki ve üç kriterli problemler için PSO algoritmasının çeşitli versiyonları geliştirilmiştir. Yapay parçacıkların (PSO tarafından geliştirilen potansiyel çözümler) gerçek MHDP'nin çözümleriyle eşleştirilmesi için bir kodlama şeması getirilmiştir. Elde edilen sonuçlar, üretilen PSO algoritmasının MHDP çözümleri açısından yüksek performans sergilediğini göstermiştir.

Chutima ve Chimklai (2012), çok amaçlı çift taraflı karma modelli MHDP'yi çözmek için negatif bilgiye sahip bir PSO algoritması (PSONB) geliştirmişlerdir. Sundukları bu algoritma, farklı parçacıkların göreceli konum bilgisini kullanan negatif bilgisi olan yeni bir PSO algoritmasıdır. Negatif çözüm bilgisi, bitişik görev çiftlerinin bir sonraki nesilde yeni

çözüm dizilerinin bir parçası olarak seçilmesini önlemek için de kullanılır. Deneysel sonuçlar, PSONB'nin rekabetçi ve gelecek vaat eden bir algoritma olduğunu göstermiştir.

Hamta vd. (2013), görevlerin çalışma sürelerinin bilinmeyen değişkenler ve bilinen tek bilginin her görevin çalışma süresi için alt ve üst sınırlar olduğu çok amaçlı MHDP için PSO algoritması ile değişken komşu arama kombinasyonuna dayanan yeni bir çözüm yöntemi önermiştir. Gerçek endüstriyel koşulları yeterince yansıtmak için, görev süresinin aynı veya benzer etkinlik için çalışan makine ve operatörler için öğrenme süreciyle değişebildiği ve sıra bağımlı hazırlık zamanlı olduğu varsayılmıştır.

Delice vd. (2014), karışık modelli çift taraflı MHDP'yi çözmek için negatif bilgiye sahip yeni bir PSO algoritması geliştirmiştir. Önerilen yaklaşım, birleşik seçim mekanizması ve kod çözme prosedürüne dayanan yeni prosedürleri içerir. Deneysel sonuçlar, önerilen yaklaşımın her test problemi için kısa bir hesaplama süresi içinde iyi çözümler elde ettiğini göstermektedir.

Janardhanan vd. (2015), çevrim zamanını ve toplam enerji kullanımını eş zamanlı olarak minimuma indirmek için, biri çevrim zamanını optimize etmeye odaklanacak model (zaman bazlı model) ve diğeri de toplam enerji kullanımını optimize etmeye odaklanacak model (enerji bazlı model) olmak üzere çift odak noktası bulunan modeller önermişlerdir. PSO, bu problemi çözmek için kullanılmıştır. Literatürdeki test problemleri üzerinde yapılan deneylerde önerilen modellerin, robotik montaj hatlarındaki toplam enerji kullanımını ve çevrim zamanını azalttığı bulunmuştur.

Janardhanan vd. (2015), çevrim süresini minimuma indirme amacıyla robotik montaj hattını dengelemek için PSO algoritması ve hibrit bir guguk kuşu arama algoritması (GKA) ve PSO önermiştir. Önerilen PSO ve hibrit GKA-PSO performansı literatürdeki 32 test problemi kullanılarak değerlendirilmiştir. Analiz aracılığıyla elde edilen sonuçlar hibrit GKA-PSO algoritmasının PSO'ya ve literatürde verilen hibrit GA'ya kıyasla daha iyi performans verdiğini göstermektedir.

Janardhanan vd. (2016) U-tipi robotik montaj hatlarındaki (URMH) total enerji kullanımını minimuma indirerek hat verimliliğini maksimuma çıkarmak amacıyla PSO ve DGA yöntemlerini kullanmıştır. Önerilen algoritmaların, montaj hattı üzerindeki toplam

enerji kullanımını azaltma konusunda etkili olduğu görülmüştür. DGA'nın, PSO algoritmasına kıyasla hat verimliliğini daha çok arttırdığı sonucuna ulaşılmıştır.

Delice vd. (2016) çift-taraflı U-tipi MHDP'yi çözmek için PSO'ya dayalı yeni bir algoritma önermiştir. Bu önerilen yaklaşımda birincil amaç; verilen bir çevrim süresi için istasyon sayısını ve ikincil amaç olarak pozisyon sayısını en aza indirmektedir. Literatürden alınan test problemleri önerilen algoritmanın verimliliğini göstermek amacıyla çözülmüştür. Deneyden elde edilen sonuçlar önerilen yaklaşımın tüm test problemleri için iyi çözümler elde ettiğini göstermektedir.

Che (2016), hem montaj sırası planlama ve hem de montaj hattı dengelemeyi göz önünde bulunduran bir model önermiştir. Çok amaçlı PSO ve kılavuz arama algoritmasını birleştiren yeni hibrit bir algoritma, modeli çözmek için geliştirilmiştir. Bir bilgisayar toplama fabrikasından alınan gerçek bir örnekten alınan veriler modelin performansını ölçmek için kullanılmıştır. Analiz sonuçları, önerilen algoritmanın etkin çözümleri saptamakla kalmayıp aynı zamanda daha yüksek Pareto-optimal çözüm oranlarına ulaştığını göstermektedir.

Rabbani vd. (2016), robot alım masraflarını, robot kurulum masraflarını, sıraya bağlı kurulum masraflarını ve tip 2 robotik karışık modelli U-tipi montaj hattı dengelemenin çevrim süresini azaltmaya yönelik baskılanamayan sıralamalı genetik algoritma-II (BSGA-II) ve çok amaçlı PSO olmak üzere iki farklı çok amaçlı evrimsel algoritma geliştirmiştir. Elde edilen sonuçlar, tüm problemler çeşitlerinde Pareto-optimal sayısı ve ortalama ideal uzaklık açısından BSGA-II'nin verimliliğinin daha iyi olduğunu göstermiştir.

Yuguang ve Yong (2016), gövde montaj hattını (GMH) dengelemek için geliştirilmiş kesikli parçacık sürü optimizasyonu (GKPSO) adlı yeni bir çok amaçlı evrimsel algoritma geliştirmiştir. Gövde montaj hattı (GMH) farklı gövde bloklarının aynı anda monte edilebildiği karışık modelli bir montaj hattıdır. Karşılaştırmalı sonuçlar, GKPSO performansının daha yüksek olduğunu göstermiştir.

Dou vd. (2017), basit montaj hattı dengeleme problemlerini (BMHDP) çözmek için uygulanabilir sıra odaklı yeni bir kesikli parçacık sürü optimizasyonu (KPSO) algoritması geliştirmiştir. KPSO'da, görev odaklı gösterim tip 1 ve tip 2 BMHDP'leri çözmek için uyarlanmıştır ve bir parçacık, bir uygulanabilir görev sırasını permütasyon olarak temsil etmektedir. KPSO, var olan gerçek-şifrelenmiş PSO (GPSO) ve doğrudan kesikli PSO

(DKPSO) arasındaki kıyaslamaların sonuçları, önerilen KPSO performansının daha yüksek olduğunu göstermektedir.

Yang vd. (2017), karışık akışlı montaj hattında görülen düzensiz sıralama problemini çözmek için kesikli parçacık sürü optimizasyonu (KPSO) geliştirmiştir. Ayrıca, montaj hattındaki anlık yük ve ortalama yükü optimize etmek için değiştirilmiş bir genetik algoritma önerilmiştir. Sıralama analizinin sonuçları, karışık modelli montaj hattının sıralamasına yönelik çözümde KPSO algoritmasının kullanılmasıyla düzensiz ve verimsiz çok amaçlı sıralama probleminin etkin bir şekilde çözülebileceğini göstermektedir.

Dong vd. (2018), esnek görev zamanları ve bölgeleme kısıtları olan stokastik montaj hattı dengeleme probleminin ekipman masrafını ve çevrim süresini eş zamanlı olarak en aza indirmek için iki amaçlı bir rassal kısıtlı karışık 0-1 programlama modeli geliştirmiştir. Ayrıca, yerel arama stratejisi olarak benzetilmiş tavlama algoritmasını kullanan ve Pareto-optimal çözümleri aramak için hibrit bir PSO önerilmiştir. Taguchi yöntemi, parametrelerin etkisini incelemek amacıyla kullanılmıştır ve buna göre uygun bir parametre ayarı önerilmiştir. Karşılaştırmalı sonuçlar, önerilen algoritmanın aynı sürede daha iyi çözümler elde ederek var olan algoritmalardan daha iyi performans verdiğini göstermektedir.

Rabbani vd. (2018), ilki sıradan bir hat olan ikincisi de daha modern teknoloji ve daha yüksek becerilere sahip operatörlerden dolayı ekspres bir hat olan karışık modelli iki paralel montaj hattını dengelemek için baskılanamayan sıralamalı genetik algoritma-II (BSGA-II) ve çok amaçlı parçacık sürü optimizasyonu (ÇAPSO) önermiştir. Ekipman masrafı ve nitelikli işgücü ekspres hatta daha yüksektir. Ayrıca; görev ve kurulum zamanlarında operatörlerin öğrenme etkisi, göz önünde bulundurulmaktadır. BSGA-II ve ÇAPSO yöntemlerini karşılaştırmak için, küçük ve büyük ölçekli problemlerden oluşan sekiz sayısal örnek uygulanmış ve elde edilen sonuçlar ÇAPSO'nun BSGA-II modeline kıyasla daha iyi performans verdiğini göstermiştir.

Ab Rashid vd. (2019), eş zamanlı olarak çok amaçlı kesikli parçacık sürü optimizasyonunu (ÇAKPSO) kullanarak, bütünleşik karışık modelli montaj sıra planlama (MSP) ve MHD problemlerini çözmüşlerdir. Bu araştırma, karışık modelli hat problemleri için MSP ve MHD problemlerini bütünleşik olarak modelleyip optimize eden yayınlanmış ilk araştırmadır. ÇAKPSO performansını test etmek için, farklı zorluk seviyelerindeki 51 test

problemi kullanılmıştır. Bu deneyden elde edilen sonuçlara göre, çözüm kalitesi açısından ÇAKPSO'nun çok iyi performans gösterdiği görülmüştür.

Şahin ve Kellegöz (2019), birden fazla insanın çalıştığı paralel iş istasyonlarının olduğu bir MHDP için yeni bir karışık tamsayılı doğrusal programlama önermiştir. Modelin amacı, iş istasyonlarının ve diğer kaynakların maliyetlerini azaltmak için görevlerin dağılımını yapmaktır. Ayrıca, daha büyük ölçekli problemleri çözmek için PSO bir özel kurucu sezgiselle birleştirilerek yeni hibrit bir yöntem geliştirilmiştir. Literatürde çeşitli hat dengeleme problemlerini çözmek için yaygın olarak kullanılan pek çok öncelik diyagramı dikkate alınarak bu problem tipi için test problemleri oluşturulmuştur. Sonuç olarak önerilen hibrit metasezgisel alt sınırlardan kabul edilebilir düzeyde sapmalar göstererek etkinliğini kanıtlamıştır.

Literatür incelendiğinde MHDP'de yeterli sayıda PSO uygulaması yapıldığı söylenebilir. Bu çalışmalarda PSO kullanılmasının sebeblerinden bazıları şunlardır: Kompleks bir algoritma olmaması, kolay kodlanabilmesi ve hızlı yakınsama özelliği vb. Yerel en iyilere takılmasından dolayı global arama yeteneğinin artırılması için bazı çalışmalarda başka metasezgisellerle hibritize edilmiştir. DGA'da olduğu gibi sürekli problemlerin çözümü için geliştirilmiştir. Dolayısıyla, kesikli olan MHDP problemlerine uygulanabilmesi için çeşitli kodlama yöntemleri kullanılmıştır. Yer ve hız formülleri kesikli problem yapısına uygun olarak geliştirilmiştir. Çalışmaların çoğunda PSO etkin sonuçlar vermiştir.

ÜÇÜNCÜ BÖLÜM

YÖNTEM

Bu bölümde sıra-bağımlı hazırlık zamanlı GMHDP'nin çözümü için geliştirilen iki metasezgisel yöntem detaylı açıklanmıştır. Bunlar:

1. Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA)
2. Geliştirilmiş Diferansiyel Gelişim Algoritması- Parçacık Sürü Optimizayonu Hibrit Algoritması (HA)

3.1. Geliştirilmiş Diferansiyel Gelişim Algoritması

Bu çalışmada bazı nedenlerden dolayı DGA yöntemi kullanılmıştır. Birincisi; sürekli problemlere başarıyla uygulanan DGA'nın, kesikli ve permütasyonel problemlere uygulanması literatürde çok fazla rastlanılmamaktadır. İkincisi; DGA'nın kolayca kodlanabilmesi özelliğidir. Diger algoritmalar için binlerle ifade edilen satırdan oluşan kodlar söz konusu iken DGA için yaklasık 20 satırlık kod yeterli olmaktadır (Mayer vd., 2005).

Bu bölümde sıra-bağımlı hazırlık zamanlı GMHDP'nin çözümü için Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA) açıklanmıştır. GDGA, 6 bölümden oluşmaktadır. Bunlar:

1. Başlangıç popülasyonun oluşturulması
2. Mutasyon
3. Çaprazlama
4. Tamir Operatörü
5. Uygunluk Fonksiyonu
6. Seçilim

3.1.1. Başlangıç Populasyonunun Oluşturulması

Sıra-bağımlı hazırlık zamanlı GMHDP, kesikli ve permütasyonel olduğu için GDGA'da çözüm vektörleri gerçel sayılardan değil kesikli sayılardan oluşmaktadır. Her bir çözüm vektörü D (görev sayısı) büyüklüğündeki vektörden oluşmaktadır. Ayrıca başlangıç popülasyonu sayısı N_p kadar olmaktadır. Burada populasyon sayısı N_p kullanıcı tarafından belirlenmektedir. N_p sayısı büyüdükçe modelin en iyi çözümü bulma yeteneği artmaktadır. Ancak aynı zamanda çözüm bulma süresi de artmaktadır. Başlangıç popülasyonundaki her bir çözüm vektörü, 1'den D'ye kadar ve birbirinden farklı tamsayılardan oluşan D büyüklüğünde bir vektördür. Örneğin, montaj hattının 8 görevden oluştuğunu ve popülasyon sayısının 20 olarak belirlendiğini varsayalım. Popülasyon sayısı 20 olduğu için başlangıç popülasyonu 20 adet 8 elemanlı ve elemanları birbirinden farklı çözüm vektörlerinden oluşur.

3.1.2. Mutasyon

Sıra-bağımlı hazırlık zamanlı GMHDP'ye klasik DGA mutasyon operatörü uygulanamaz. GDGA'da klasik DGA'nın aksine başlangıç populasyondaki çözüm vektörleri sürekli değil kesikli sayılardan oluşmaktadır. Dolayısıyla GDGA'da mutasyon operatörü için yeni bir bakış açısı gerekmekteydi. Bu yüzden bu çalışmada mutasyon operasyonu için yer değiştirme (swap) işlemi uygulanmıştır.

Bu çalışmadaki mutasyon operasyonu için geliştirilen yer değiştirme (swap) algoritması için üç birbirinden farklı vektör S_{1i}, S_{2i}, S_{3i} seçilmektedir. Burada klasik DGA'daki gibi iki vektörün birbirinden farkı bulunmaz. Vektörlerin birbirinden farkının bulunmamasının sebebi, negatif değerlerin çıkmasını engellemektir. Çünkü negatif sonuçların çıkması çözüm vektörünü uygunsuz bir çözüm yapar. Örnek olarak, Tablo 3'te verilen vektörlerin $S_{2i} - S_{3i}$ işleminin sonucu [0 1 0 -1 0 0 2 -2] çıkacaktır. Negatif bir görev sayısı olmadığı için bu işlem sonucu uygunsuz (infeasible) olacaktır. Bu yüzden GDGA'da mutasyon işlemi için bir yer değiştirme prosedürü uygulanmıştır. Bu prosedüre göre seçilen üç vektörden ikisinin her bir elemanın birbirine eşit olup olmadığına bakılır. Aynı ise 0, değilse 1 değeri verilir (Eş. 3.1). Ayrıca GDGA'da F katsayısı da kullanılmamaktadır. Sadece

değerlerin birbirine eşit olup olmadığına bakıldığı için F katsayısının kullanımı gereksiz olmaktadır.

$$x_i = S_{2i} - S_{3i} \quad \forall i \in D \tag{3.1}$$

$$Y_i = \begin{cases} if(x_i == 0 & 0 \\ aksi\ takdirde & 1 \end{cases}$$

Daha sonra vektörler üzerinde aşağıda Tablo 3'te gösterilen yer değiştirme işlemi uygulanır.

Tablo 3. Yer Değiştirme İşlemi

				S_{1i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	3	4	5	6	7	8
				S_{2i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	3	4	2	5	6	8	7
				S_{3i}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	4	3	5	7	8	6
				Y_i				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	0	1	0	1	0	1	0	1

Tablo 3'te görüldüğü üzere Y_i vektörü S_{2i} ve S_{3i} vektörlerinin elemanlarının karşılaştırılmasıyla 0 ve 1 sayılarından oluşmaktadır. S_{2i} ve S_{3i} 'ün 1., 3., 5. ve 7. elemanları aynı görevlerden oluştuğu için Y_i vektöründe bunlara karşılık 0 değeri verilmiştir. Y_i vektörü oluşturulduktan sonra S_{1i} vektörü mutasyon işlemine tabi tutulacaktır. Buradaki mutasyon işlemi ise klasik DGA'daki mutasyon işleminden farklıdır. Buradaki mutasyon işleminde Y_i vektöründeki 1 olan indislerin S_{1i} vektöründeki karşılıkları ardışık olarak yer değiştirirler. Örneğin, 2. ve 4. indisler Y_i vektöründe 1 olduğu için S_{1i} vektöründe 2. ve 4. indisler yer değiştirirler. Aynı şekilde 6. ve 8. indislerindeki görevler de karşılıklı yer değiştirirler. Oluşan yeni mutant vektör aşağıda Tablo 4'teki gibidir.

Tablo 4. Mutant Vektör(v_{iG+1})

İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	4	3	2	5	8	7	6

3.1.3. Çaprazlama

Çaprazlama işlemi klasik DGA'daki gibi yapılır. Aday vektör $u_{k,iG+1}$ aşağıdaki gibi oluşturulmaktadır.

$$u_{k,iG+1}=\begin{cases} v_{k,iG+1}\ eğer\ rasg \leq CR\ ya\ da\ k = RasgTams(1,D) \\ x_{k,iG}\ eğer\ rasg > CR \end{cases} \quad (3.2)$$

$$k \in [1,D], i \in [1,N_p]$$

Burada *rasg*, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı; $CR \in [0,1]$, kullanıcı tarafından belirlenen çaprazlama olasılığı ve *RasgTams (1,D)* ise [1,D] aralığında rasgele tamsayı üreten bir fonksiyondur. Bu fonksiyon en azından bir parametrenin mutant vektörden alınmasını garantilemektedir.

Tablo 5. Çaprazlama İşlemi

Rastgele Değer	**0,3**	**0,7**	**0,1**	**0,8**	**0,9**	**0,4**	**0,45**	**0,2**
Mutant Vektör ($v_{k,iG+1}$)	1	4	3	2	5	8	7	6
Hedef Vektör ($x_{k,iG}$)	1	2	3	4	5	7	6	8
Çaprazlama Sonucu Oluşan Vektör ($u_{k,iG+1}$)	1	2	3	4	5	8	7	6

Yukarıdaki örnekte CR (çaprazlama oranı) değeri 0,5 olarak alınmıştır. Bu değerin altında çıkan indislerin değerleri mutant vektörden diğerleri hedef vektördeki değerlerden alınmıştır. Çaprazlama işlemi sırasında aynı görevin iki defa seçilmesinin engellenmesi için çaprazlama operasyonu aşağıda Algoritma 1'e göre gerçekleştirilir.

Algoritma 1. Çaprazlama Operatörü Sözde Kod Yapısı

Başla

for (k=1; k≤D; k++)

rasg, [0,1] aralığında eş olasılıkla üretilmiş bir gerçel sayı

if($rasg \leq$CR)

$v_{k,iG+1}$ görevinin u_{iG+1} vektöründe olup olmadığını kontrol et, varsa (c=1) yoksa (c=0)

if(c==0)

$u_{k,iG+1} = v_{k,iG+1}$

endif

else

$x_{k,iG}$ görevinin u_{iG+1} vektöründe olduğunu kontrol et, varsa (c=1) yoksa (c=0)

if(c==0)

$u_{k,iG+1} = x_{k,iG}$

endif

endif

endfor

Boş kalan indisleri belirle

Kalan görevleri küçükten büyüğe doğru boş kalan indislere ata.

Bitir

3.1.4. Tamir Operatörü

Başlangıç popülasyonunu oluşturan çözüm vektörleri rastgele oluşturulduğu için, bazı vektörler öncül ve ardıl ilişkisine uymayabilir. Bu nedenle başlangıç popülasyonunun her bir vektörü için tamir operatörü uygulanır. Ayrıca, çaprazlama operatöründe rastgele işlemlere maruz kalan aday vektörler arasında da uygunsuz çözümler çıkabilir. Bu nedenle, aday vektörler de bir tamir sürecinden geçmektedirler. Tamir algoritmasının sözde kod yapısı aşağıda Algoritma 2'de verildiği gibidir.

Algoritma 2. Tamir Algoritması Sözde Kod Yapısı

Başla

for (m=1; m≤D; m++)

k=m

while (öncül-ardıl ilişkileri sağlanıncaya kadar)

do

$X_{k,iG}$ görevi ile kendisinden sonra gelen görevlerin öncül ve ardıl ilişkisini kontrol et, sağlıyorsa ($t_k = 1$) veya sağlamıyorsa ($t_k = 0$)

if ($t_k == 0$)

Bir sonraki görevi seç, ($k = k + 1$)

else

$X_{k,iG}$ görevi ile $X_{m,iG}$ görevinin yerini değiştir

$X_{k,iG} = X_{m,iG}$

endif

endwhile

endfor

Bitir

3.1.5. Görevlerin İstasyonlara Atanması: Uygunluk Değerinin Hesaplanması

Çözüm vektörlerinin uygunluk değerlerinin hesaplanması için, görevlerin istasyonlara atanması gerekir. Bu çalışmadaki GDGA'nın amacı verilen GMHDP için istasyon sayısının minimize edilmesidir. Dolayısıyla çalışmanın uygunluk değeri istasyon sayısıdır. Tamir operasyonundan çıkan hedef ve aday vektörlerdeki görevler, vektörün fenotipindeki sırayla tek tek istasyonlara atanırlar. Bu atamalar aşağıda Eş. 3.3 (Scholl vd., 2013) kullanılarak görev ve hazırlık sürelerinin toplamına bakılır. Eğer bu süre çevrim zamanını geçerse o istasyon kapanır ve yeni bir istasyon açılır. Böylece çözüm vektöründeki bütün görevler istasyonlara atanıncaya kadar bu işlem devam eder. En sonunda açılan istasyon sayısı çözüm vektörünün uygunluk değeri olur.

$$t_i + \tau_{i,j} + t_j + \tau_{j,h} + t_h + \mu_{h,i} \leq c \tag{3.3}$$

Eş. 3.3'deki t_i: i. görevin işlem zamanı, $\tau_{i,j}$: i ve j görevleri arasındaki ileri (forward) hazırlık zamanı, $\mu_{h,i}$: h ve i görevleri arasındaki geri (backward) hazırlık zamanı ve c: Çevrim zamanını göstermektedir.

3.1.6. Seçilim

Klasik DGA'da seçilim prosedürü aşağıdaki gibidir:

$$x_{iG+1}=\begin{cases} x_{iG} & eğer\ f(x_{iG}) \leq f(u_{iG+1}) \\ u_{iG+1} & aksi\ takdirde \end{cases} \tag{3.4}$$

$i \in [1, N_p]$

Seçilim işleminde hedef ve aday çözüm vektörlerinin uygunluk değerlerine bakılır ve hangisi daha küçükse o vektör bir sonraki nesle aktarılır. Bu çalışmadaki GDGA'nın seçilim işlemi, klasik DGA'nın seçilim işlemiyle aynıdır.

GDGA, tümüyle aşağıda Tablo 6'daki gibi özetlenebilir.

Tablo 6. Geliştirilmiş Diferansiyel Gelişim Algoritması Prosedürü

Başla

N_p sayısı kadar rastgele hedef vektör *i* oluştur (*i*=1…N_p)

Algoritma 2'de verilen tamir işlemlerini bütün vektörler üzerinde uygula

while(maksimum jenerasyon sayısına ulaşıncaya kadar)

for (i=1; i≤N_p;i++)

do

Vektör *i*'ye Bölüm 3.1.2'de verilen mutasyon işlemlerini uygula

Vektör *i*'ye Algoritma 1'de verilen çaprazlama işlemlerini uygula

Aday vektör *i*'ye Algoritma 2' de verilen tamir işlemlerini uygula

Hedef vektör *i* ve aday vektör *i* üzerinde verilen Eş. 3.4'te verilen seçilim işlemlerini uygula

endfor

endwhile

Bitir

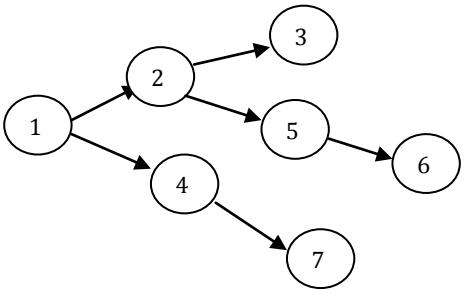
3.1.7. Geliştirilmiş Diferansiyel Gelişim Algoritması İçin Sayısal Örnek

Bu bölümde GDGA'nın nasıl çalıştığını daha iyi anlayabilmek için literatür problemlerinden Mertens'in 7 görevli ve çevrim zamanı 18 birim olan problemi ele alınacaktır. Problemde verilen görevlerin işlem zamanları Tablo 7'deki gibidir.

Tablo 7. Mertens Örneği İşlem Zamanları

Görevler	**1**	**2**	**3**	**4**	**5**	**6**	**7**
İşlem Zamanları	1	5	4	3	5	6	5

Mertens örneğinin öncül-ardıl ilişkileri Şekil 14'teki gibidir



Şekil 14. Mertens Örneği Öncelik Diyagramı

Sıra-bağımlı hazırlık zamanlı GMHDP'de görevler arasında hem ileri (forward), hem de geri (backward) hazırlık zamanları olmaktadır. Mertens örneğinin Scholl vd. (2013) tarafından eklenen ileri ve geri hazırlık zamanları aşağıda Tablo 8 ve Tablo 9'daki gibidir.

Tablo 8. Mertens Örneği İleri Hazırlık Zamanları

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	0	1	0	0	0	0	0
2	0	0	2	1	1	0	1
3	0	0	0	1	1	0	1
4	0	1	1	0	0	1	0
5	0	0	1	0	0	1	0
6	0	0	0	1	0	0	1
7	0	1	1	0	0	1	0

Tablo 9. Mertens Örneği Geri Hazırlık Zamanları

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	3	0	0	0	0	0	0
2	4	3	0	4	0	0	4
3	2	1	3	2	2	3	2
4	3	2	4	3	3	4	0
5	3	2	4	3	3	0	3
6	2	1	3	2	2	3	2
7	3	2	4	3	3	4	3

GDGA'nın ilk aşamasında başlangıç popülasyonu N_p oluşturulur. Bu örnekte başlangıç popülasyonunun kullanıcı tarafından N_p=8 olarak belirlendiğini varsayalım. Görev sayısı D=7 olduğu için, 8 adet 7 elemanlı çözüm vektörleri Tablo 10'daki gibi oluşturulmuştur.

Tablo 10. Mertens Örneği Başlangıç Populasyonu

X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	X_{61}	X_{71}	X_{81}
7	5	5	2	3	2	1	6
6	4	1	1	2	5	3	4
3	1	4	6	6	6	5	5
1	7	7	4	5	7	6	7
4	6	3	5	4	3	7	2
5	3	2	7	7	4	4	3
2	2	6	3	1	1	2	1

İlk nesil oluşturulduktan sonra her bir çözüm vektörü için aşağıda yapılan işlemler teker teker yapılır. Bu örnekte bu işlemler sadece bir çözüm vektörü üzerinde gösterilecektir. Populasyonun ilk çözüm vektörü X_{11} olduğu için hedef vektör de X_{11} olacaktır. GDGA'nın ilk aşamasında X_{11} hedef vektörü uygun (feasible) bir çözüm olmadığı için üzerinde tamir operasyonu olan Algoritma 2 uygulanacaktır. Tamir aşamalarından geçen X_{11} hedef vektörü Tablo 11'deki gibi son halini alacaktır.

Tablo 11. X_{11} Hedef Vektörü Tamir İşlemi

X_{11} Hedef Vektörü	**1.İşlem**	**2.İşlem**	**3.İşlem**	**4.İşlem**	**5.İşlem**	**X_{11} Hedef Vektörünün Son Hali**
7	1	1	1	1	1	1
6	6	4	4	4	4	4
3	3	3	7	7	7	7
1	7	7	3	2	2	2
4	4	6	6	6	5	5
5	5	5	5	5	6	6
2	2	2	2	3	3	3

Yukarıda Tablo 11'de gösterilen tamir işlemi populasyondaki bütün vektörler üzerinde uygulanır.

Tablo 12. Tamir İşleminden Sonra Vektörlerin Yapısı

X_{11}	X_{21}	X_{31}	X_{41}	X_{51}	X_{61}	X_{71}	X_{81}
1	1	1	1	1	1	1	1
4	4	2	2	2	2	2	4
7	2	4	5	5	5	5	2
2	7	7	4	6	4	6	7
5	5	3	6	4	3	4	5
6	3	5	7	7	7	7	3
3	6	6	3	3	6	3	6

Bütün vektörlere tamir işlemi uygulandıktan sonra mutasyon aşamasına geçilmektedir. Mutasyon işlemi için hedef vektörden ve birbirinden farklı 3 çözüm vektörü rassal seçilmektedir. Bu seçilen vektörlerin $S_{11} = X_{51}$, $S_{21} = X_{21}$, S_{31}=X_{31} olduğunu varsayalım. Daha sonra Eş. 3.1'de verilen işlemler uygulanır ve yer değiştirme noktalarını veren Y_i vektörü Tablo 13'teki gibi oluşturulur.

Tablo 13. Y_i Yer Değiştirme Vektörü

X_{21}	X_{31}	Y_i
1	1	0
4	2	1
2	4	1
7	7	0
5	3	1
3	5	1
6	6	0

Yer değiştirme Y_i vektörü oluşturulduktan sonra $S_{11} = X_{51}$ vektöründeki Y_i vektörünün 1'e karşılık gelen indisleri yer değiştirilir. Bu durumda X_{51} vektörünün 2., 3. ve 5., 6. indisleri yer değiştirilir. Böylece mutant v_{12} vektörü Tablo 14'deki gibi oluşturulur.

Tablo 14. Mutasyon İşlemi

			X_{51}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	2	5	6	4	7	3
			⇩				
			v_{12}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	5	2	6	7	4	3

Mutasyon işleminden sonra hedef vektör üzerinde çaprazlama işlemi uygulanır. Aday vektör u_{12} Algoritma 1'deki gibi oluşturulup Tablo 15'te verilmiştir.

Tablo 15. Çaprazlama İşlemi

Rastgele Değer	0,6	0,7	0,1	0,8	0,3	0,4	0,6
Mutant Vektör (v_{12})	1	5	2	6	7	4	3
Hedef Vektör (x_{11})	1	4	7	2	5	6	3
Çaprazlama Sonucu Oluşan Aday Vektör (u_{12})	1	4	2	-	7	-	3

Bu örnekte CR (çaprazlama oranı) değeri 0,5 alınmıştır. Aday vektörün 4. ve 6. indislerine gelen görevler daha önce vektörde bulunduğu için kalan 5 ve 6 görevleri kalan boş yerlere küçükten büyüğe doğru atanırlar. Çaprazlama sonucu oluşan aday vektörün son hali u_{12} Tablo 16'daki gibidir.

Tablo 16. Aday Vektör u_{12} Son Hali

			u_{12}				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	4	2	5	7	6	3

Çaprazlama sonucu oluşan aday vektör öncül ve ardıl ilişkisine uyduğu için tekrar bir tamir işlemi uygulamaya gerek kalmamıştır. GDGA'nın son aşamasında seçilim işlemi

bulunmaktadır. Eş. 3.4'de verilen seçilim işlemine göre hedef vektör ve aday vektörün uygunlukları karşılaştırılacaktır. Bu çalışma bir minimizasyon problemi olduğu için uygunluk fonksiyonu küçük olan; yani istasyon sayısı küçük olan vektör bir sonraki nesle aktarılacaktır. Uygunluk fonksiyonunun hesaplanması için öncelikle görevlerin istasyonlara atanması gerekir. Görevlerin istasyonlara atanması için Eş. 3.3 kullanılmıştır. Hedef vektör X_{11} ve aday vektörün u_{12} görevlerinin istasyon ataması Tablo 17 ve Tablo 18'deki gibidir:

Tablo 17. Hedef Vektör X_{11} İstasyon Ataması

İstasyon 1			**İstasyon 2**		**İstasyon 3**	
1	4	7	2	5	6	3

Tablo 18. Aday Vektör u_{12} İstasyon Ataması

İstasyon 1			**İstasyon 2**		**İstasyon 3**	
1	4	2	5	7	6	3

Burada her bir istasyondaki görevlerin ve ileri ve geri hazırlık zamanlarının toplamı çevrim süresi olan 18 birimi geçmemesi gerekir. Örneğin hedef vektörün X_{11} İstasyon 1'i şu şekilde hesaplanmıştır:

$$t_1 + \tau_{1,4} + t_4 + \tau_{4,7} + t_7 + \mu_{7,1} \leq 18$$

$$\rightarrow 1+0+3+0+5+3 \leq 18 \rightarrow 12 \leq 18$$

Diğer istasyonlar da aynı şekilde hesaplanmıştır. Seçilim işlemi için hem hedef vektörün hem de aday vektörün uygunluklarına bakılır. Uygunluk değeri vektörlerin istasyon sayılarına eşittir. Hedef vektörün uygunluk değeri $f(X_{11})$=3 ve aday vektörün uygunluk değeri $f(u_{12}) = 3$. Eş. 3.4'teki seçilim işlemine göre uygunluk değerleri birbirine eşitse bir sonraki nesle hedef vektör X_{11} taşınır. Dolayısıyla X_{12}= X_{11} olacaktır.

3.2. Geliştirilmiş Diferansiyel Gelişim Algoritması-Parçacık Sürü Optimizasyonu Hibrit Algoritması

Bu bölümde sıra-bağımlı hazırlık zamanlı GMHDP için Bölüm 3.1'de verilen Geliştirilmiş Diferansiyel Gelişim Algoritması (GDGA) ve Parçacık Sürü Optimizasyonu (PSO) metasezgisellerinin hibritize edilmesiyle oluşturulan Hibrit Algoritma (HA) açıklanmaktadır. HA'nın büyük bir kısmı GDGA'dan alınmasına rağmen, PSO'dan alınan bazı özellikler ile HA'nın yakınsama performansı yükselmiştir. HA, başlıca 5 aşamadan oluşmaktadır:

1.Başlangıç popülasyonun oluşturulması
2.Yer Değiştirme (Swap) işlemleri
3.Tamir Operatörü
4.Uygunluk Fonksiyonu
5.Seçilim

HA'nın büyük bir kısmı GDGA'dan alınmış olup, seçilim kısmı ise PSO'dan alınmıştır. Ayrıca PSO'da kullanılan global en iyi (global best) ve yerel en iyi (local best) kavramları da kullanılmıştır. Bunun sebebi, DGA'daki mutasyon işleminde rastgele 4 vektör seçildiği için en iyi uygunluk değerine sahip vektörün seçilme olasılığı düşük olmasıdır. Dolayısıyla değerli bilgilere sahip olan vektörün bilgilerinden yeterince istifade edilememektedir. HA'da ise global en iyi vektörün bilgileri saklanarak bu bilgilerden daha fazla yararlanma amacı güdülmüştür. HA'da her bir parçacığın yerel en iyisi (P_{best}) ve tüm populasyonun bir tane global en iyisi (G_{best}) kaydedilmektedir. Bu değerler her bir iterasyonda güncellenmektedir. Ayrıca, GDGA'daki çaprazlama operatörü yerel aramada etkili olmadığı için HA'da çaprazlama operatörüne yer verilmemiştir. Onun yerine yerel aramayı HA'da tek noktalı yer değiştirme işlemi yapmaktadır. Dolayısıyla HA'da arama operatörleri olarak çoklu yer değiştirme ve tek noktalı yer değiştirme işlemleri kullanılmıştır.

3.2.1. Başlangıç Populasyonunun Oluşturulması

Sıra-bağımlı hazırlık zamanlı GMHDP, kesikli ve permütasyonel olduğu için HA'nın çözüm parçacıkları gerçel sayılardan değil kesikli sayılardan oluşmaktadır. Her bir çözüm parçacığı D (görev sayısı) büyüklüğündeki vektörden oluşmaktadır. Ayrıca başlangıç sürüsünün sayısı S_n kadar olmaktadır. Burada populasyon sayısı S_n kullanıcı tarafından belirlenmektedir. S_n sayısı büyüdükçe modelin en iyi çözümü bulma yeteneği artmaktadır. Ancak aynı zamanda çözüm bulma süresi de artmaktadır. Başlangıç sürüsündeki her bir çözüm parçacığı, 1'den D'ye kadar ve birbirinden farklı tamsayılardan oluşan D büyüklüğünde bir vektördür. Örneğin, montaj hattının 8 görevden oluştuğunu ve popülasyon sayısının 20 olarak belirlendiğini varsayalım. Popülasyon sayısı 20 olduğu için başlangıç popülasyonu 20 adet 8 elemanlı ve elemanları birbirinden farklı çözüm parçacıklarından oluşur.

3.2.2. Yer Değiştirme (Swap) İşlemleri

Metasezgisel algoritmaların oluşturulması aşamasında keşif (exploration) ve faydalanma (exploitation) çok önemli iki kavramdır. Keşif, aşamasında çözüm uzayının çok farklı noktalarında en iyi çözüm bulunmaya çalışılır. Faydalanma da ise bulunan iyi bir çözümün yakınlarında daha iyi bir çözüm bulunmaya çalışılır. Dolayısıyla keşif, global arama yaparken, faydalanma ise yerel arama yapmaktadır. Arama stratejisinin önceliğine göre bazı yöntemler global arama, bazı yöntemler ise yerel arama konusunda ön plana çıkmaktadır. Bu nedenle keşif ve faydalanma arasındaki dengeyi sağlayacak bir arama stratejisi belirlemek önemlidir. GDGA'da kullanılan çaprazlama operatörü faydalanma kısmında yeterince başarılı olmamaktadır. Bu yüzden HA'da, hem global aramada hem de yerel aramada başarılı olan yer değiştirme işlemleri kullanılmıştır. HA'da iki çeşit yer değiştirme işlemi uygunlanmaktadır. Bunlar:

1.Çoklu yer değiştirme

2.Tek noktalı yer değiştirme

3.2.2.1. Çoklu Yer Değiştirme

HA'da kullanılan yer değiştirme işlemlerinden biri GDGA'nın mutasyon aşamasında kullanılan çoklu yer değiştirme işlemidir. Bu yer değiştirme işlemi çözüm uzayında keşif yaparak global arama yapar. Bu işlemin ilk aşamasında, populasyondaki her bir i parçacığı için birbirinden farklı rastgele 3 farklı parçacığın yerel en iyileri ($Pb_{x,}Pb_y, Pb_z$) seçilir. Bu parçacıklar seçildikten sonra yer değiştirme noktalarının bulunmasında GDGA'da kullanılan prosedür uygulanmaktadır. Bu prosedüre göre seçilen iki parçacığın her bir elemanının birbirine eşit olup olmadığına bakılır. Aynı ise 0, değilse 1 değeri verilir (Eş. 3.5).

$$S_1 = Pb_y - Pb_z$$

$$S_1 = \begin{cases} if(X == 0) & 0 \\ Aksi\ taktirde & 1 \end{cases} \tag{3.5}$$

Çoklu yer değiştirmenin ikinci aşamasında global en iyi (G_{best}) parçacığın bilgilerine bakılır. Her bir i parçacığının yerel en iyisi (Pb_i) ve global en iyi (G_{best}) karşılaştırılır. Yer değiştirme noktalarının bulunmasında yine GDGA'da kullanılan prosedür uygulanmaktadır. Bu prosedüre göre seçilen iki parçacığın her bir elemanının birbirine eşit olup olmadığına bakılır. Aynı ise 0, değilse 1 değeri verilir (Eş. 3.6).

$$S_2 = G_{best} - Pb_i$$

$$S_2 = \begin{cases} if(X == 0) & 0 \\ Aksi\ taktirde & 1 \end{cases} \tag{3.6}$$

Son aşamada eşit olasılıkla hangi bilginin kullanılacağına karar verilir (Eş. 3.7).

$$S = \begin{cases} G_{best} - Pb_i & eğer\ rast < 0.5 \\ Pb_y - Pb_z & aksi\ taktirde \end{cases} \tag{3.7}$$

Yer değiştirme için hangi parçacıkların seçileceğine karar verildikten sonra Pb_x parçacığı üzerinde yer değiştirme işlemi yapılır. Örneğin, yer değiştirme noktaları için

$S_1 = Pb_y - Pb_z$ bilgisinin alınacağına karar verilirse aşağıdaki tabloda verilen işlemler Pb_x parçacığı üzerinde Tablo 19'daki gibi uygulanır.

Tablo 19. Çoklu Yer Değiştirme İşlemi

				Pb_x				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	3	4	5	6	7	8
				Pb_y				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	3	4	2	5	8	7	6
				Pb_z				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	4	3	5	6	7	8
				S				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	0	1	0	1	0	1	0	1

Tablo 19'da görüldüğü üzere S vektörü Pb_yve Pb_z parçacıklarının elemanlarının karşılaştırılmasıyla 0 ve 1 sayılarından oluşmaktadır. Pb_y ve Pb_z 'nin 1., 3., 5. ve 7. elemanları aynı görevlerden oluştuğu için S vektöründe bunlara karşılık 0 değeri verilmiştir. S vektörü oluşturulduktan sonra Pb_x parçacığı yerdeğiştirme işlemine tabi tutulacaktır. Yer değiştirme işleminde S vektöründeki 1 olan indislerin Pb_x parçacığındaki karşılıkları ardışık olarak yer değiştirirler. Örneğin, 2. ve 4. indisler S vektöründe 1 olduğu için Pb_x vektöründe 2. ve 4. indisler yer değiştirirler. Aynı şekilde 6. ve 8. indislerindeki görevler de karşılıklı yer değiştirirler. Oluşan yeni parçacık P_i aşağıda Tablo 20'deki gibidir.

Tablo 20. Çoklu Yer Değiştirme İşleminden Sonra Oluşan P_i

İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	4	3	2	5	8	7	6

3.2.2.2. Tek Noktalı Yer Değiştirme

Yer değiştirme işlemlerinin diğer bir çeşidi ise tek noktalı yer değiştirme işlemidir. Burada amaç faydalanma ile yerel aramayı artırmaktır. Tek noktalı yer değiştirmede çoklu yer değiştirme işlemindeki gibi yer değiştirme noktaları Eş. 3.5'deki gibi belirlenir. S vektöründe 1'e karşılık gelen noktalardan biri rastgele seçilir. Daha sonra kendisinden bir sonraki 1'e karşılık gelen noktanın indisi de seçilir. Pb_x parçacığında bu indislere karşılık gelen görevleri yer değiştirilir.

Tablo 21. Tek Noktalı Yer Değiştirme İşlemi

				Pb_x				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	3	4	5	6	7	8
				S				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	0	1	0	1	0	1	0	1
				P_i				
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Değer	1	2	3	4	5	8	7	6

Yukarıdaki Tablo 21'de verilen örnekte S vektöründe rastgele 6. indis seçilmiştir. Ondan sonraki yer değiştirme noktası ise 8. indistir. Bu durumda Pb_x parçacığında bu indislere karşılık gelen görevlerin yerleri birbiriyle değiştirilir. Tek noktalı yer değiştirme sonucunda ise P_i parçacığı oluşmaktadır.

Daha önce de söylendiği gibi metasezgisel algoritmalarda keşif ve faydalanma arasındaki denge arttıkça en iyi (optimal) sonuca ulaşma ihtimali artar. Bu çalışmada, bu dengenin sağlanması için bir β parametresi tanımlanmıştır. β anlık iterasyonun (t) toplam iterasyona (T) oranı olup Eş. 3.8'de verildiği gibidir

$$\beta = \frac{t}{T} \tag{3.8}$$

Daha sonra 0 ve 1 arası rastgele bir sayı seçilir ve bu sayı β'dan büyükse keşif aşaması çalışır, daha küçükse faydalanma aşaması çalışır. Burada keşif aşaması çoklu yer değiştirme ve faydalanma aşaması ise tek noktalı yer değiştirme olmaktadır. Program ilk çalıştığında keşif aşamasıyla başlayacak olup, β değeri 1'e yaklaştıkça daha çok faydalanma yani tek noktalı yer değiştirme işlemini uygulama olasılığı artacaktır. Ayrıca, keşif ve faydalanma arasında olasılıklı bir değer olması da başlangıçta keşif arasında faydalanma yapılma olasılığı da az da olsa sağlanacak, aynı şekilde iterasyonların sonuna doğru faydalanmanın arasında düşük bir olasılıkla keşif yapılmasına da olanak verecektir.

Algoritma 3. Yer Değiştirme İşlemleri Sözde Kod Yapısı

Başla

Tablo 19 ve Tablo 21'de verildiği gibi bütün yer değiştirme noktalarını (SL) ve indislerini (V) belirle

[0, 1] arasında bir rastgele sayı (R) oluştur.

if (R> β) ise

for (i=1:2:SL)

do

P_{V^i} ve $P_{V^{i+1}}$ arasında yer değiştirme işlemini uygula

endfor

else

Rastgele bir yer değiştirme noktası seç $j| \ 1 < j < SL$

P_{V^j} and $P_{V^{j+1}}$ arasında yer değiştirme işlemini uygula

endif

Bitir

3.2.3.Tamir Operatörü

Başlangıç popülasyonunu oluşturan çözüm parçacıkları rastgele oluşturulduğu için, bazı parçacıklar öncül ve ardıl ilişkisine uymayabilir. Bu nedenle başlangıç popülasyonunun her bir elemanı için tamir operatörü uygulanır. Ayrıca, yer değiştirme operatöründe rastgele işlemlere maruz kalan parçacıklar da uygunsuz çözümlere dönebilirler. Bu nedenle, yer değiştirme işlemine maruz kalan parçacıklar da bir tamir sürecinden geçmektedirler. Tamir algoritmasının sözde kod yapısı aşağıda Algoritma 4'de verildiği gibidir.

Algoritma 4. Tamir Algoritması Sözde Kod Yapısı

Başla
for (m=1; m≤D; m++)
k=m
while (öncül ve ardıl ilişkileri sağlanıncaya kadar)
do
$X_{k,iG}$ görevi ile kendisinden sonra gelen görevlerin öncül ardıl ilişkisini kontrol et, sağlıyorsa ($t_k = 1$) veya sağlamıyorsa ($t_k = 0$)
if ($t_k == 0$)
Bir sonraki görevi seç, ($k = k + 1$)
else
$X_{k,iG}$ görevi ile $X_{m,iG}$ görevinin yerini değiştir
$X_{k,iG} = X_{m,iG}$
endif
endwhile
endfor
Bitir

3.2.4. Görevlerin İstasyonlara Atanması: Uygunluk Değerinin Hesaplanması

Çözüm parçacıklarının uygunluk değerlerinin hesaplanması için, görevlerin istasyonlara atanması gerekir. Bu çalışmadaki HA'nın amacı verilen GMHDP için istasyon sayısının minimize edilmesidir. Dolayısıyla çalışmanın uygunluk değeri istasyon sayısıdır. Tamir operasyonundan çıkan parçacıklar, fenotipindeki sıraya göre görevler tek tek istasyonlara atanırlar. Bu atamalar aşağıda Eş. 3.9 (Scholl vd., 2013) kullanılarak görev ve hazırlık sürelerinin toplamına bakılır. Eğer bu süre çevrim zamanını geçerse o istasyon kapanır ve yeni bir istasyon açılır. Böylece çözüm parçacığındaki bütün görevler istasyonlara atanıncaya kadar bu işlem devam eder. En sonunda açılan istasyon sayısı çözüm parçacığının uygunluk değeri olur.

$$t_i + \tau_{i,j} + t_j + \tau_{j,h} + t_h + \mu_{h,i} \leq c \tag{3.9}$$

Eş. 3.9'daki t_i: i. görevin işlem zamanı, $\tau_{i,j}$: i ve j görevleri arasındaki ileri (forward) hazırlık zamanı, $\mu_{h,i}$: h ve i görevleri arasındaki geri (backward) hazırlık zamanı ve c: Çevrim zamanını göstermektedir.

3.2.5. Seçilim

Seçilim aşamasında eğer yer değiştirme sonrası oluşan i parçacığının P_i uygunluk değeri $f(P_i)$ yerel en iyiden Pb_i daha iyi ise Pb_i=P_i olarak güncellenir. Aynı şekilde i parçacığının P_i uygunluk değeri $f(P_i)$ global en iyiden G_{best} daha iyi ise G_{best}=P_i olarak güncellenir. HA'daki seçilim işlemi PSO'nun seçilim işlemi ile aynıdır.

HA, tümüyle aşağıda Tablo 22'deki gibi özetlenebilir.

Tablo 22. Hibrit Algoritma Prosedürü

Başla

S_n sayısı kadar rastgele parçacık P_i oluştur (i=1…S_n)

Algoritma 4'te verilen tamir işlemlerini sürüdeki bütün parçacıklar üzerinde uygula

Sürüdeki bütün parçacıkların yerel en iyilerini P_{best} ve sürünün global en iyisini G_{best} belirle

while(maksimum iterasyon sayısına ulaşıncaya kadar)

for (i=1; i≤S_n;i++)

do

Parçacık P_i 'ye Algoritma 3'te verilen çoklu yer değiştirme veya tek noktalı yer değiştirme işlemlerini uygula

Algoritma 4'te verilen tamir işlemlerini parçacık P_i üzerinde uygula

Yerel en iyiyi Pb_i ve global en iyiyi G_{best} güncelle

endfor

endwhile

Bitir

3.2.6. GDGA-PSO Hibrit Algoritması İçin Sayısal Örnek

Bu bölümde HA'nın nasıl çalıştığını daha iyi anlayabilmek için literatür problemlerinden Mertens'in 7 görevli ve çevrim zamanı 18 birim olan problemi ele alınacaktır. Mertens örneği ile ilgili detaylar Bölüm 3.1.7'de verilmiştir.

HA'nın ilk aşamasında başlangıç çözümü P_n oluşturulur. Bu örnekte başlangıç sürü büyüklüğünün kullanıcı tarafından S_n=8 olarak belirlendiğini varsayalım. Görev sayısı D=7 olduğu için, 8 adet 7 elemanlı çözüm parçacıkları Tablo 26'daki gibi oluşturulmuştur.

Tablo 23. Mertens Örneği Başlangıç Sürüsü

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
7	5	5	2	3	2	1	6
6	4	1	1	2	5	3	4
3	1	4	6	6	6	5	5
1	7	7	4	5	7	6	7
4	6	3	5	4	3	7	2
5	3	2	7	7	4	4	3
2	2	6	3	1	1	2	1

Başlangıç sürüsü oluşturulduktan sonra her bir parçacık için yerel en iyi P_{best} ve global en iyinin G_{best}seçilmesi gerekir. İlk iterasyon olduğu için yerel eniyi P_{best} her bir parçacığın uyum değerine (fitness) eşittir. Global en iyi G_{best} ise bu parçacıklar arasında değeri en iyi olan değerdir. Parçacıkların uyum değerlerini bulmak için öncelikle parçacıkların öncül ve ardıl ilişkisine uyması gerekir. Bu nedenle her bir parçacık üzerinde tamir prosedürünün (Algoritma 4) uygulanması gerekir. Tamir aşamalarından geçen P_1 parçacığı Tablo 24'teki gibi son halini alacaktır.

Tablo 24. P_1 Parçacığı Tamir İşlemi

P_1 Parçacığı	1.İşlem	2.İşlem	3.İşlem	4.İşlem	5.İşlem	P_1 Parçacığının Son Hali
7	1	1	1	1	1	1
6	6	4	4	4	4	4
3	3	3	7	7	7	7
1	7	7	3	2	2	2
4	4	6	6	6	5	5
5	5	5	5	5	6	6
2	2	2	2	3	3	3

Diğer parçacıklar da aynı tamir işleminden geçirilmiştir. Tamir işleminden geçen parçacıkların son hali Tablo 25'te verilmiştir.

Tablo 25. Tamir İşleminden Sonra Parçacıkların Yapısı

P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
1	1	1	1	1	1	1	1
4	4	2	2	2	2	2	4
7	2	4	5	5	5	5	2
2	7	7	4	6	4	6	7
5	5	3	6	4	3	4	5
6	3	5	7	7	7	7	3
3	6	6	3	3	6	3	6

Tamir işleminden geçen parçacıkların yerel en iyileri ilk iterasyon olduğu için parçacıkların değerleri ile aynıdır $Pb_i = P_i$.

Tablo 26. Tamir İşleminden Sonra Parçacıkların Yerel En İyileri

Pb_1	Pb_2	Pb_3	Pb_4	Pb_5	Pb_6	Pb_7	Pb_8
1	1	1	1	1	1	1	1
4	4	2	2	2	2	2	4
7	2	4	5	5	5	5	2
2	7	7	4	6	4	6	7
5	5	3	6	4	3	4	5
6	3	5	7	7	7	7	3
3	6	6	3	3	6	3	6

Sürüdeki bütün parçacıkların istasyon atamaları Tablo 27'de verilmiştir.

Tablo 27. İstasyon Ataması

	İstasyon 1	**İstasyon 2**	**İstasyon 3**
P_{b1}	1,4,7	2,5	6,3
P_{b2}	1,4,2	7,5,3	6
P_{b3}	1,2,4,7	3,5	6
P_{b4}	1,2,5	4,6	7,3
P_{b5}	1,2,5	6,4	7,3
P_{b6}	1,2,5	4,3,7	6
P_{b7}	1,2,5	6,4	7,3
P_{b8}	1,4,2	7,5,3	6

Burada herbir istasyondaki görevlerin ileri ve geri hazırlık zamanlarının toplamı çevrim süresi olan 18 birimi geçmemesi gerekir. Örneğin Pb_1 parçacığının İstasyon 1'i şu şekilde hesaplanmıştır:

$$t_1 + \tau_{1,4} + t_4 + \tau_{4,7} + t_7 + \mu_{7,1} \leq 18$$

$$\rightarrow 1+0+3+0+5+3 \leq 18 \rightarrow 12 \leq 18$$

Diğer istasyonlar da aynı şekilde hesaplanmıştır. Aynı işlemler diğer parçacıklar için uygulanmış ve her bir parçacığın istasyon ataması yapılmıştır. Global en iyinin G_{best} seçilmesi için parçacıkların uyum değerlerinin karşılaştırılması gerekir. Parçacıkların uyum değerleri Tablo 28'de verildiği gibidir.

Tablo 28. Parçacıkların Uyum Değerleri

$f(P_{b1})$	$f(P_{b2})$	$f(P_{b3})$	$f(P_{b4})$	$f(P_{b5})$	$f(P_{b6})$	$f(P_{b7})$	$f(P_{b8})$
3	3	3	3	3	3	3	3

Tablo 28'de görüldüğü gibi bütün parçacıkların yerel en iyilerinin değerleri birbirine eşit ve 3 çıkmıştır. Bu eşitlik durumunda rastgele bir parçacık global eniyi G_{best} olacaktır. Rastgele atama sonunda $G_{best} = Pb_8$ olduğunu varsayalım.

HA'ya göre bir sonraki aşamada hangi yer değiştirme işleminin yapılacağına karar vermek için β parametresinin belirlenmesi gerekir. β parametresi Eş. 3.8'de verildiği gibi anlık iterasyonun (t), toplam iterasyona (T) oranıdır. İlk iterasyon olduğu için t=1 olacaktır. Toplam iterasyon sayısı da kullanıcı tarafından belirlenecek olup, bu örnekte T=1.000 olduğunu varsayalım. Bu durumda $\beta = \frac{1}{1.000}$ =0,001 olacaktır. Daha sonra program rastgele bir sayı (R) döndürecektir. Bu örnek için R=0,3 olduğunu varsayalım. Algoritma 3'e göre R> β olduğu için parçacık üzerinde çoklu yer değiştirme işlemi uygulanacaktır.

Çoklu yer değiştirme işlemini Pb_1 üzerinde uygulamak için rastgele Pb_1'den ve birbirinden farklı 3 farklı parçacığın yerel en iyisi seçilecektir. Bu örnek için bunların $Pb_x = Pb_3$, $Pb_y = Pb_4$, $Pb_z = Pb_5$ olduğunu varsayalım. Eş. 3.5'e göre yer değiştirme vektörü S_1'i oluştumak için öncelikle Pb_4 ve Pb_5 parçacıklarının üzerinde işlem yapılacaktır.

Tablo 29. S_1 Yer Değiştirme Vektörünün Oluşturulması

				Pb_4			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	2	5	4	6	7	3
				Pb_5			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	2	5	6	4	7	3
				S_1			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	0	0	0	1	1	0	0

Yukarıda Tablo 29'da Pb_4 ve Pb_6 parçacıklarının oluşturduğu S_1 yer değiştirme vektörü oluşturulmuştur. Aynı şekilde Eş. 3.6'da verildiği gibi S_2 yer değiştirme vektörünü oluşturmak için $Pb_1 \; ve \; G_{best}$ parçacıklarının üzerinde işlem uygulanacaktır. Bu örnekte ilk iterasyonda $G_{best} = Pb_8$ olduğu için $Pb_1 \; ve \; Pb_8$ parçacıklarının elemanları karşılaştırılacaktır.

Tablo 30. S_2 Yer Değiştirme Vektörünün Oluşturulması

				Pb_1			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	4	7	2	5	6	3
				G_{best}			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	4	2	7	5	3	6
				S_2			
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	0	0	1	1	0	1	1

Yer değiştirme vektörleri S_1 ve S_2 oluşturulduktan sonra, $Pb_x = Pb_3$, parçacığının yer değiştirme noktalarının belirlenmesi için Eş. 3.7'ye göre eşit olasılıkla S_1 ve S_2 vektörlerinden

biri seçilir. Bu örnekte S_2 vektörünün seçildiğini varsayalım. Bu durumda S_2 vektörünün 1'e karşılık gelen indislerinde $Pb_{3,}$ parçacığında yer değiştirme işlemi yapılacaktır. Dolayısıyla Pb_3'ün 3. ve 4. indislerine karşılık gelen görevler yer değiştireceklerdir. Aynı şekilde Pb_3'ün 6. ve 7. indislerine karşılık gelen görevler de yer değiştireceklerdir. Dolayısıyla sürünün yeni elemanı $P_{1,}$ Pb_3′ün yer değiştirme işleminden sonraki hali olacaktır.

Tablo 31. P_1 Parçacığı

	P_1						
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	2	7	4	3	6	5

Ancak oluşan P_1 öncül ve ardıl ilişkisine uymadığı için Algoritma 4'te verilen tamir işleminden geçecektir. Tamir işleminden geçen P_1'in son hali aşağıda Tablo 32'deki gibi olacaktır.

Tablo 32. P_1 Parçacığının Son Hali

	$\boldsymbol{P_1}$						
İndis	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Değer	1	2	4	7	3	5	6

Sürünün yeni üyesi P_1 oluşturulduktan sonra yerel en iyi Pb_1 ve global en iyi G_{best} güncellenmektedir. Ancak bu örnekte P_1 parçacığının yapısı ve uyum değeri değişmediği için yerel en iyi Pb_1 ve global en iyi G_{best} aynı kalmıştır.

Yukarıda verilen aşamaların tümü geri kalan P_2, P_3, P_4, P_5, P_6, P_7 ve P_8 oluşturulması aşamasında da kullanılacaktır. Her bir aşamada her bir parçacığın yerel en iyileri P_{best} ve global en iyi G_{best} güncellenmektedir. Bu aşamalar da bittikten sonra algoritmanın 1.

iterasyonu bitmiş olacaktır. Kullanıcı tarafından belirlenen iterasyon sayısı kadar tüm bu işlemler tekrarlanmaktadır. En son iterasyonda G_{best} olan parçacık en iyi çözüm olmaktadır.

DÖRDÜNCÜ BÖLÜM

ARAŞTIRMANIN BULGULARI

Bu bölümde sıra-bağımlı hazırlık zamanlı GMHDP için geliştirilmiş olan GDGA ve HA metasezgisel yöntemlerinin test problemleri üzerindeki deneysel çalışma sonuçları detaylı bir şekilde açıklanmıştır.

4.1. Deneysel Sonuçlar

GDGA ve HA, MATLAB programında kodlanmış ve test problemleri Intel Core i5, 2.4 GHz, 6 GB RAM özelliklerine sahip PC kullanılarak test edilmiştir. Test problemleri http://www.assembly-line-balancing.de adresinden alınmıştır. Literatürde geçen ve optimum çözümleri bilinen BMHDP'ye Scholl vd. (2013) tarafından ileri (forward) ve geri (backward) hazırlık zamanları eklenerek sıra-bağımlı hazırlık zamanlı GMHDP için 269 adet test problemi oluşturulmuştur. Bu çalışmada, deneysel çalışma için bu test problemleri kullanılacaktır. GDGA'nın kontrol parametreleri; çaprazlama oranı (CR) ve başlangıç populasyonu sayısıdır (N_p). GDGA'nın mutasyon işlemi yer değiştirme işleminden oluştuğu için bu aşamada kontrol parametresi bulunmamaktadır. HA'nın kontrol parametresi ise sadece sürü büyüklüğüdür (S_n). Başlangıç populasyonu N_p, başlangıç sürüsü S_n, ve çaprazlama oranı (CR)' nin farklı değerleri için GDGA ve HA çalıştırılmıştır. Bulunan sonuçlara göre S_n=N_p=50 ve CR =0,5 değerlerinde algoritmanın makul zamanda en iyi sonuçlar verdiği gözlenmiştir. Yolmeh ve Kianfar (2011) ve Pitakaso ve Sethanan (2015), test problemlerini, geliştirdikleri algoritma ile 5 kez çözmüş ve buldukları en iyi sonuçları literatürdeki diğer sonuçlarla karşılaştırmışlardır. Bu çalışmada da benzer şekilde her bir test problemi için GDGA ve HA 5'er kez çalıştırılmış ve bulunan en iyi sonuçlar kaydedilmiştir. Toplam 269 adet test problemi olduğu için toplamda 10.760 adet deney yapılmıştır.

Scholl vd. (2013), hazırlık zamanlarını oluşturmak için $\alpha.\, t_{ort}$ sayısını üst sınır seçmiş ve üçgensel eşitsizliği sağlayacak şekilde ileri ve geri hazırlık zamanları oluşturulmuştur. Burada t_{ort}, ortalama işlem zamanını temsil etmekte ve α ise 0,25, 0,50, 0,75, 1,00

sayılarından birini almaktadır. Bu durumda ileri ve geri hazırlık zamanları ortalama işlem süresinin 12,5, 25, 37,5, 50%'sine kadar değer alabilmektedir. Dolayısıyla 4 farklı α seviyesinde 4 farklı deney seti oluşturulmuştur. Detaylı bilgi için Scholl vd. (2013) çalışmasına bakılabilir. Bu çalışmada bütün α seviyeleri için GDGA ve HA çalıştırılmıştır.

Test problemleri daha önce de söylendiği gibi literatürde çalışılan BMHDP'ye, Scholl vd. (2013) tarafından ileri ve geri hazırlık zamanları eklenerek oluşturulmuştur. BMHDP'nin en iyi çözümleri bilinmesine rağmen, oluşturulan bu test problemleri çok daha karmaşık olduğu için en iyi çözümleri bilinmemektedir. Dolayısıyla bu sorunun üstesinden gelebilmek için ve Andres vd. (2008), buldukları algoritmaların performanslarını belirleyebilmek için bir alt sınır (lower bound) metodu kullanmışlardır. Andres vd. (2008), tarafından kullanılan alt sınır metodu daha sonra Scholl vd. (2013) tarafından geliştirilmiştir. Bu çalışmada da Scholl vd. (2013) tarafından geliştirilen alt sınır denklemi (Eş. 4.1) kullanıldı.

$$\mathrm{LB} = \min\left\{\mathrm{m} \geq \mathrm{LB1} \middle| \mathrm{t_{top}} + \tau_{\mathrm{top}}^{\mathrm{n-m}} + \mu_{\mathrm{top}}^{\mathrm{m}} \leq \mathrm{TC(m)}\right\} \quad (4.1)$$

$\mathrm{LB1} = \left\lceil \frac{t_{top}}{c} \right\rceil$, TC(m)=m.c

Eş. 4.1'deki LB: İstasyon sayısının alt sınırı, t_{top}: Bütün görevlerin toplam işlem zamanı, n: Toplam görev sayısı, m: Bilinen en iyi istasyon sayısı, τ_{top}^{n-m}: n-m sayıdaki görevin arasındaki toplam ileri hazırlık zamanı, μ_{top}^{m}: m sayıdaki görevin arasındaki toplam geri hazırlık zamanı ve c: Çevrim zamanını göstermektedir.

Bu çalışmadaki GDGA ve HA, Scholl vd. (2013) tarafından sıra-bağımlı hazırlık zamanlı GMHDP çözümü için geliştirilen ve en iyi sonuçları bulan 4 farklı sezgisel olan FBRI10, FBCRI10, FBCRI100 ve FBLS10 ile karşılaştırılmıştır. Bulunan sonuçlar Tablo 33 Tablo 34, Tablo 35 ve Tablo 36'da verilmiştir.

Tablo 33. GDGA, HA ve diğer sezgisellerin karşılaştırması (α=1,00 veri seti için)

	FBRI 10	FBCRI 10	FBCRI 100	FBLS 10	GDGA	HA
Rel.LB (%)	31,19	31,19	30,27	35,39	29,87	27,16
#Opt	8	8	8	4	20	24
CPU (sn)	1,86	1,86	18,85	6,32	54	16,16

Tablo 34. GDGA, HA ve diğer sezgisellerin karşılaştırması (α=0,75 veri seti için)

	FBRI 10	FBCRI 10	FBCRI 100	FBLS 10	GDGA	HA
Rel.LB (%)	26,04	26,03	25,23	29,02	27,92	23,61
#Opt	14	14	15	13	32	40
CPU (sn)	1,78	1,77	17,97	6,20	53,31	16,22

Tablo 35. GDGA, HA ve diğer sezgisellerin karşılaştırması (α=0,50 veri seti için)

	FBRI 10	FBCRI 10	FBCRI 100	FBLS 10	GDGA	HA
Rel.LB (%)	20,64	20,64	20,09	22,51	20,57	17,37
#Opt	20	20	20	18	39	47
CPU (sn)	1,68	1,66	17,23	6,14	54,78	16,91

Tablo 36. GDGA, HA ve diğer sezgisellerin karşılaştırması (α=0,25 veri seti için)

	FBRI 10	**FBCRI 10**	**FBCRI 100**	**FBLS 10**	**GDGA**	**HA**
Rel.LB (%)	14,00	14,00	13,57	14,85	11,52	9,40
#Opt	30	30	32	29	87	96
CPU (sn)	1,66	1,67	16,19	6,41	49,50	15,84

Yukarıdaki tablolarda verilen Rel.LB (%): Bulunan sonucun istasyon alt sınır sayısından ortalama sapma yüzdesi, #Opt: Bulunan en iyi çözüm sayısı, CPU (sn): Algoritmanın her bir problem için saniye cinsinden ortalama çalışma süresidir.

Tablo 33'e bakıldığında GDGA'nın α=1,00 veri seti için, 269 test problemi içinde bulduğu çözümlerin alt sınırdan ortalama sapma yüzdesi %29,87 ile sezgisel algoritmalardan daha iyi sonuçlar bulduğu görülmektedir. HA'nın ise alt sınırdan ortalama sapma yüzdesi %27,16 ile hem GDGA'dan hem de sezgisellerden daha iyi sonuçlar vermiştir. Bununla birlikte sezgisel algoritmalar 269 test problemi içinde 8 adet en iyi sonuca ulaşmışken, GDGA 20 adet ve HA ise 24 adet en iyi sonuca ulaşmıştır. GDGA, sezgisellere göre daha fazla en iyi sonuca ulaşmıştır. HA ise hem GDGA'dan hem de diğer sezgisellerden daha fazla en iyi sonuca ulaşarak daha iyi bir performans gösterdiği söylenebilir. GDGA, 54 saniye ortalama CPU çalışma süresiyle sezgisellere göre daha uzun zamanda sonuca ulaşmıştır. HA ise 16,16 saniye ortalama CPU çalışma süresiyle, en iyi sonucu veren FBCRI100 sezgiselinden daha kısa zamanda sonuca ulaşmıştır.

Tablo 34'e bakıldığında GDGA'nın α=0,75 veri seti için, 269 test problemi içinde bulduğu çözümlerin alt sınırdan ortalama sapma yüzdesi %27,92 ile FBRI10, FBCRI10, FBCRI100 sezgisel algoritmalarından daha düşük; FBLS10 sezgiselinden ise daha iyi

performans gösterdiği görülmektedir. HA'nın ise alt sınırdan ortalama sapma yüzdesi %23,61 ile hem GDGA'dan hem de sezgisellerden daha iyi sonuçlar vermiştir. Bununla birlikte 269 test problemi içinde sezgiseller arasında en iyi performans gösteren FBCRI100 sezgiseli 15 adet en iyi sonuca ulaşmışken, GDGA 32 adet ve HA ise 40 adet en iyi sonuca ulaşmıştır. GDGA, sezgisellere göre daha fazla en iyi sonuca ulaşmıştır. Burada yine HA'nın hem GDGA'dan hem de diğer sezgisellerden daha fazla en iyi sonuca ulaşarak daha iyi bir performans gösterdiği söylenebilir. GDGA; 53,31 saniye ortalama CPU çalışma süresiyle sezgisellere göre daha uzun zamanda sonuca ulaşmıştır. HA ise 16,22 saniye ortalama CPU çalışma süresiyle, en iyi sonucu veren FBCRI100 sezgiselinden daha kısa zamanda sonuca ulaşmıştır.

Tablo 35 incelendiğinde GDGA'nın α=0,50 veri seti için, 269 test problemi içinde bulduğu çözümlerin alt sınırdan ortalama sapma yüzdesi %20,57 ile FBCRI100 sezgiselinden daha düşük; FBRI10, FBCRI10, FBLS10 sezgisellerinden ise daha iyi performans gösterdiği görülmektedir. HA ise alt sınırdan ortalama sapma yüzdesi %17,37 ile hem GDGA'dan hem de sezgisellerden daha iyi sonuçlar vermiştir. Bununla birlikte 269 test problemi içinde sezgiseller arasında FBRI10, FBCRI10 ve FBCRI100 sezgiselleri 20 adet en iyi sonuca ulaşmışken, GDGA 39 adet ve HA ise 47 adet en iyi sonuca ulaşmıştır. GDGA, sezgisellere göre daha fazla en iyi sonuca ulaşmıştır. Bu sonuçlara göre HA'nın hem GDGA'dan hem de sezgisellerden daha fazla en iyi sonuca ulaşarak daha iyi bir performans gösterdiği söylenebilir. GDGA; 54,78 saniye ortalama CPU çalışma süresiyle sezgisellere göre daha uzun zamanda sonuca ulaşmıştır. HA ise 16,91 saniye ortalama CPU çalışma süresiyle, en iyi sonucu veren FBCRI100 sezgiselinden daha kısa zamanda sonuca ulaşmıştır.

Tablo 36 incelendiğinde GDGA'nın α=0,25 veri seti için, 269 test problemi içinde bulduğu çözümlerin alt sınırdan ortalama sapma yüzdesi %11,52 ile tüm sezgisellerinden daha iyi performans gösterdiği görülmektedir. HA ise alt sınırdan ortalama sapma yüzdesi %9,40 ile hem GDGA'dan hem de sezgisellerden daha iyi sonuçlar vermiştir. Bununla

birlikte 269 test problemi içinde sezgiseller arasında en iyi performans gösteren FBCRI100 sezgiseli 32 adet en iyi sonuca ulaşmışken, GDGA 87 adet ve HA ise 96 adet en iyi sonuca ulaşmıştır. GDGA, sezgisellere göre daha fazla en iyi sonuca ulaşmıştır. HA'nın ise yine hem GDGA'dan hem de sezgisellerden daha fazla en iyi sonuca ulaşarak daha iyi bir performans gösterdiği söylenebilir. GDGA; 49,50 saniye ortalama CPU çalışma süresiyle sezgisellere göre daha uzun zamanda sonuca ulaşmıştır. HA ise 15,84 saniye ortalama CPU çalışma süresiyle, en iyi sonucu veren FBCRI100 sezgiselinden daha kısa zamanda sonuca ulaşmıştır.

Tablolar karşılaştırıldığında α seviyeleri düştükçe problemlerin zorluk derecesi azaldığı için tüm algoritmaların daha iyi sonuçlar verdiği görülmektedir. GDGA, α=1,00 ve α=0,25 seviyelerinde sezgisellerden daha iyi sonuçlar bulmuşken, α=0,50 ve α=0,75 seviyelerinde ise daha düşük performans göstermiştir. Fakat GDGA'nın tüm α seviyelerinde bulduğu en iyi çözüm sayısı sezgisellerin bulduğu en iyi çözüm sayısından fazla olmuştur. Dolayısıyla GDGA'nın genel itibariyle sezgisellerden daha iyi bir performans gösterdiği söylenebilir. HA ise tüm α seviyelerinde hem sezgisellerden hem de GDGA'dan daha iyi sonuçlar vermiştir. GDGA'nın bilgisayardaki ortalama çalışma süresi sezgisellere ve HA'ya göre tüm α seviyelerinde daha uzun sürmüştür. HA'nın bilgisayardaki ortalama çalışma süresi ise diğer sezgisellere göre en iyi sonuç veren FBCRI100 sezgiselinden tüm α seviyelerinde daha kısa sürmüştür. Ancak, HA'nın çalışma süresi diğer sezgisellerden ise biraz daha uzun sürmüştür. Ancak, çözüm kalitesi bakımından daha üstün olduğu için bu fark kabul edilebilir bir seviyededir.

BEŞİNCİ BÖLÜM

SONUÇ VE ÖNERİLER

MHDP literatüründe şu ana kadar çok fazla çalışma yapılmıştır. Bu çalışmada ele alınan sıra-bağımlı hazırlık zamanlı GMHDP ise pratikte çok fazla karşılaşılan bir problem olmasına rağmen literatürde çok fazla yer bulmamıştır. Literatürdeki bu açığı kapatmak için bu çalışma yapılmıştır. Daha önce bu konuyla ilgili yapılan çalışmalarda (Andres vd. (2008), Martino ve Pastor (2010), Scholl vd. (2013)), bu problem tipi sezgisel yöntemlerle çözülmüş olup ileride yapılacak çalışmalarda metasezgisel yöntemlerin uygulanabileceği önerilmiştir. Dolayısıyla bu öneriler dikkate alınarak bu çalışmada literatürde ilk defa sıra-bağımlı hazırlık zamanlı tip-1 GMHDP bir hibrit metasezgisel yöntemle çözülmüştür.

Bu çalışmada, sıra-bağımlı hazırlık zamanlı tip-1 GMHDP'nin çözümü için öncelikle yeni bir DGA geliştirilmiştir. Daha sonra geliştirilen bu DGA (GDGA) ve PSO algoritmaları hibritize edilerek bir hibrit algoritma (HA) önerilmişir. Bu çalışmada çözülmeye çalışılan sıra-bağımlı hazırlık zamanlı GMHDP, BMHDP'den çok daha karmaşık yapıya sahiptir. Bu MHDP tipinde hem görevlerin istasyonlara atanması, hem de istasyonlara atanan görevlerin arasında bir çizelgeleme yapılması gerekmektedir. Dolayısıyla bu durum bu problemi çok daha karmaşık hale getirmektedir. Bu tür NP-zor problemler için doğrusal programlama, dinamik programlama ve dal-sınır algoritmaları gibi kesin yöntemlerle makul zamanlarda optimum çözümlerinin bulunması büyük problemler için çok zordur. Bu nedenle, bu tür problemlerin çözümleri için literatürde sezgisel veya metasezgisel yöntemler çok sık kullanılmaktadır. Bu çalışmada da GDGA ve HA metasezgiselleri kullanılarak sıra-bağımlı hazırlık zamanlı GMHDP çözülmüştür.

Bu çalışmada önerilen GDGA ve HA, klasik DGA ve PSO'dan farklı olarak kesikli bir yapıda olan sıra-bağımlı hazırlık zamanlı GMHDP'ye uyarlanmıştır. Bu yüzden klasik

DGA'da kullanılan çaprazlama ve mutasyon operatörleri GDGA ve HA'da çoğunlukla değiştirilmiştir. Her iki yöntemde yer değiştirme (swap) işlemleri kullanılmıştır. GDGA ve HA'nın literatürdeki diğer sezgisel yöntemlere üstünlük sağlamasının sebebi kullanılan bu yer değiştirme işlemleridir. Yapılan deneylerde GDGA, literatürdeki sezgisel yöntemlere göre en iyi çözüme yakınsama yeteneği açısından genelde üstünlük sağlamıştır. GDGA'da mutasyon operatörü olarak kullanılan yer değiştirme işlemi, global aramayı etkin bir şekilde yapmasına rağmen, çaprazlama operatörü yerel aramada aynı başarıyı göstermemiştir. Bu nedenle HA'da çaprazlama operatörü yerine tek noktalı yer değiştirme işlemi uygulanmıştır. Tek noktalı yer değiştirme işlemi yerel aramada çaprazlama operatöründen daha başarılı olmuştur. HA'nın diğer bir avantajı ise global arama (keşif) ve yerel arama (faydalanma) arasındaki dengeyi sağlamak için kullandığı β parametresidir. PSO'da kullanılan global en iyi (G_{best}) ve lokal en iyi (P_{best}) kavramları ile GDGA'nın birleşmesi HA'nın en güçlü yanlarından biri olduğu söylenebilir. Bu nedenlerden dolayı HA, hem sezgisellere hem de GDGA'ya göre daha kısa CPU çalışma sürelerinde daha fazla en iyi çözümlere ulaşabilmiştir.

Bundan sonraki yapılacak çalışmalarda aşağıda belirtilen hususlar değerlendirmeye alınabilir:

- Bu çalışmada GDGA'nın parametrelerinin en iyilenmesi için kapsamlı bir deney tasarımı yapılmamıştır. İleriki çalışmalarda parametre en iyilenmesi için bir deney tasarımı yapılabilir.
- Sıra-bağımlı hazırlık zamanlı tip-2 GMHDP için bu çalışmada kullanılan GDGA ve HA uygulanabilir.
- Sıra-bağımlı hazırlık zamanlı GMHDP'nin daha karmaşık ve gerçekçi hali olan sıra-bağımlı hazırlık zamanlı U-tipi, çift taraflı veya paralel istasyonlu GMHDP için de bu çalışmada kullanılan GDGA ve HA uygulanabilir.

KAYNAKÇA

Ab Rashid, M. F. F., Tiwari, A. and Hutabarat, W. (2019). Integrated optimization of mixed-model assembly sequence planning and line balancing using multi-objective discrete particle swarm optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 33(3), 332-345.

Adham, A. Zainuddin, H., Siali, F. and Azizan, N. (2013). Assembly line balancing in manufacturing processes: Using simulation model. *Advanced Materials Research*, 748, 1183-1187.

Ağpak, K., Gökçen, H., Saray, N.N. ve Özel, S. (2002). Stokastik Görev Zamanlı Tek Modelli U Tipi Montaj Hattı Dengeleme Problemleri İçin Bir Sezgisel. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 17(4), 115-124.

AitZai, A., Benmedjdoub, B. and Boudhar, M. (2016). Branch-and-bound and PSO algorithms for no-wait job shop scheduling. *Journal of Intelligent Manufacturing*, 27, 679-688.

Akpınar, Ş., Bayhan, G.M. ve Baykaşoğlu, A. (2013). Hybridizing Ant Colony Optimization via Genetic Algorithm for Mixed-Model Assembly Line Balancing Problem with Sequence Dependent Setup Times between Tasks. *Applied Soft Computing*, 13(1), 574-589.

Ali, M.M. and Törn, A., (2004). Population set-based global optimization algorithms: Some modifications and numerical studies. *Computers and Operations Research*, 31, 1703–1725.

Amen, M. (2001). Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. *International Journal of Production Economics*, 69 (3), 255–264.

Andres, C., Miralles, C. and Pastor, R. (2008). Balancing and Scheduling Tasks in Assembly Lines with Sequence-Dependent Setup Times. *European Journal of Operational Research*, 187(3), 1212-1223.

Bagher, M., Zandieh, M. and Farsijani, H. (2011). Balancing of stochastic U-type assembly lines: an imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology*, 54 (1-4), 271-285.

Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52(3), 422-439.

Bansal, J., Singh, P., Saraswat, M., Verma, A., Jadon, S., and Abraham, A. (2011). Inertia weight strategies in particle swarm optimization. *In Nature and Biologically Inspired Computing (NaBIC)*, 2011 Third World Congress, 633-640.

Bartholdi, J.J. (1993). Balancing two-sided assembly lines: A case study. *International Journal of Production Research*, 31(10), 2447–2461.

Bautista, J. and Pereira, J. (2002). Ant algorithms for Assembly Line Balancing. *Lecture notes in Computer Science*, 2463, 65-75.

Bautista, J. and Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem. *European Journal of Operational Research*, 177 (3), 2016-2032.

Baykaşoğlu, A. ve Dereli, T. (2008). Two-sided assembly line balancing using an ant-colonybased heuristic. *International Journal of Advanced Manufacturing Technology*, 36(5-6), 582–588.

Becker, C. and Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168 (3), 694–715.

Becker, C. and Scholl, A. (2009). Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research*, 199, 359–374.

Belmecheri, F., Prins, C. , Yalaoui, F. and Amodeo, L. (2010). Particle Swarm Optimization to solve the Vehicle Routing Problem with Heterogeneous fleet, Mixed Backhauls, and time windows. *Journal of Intelligent Manufacturing*, 24, 1-6.

Benzer, R. (2005). *Paralel montaj hattı dengeleme problemi için yeni modeller*. Doktora Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

Blum, C., and Miralles, C. (2010). On solving the assembly line worker assignment and balancing problem via beam search. *Computers & Operations Research*, 38, 328-339.

Blum, C., Puchinger, J., Raidl, G. and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11, 4135-4151.

Borisovsky, P., Dolgui, A. and Kovalev, S. (2012) Algorithms and implementation of a set partitioning approach for modular machining line design. *Computers & Operations Research*, 39(12), 3147- 3155.

Boussaïd, I., Lepagnot, J., and Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237, 82-117.

Bowman, E. H. (1960). Assembly Line Balancing by Linear Programming. *Operations Research*, 3, 385-389.

Boysen, N., Fliedner, M. and Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2), 674–693.

Boysen, N., Fliedner, M. and Scholl, A. (2008). Assembly line balancing: Which model to use when?. *International Journal of Production Economics*, 111(2), 509–528.

Buzacott, J.A. and Shanthikumar, J.G. (1993). *Stochastic models of manufacturing systems.* Prentice-Hall, Englewood Cliffs, NJ.

Capacho L. and Pastor, R. (2006). The ASALB problem with processing alternatives involving different tasks: definition, formalization and resolution. *Lecture Notes in Computer Science*, 3982, 554-563.

Chakaravarthy, G., Marimuthu, S. and Sait, A. (2013). Performance evaluation of proposed Differential Evolution and Particle Swarm Optimization algorithms for scheduling m - machine flow shops with lot streaming. *Journal of Intelligent Manufacturing*, 24.

Che, Z.H. (2016). A multi-objective optimization algorithm for solving the supplier selection problem with assembly sequence planning and assembly line balancing. *Computers & Industrial Engineering.* 105, 247-259.

Cheldelin, B. and Ishii, K. (2004). Mixed model assembly quality: An approach to prevent human errors. *American Society of Mechanical Engineers, Design Engineering Division (Publication) DE,* 117, 109–119.

Cheng, S.L. and Hwang, C., (2001). Optimal approximation of linear systems by a differential evolution algorithm, *IEEE Transactions on Systems, Man, and Cybernetics-Part A, Systems and Humans*, 31(6), 698–707.

Chica, M., Cordón, O., Damas, S., and Bautista, J. (2010). A multiobjective GRASP for the 1/3 variant of the time and space assembly line balancing problem. *Trends in Applied Intelligent Systems*, 6098, 656-665.

Chutima, P. and Chimklai, P. (2012). Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge. *Computers & Industrial Engineering*, 62, 39-55.

Clerc, M. and Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), 58-73.

Corominas, A., Pastor, R. and Plans, J. (2008). Balancing assembly lines with skilled and unskilled workers. *Omega*, 36(6), 1126-1132.

Delice, Y., Aydogan, E., Özcan, U. ve İlkay, M. S. (2014). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, 28, 23-36.

Delice, Y., Aydogan, E., Özcan, U. ve İlkay, M. S. (2016). Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm. *4OR*, 15, 37-66.

Delorme, X., Dolgui, A. and Kovalyov, M. (2012). Combinatorial design of a minimum cost transfer line. *Omega*, 40(1), 31-41.

Diri, Z., Mete, S., Çil, Z.A. ve Ağpak, K. (2015). Stokastik Sıra-Bağımlı Hazırlık Zamanlı Montaj Hattı Dengeleme Problemi. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 21(4), 152-157.

Dolgui, A., Guschinsky, N., Levin, G. and Proth, J. (2008). Optimisation of multi-position machines and transfer line. *European Journal of Operational Research*, 185 (3), 1375-1389.

Dong, J., Zhang, L. and Xiao, T. (2015). A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints. *Journal of Intelligent Manufacturing*. 10, 737-751.

Dou, J., Li, J. and Zhao, X. (2017). A Novel Discrete Particle Swarm Algorithm for Assembly Line Balancing Problems. *Assembly Automation*, 37, 452-463.

Eberhart, R., Simpson, P. and Dobbins, R. (1996). *Computational Intelligence PC Tools*. Academic Press Professional, Inc., San Diego, CA.

Engelbrecht, A. (2012). Particle swarm optimization: Velocity initialization. *In 2012 IEEE Congress on Evolutionary Computation*, 1-8.

Engelbrecht, A. P. (2007). *Fundamentals of Computational Swarm Intelligence*. Wiley.

Erel, E., Sabuncuoğlu, İ. ve Aksu, B. (2001). Balancing of u-type assembly systems using simulated annealing. *International Journal of Production Research*, 39(13), 3003-3015.

Essafi, M., Delorme, X. and Dolgui, A. (2012). A reactive GRASP and Path Relinking for balancing reconfigurable transfer lines. *International Journal of Production Research*, 50(18), 5213- 5328.

Gamberini, R., Grassi, E.G.A. and Regattieri, A. (2009). A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. *International Journal of Production Research*, 47(8), 2141-2164.

Gangsterer, M. and Hartl, R. F. (2017). One- and two-sided assembly line balancing problems with real-world constraints. *International Journal of Production Research*, 56(8), 3025-3042.

Ghosh, S. and Gagnon, R. J. (1989). A comprehensive literature review and anal- ysis of the design, balancing and scheduling of assembly systems. *The International Journal of Production Research*, 27(4), 637-670.

Glover, F. (1989). Tabu-search-part I, *ORSA Journal on Computing*, 1(3), 190–206.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers, Boston.

Gökçen, H., Agpak, K. ve Benzer, R. (2006). Balancing of parallel assembly lines. *International Journal of Production Economics*, 103(2), 600–609.

Guschinskaya, O., Gurevsky, E., Dolgui, A. and Eremeev, A. (2011). Metaheuristic approaches for the design of machining lines. *International Journal of Advanced Manufacturing Technology*, 55(1), 11-22.

Gutierrez, A. L., Lanza, M., Barriuso, I., Valle, L., Domingo, M., Perez, J. R. and Basterrechea, J. (2011). Comparison of different PSO initialization techniques for high dimensional search space problems: A test with FSS and antenna arrays. *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, Rome, 965-969.

Gutjahr, A.L. and Nemhauser, G.L. (1964). An algorithm for the line balancing problem. *Management Science*, 11(2), 308–315.

Güner, F. (2019). *Sıra Bağımlı Hazırlık Zamanlarını Dġkkate Alan Paralel Çok İşçili Montaj Hatlarının Dengelenmesi*. Doktora Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

Güngör, A., ve Gupta, S. (2001). A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, 39 (7), 1427-1467.

Hamta, N., Fatemi Ghomi, S., Jolai, F. and Bahalke, U. (2011). Bi-criteria assembly line balancing by considering flexible operation times. *Applied Mathematical Modelling*, 35(12), 5592-5608.

Hamta, N., Ghomi, S.M.T.F., Jolai, F. and Shirazi, M. A., (2013). A Hybrid PSO Algorithm for a Multi-Objective Assembly Line Balancing Problem with Flexible Operation Times, Sequence-Dependent Setup Times and Learning Effect. *International Journal of Production Economics*, 141(1), 99-111.

Hansen, P., and Mladenović, N. (1999). An introduction to variable neighborhood search. In Voβ, S., Martello, S., Osman, I., and Roucairol, C., editors, Metaheuristics: advances and trends in local search paradigms for optimization, 433-438. Kluwer Academic Publishers.

Helander, M. (1995). *A Guide to the Ergonomics of Manufacturing*. Taylor & Francis, London

Hu, T. (1961). Parallel Sequencing and Assembly Line Problems. *Operations Research*, 9(6), 841-848.

Hu, X., Wu, E., Jinsong, B. and Jin, Y. (2010). A branch-and-bound algorithm to minimize the line length of a two-sided assembly line. *European Journal of Operational Research*, 206(3), 703- 707.

Jackson, J.R. (1956). A Computing Procedure for a Line Balancing Problem. *Management Science*, 2(3), 261-271.

Janardhanan, M. N., Huang, G. and Ponnambalam, S. G. (2015). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of Cleaner Production*, 90, 311-325.

Janardhanan, M. N., Ponnambalam S.G., and Jawahar, N. (2016). Design of energy efficient RAL system using evolutionary algorithms. *Engineering Computations*, 33, 580-602.

Janardhanan, M.N., Li, Z., Bocewicz, G., Banaszak, Z and Nielsen, P., (2018). Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times, *Applied Mathematical Modelling*, 65, 256-270.

Janardhanan, M. N., Ponnambalam S.G., Jawahar, N. and Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing and Applications*. 26, 1379-1393.

Janardhanan, M. N., Nielsen, I., Ponnambalam, S. G. and Venkataramanaiah, S. (2016). Differential evolution algorithm for solving RALB problem using cost- and time-based models. *The International Journal of Advanced Manufacturing Technology*, 89(1), 311-332.

Jayaswal, S. and Agarwal, P. (2014). Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. *Journal of Manufacturing Systems*, 33(4), 522-534.

Junsong, L. (2014). Applied technology in assembly line balancing based on genetic algorithm and simulation. *Advanced Materials Research*, 886, 564 -567.

Kaelo, P. and Ali, M.M., (2005). A numerical study of some modified differential evolution algorithms. *European Journal of Operational Research*, 169, 1176–84.

Kazemi, S., Ghodsi, R., Rabanni, M. and Tavakkoli-Moghaddam, R. (2011). A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. *International Journal of Advanced manufacturing Technology*, 55 (9-12), 1111-1122.

Kennedy, J. (1998). *The behavior of particles*. Springer Berlin Heidelberg, Berlin, 579-589.

Kennedy, J. and Eberhart, R.C. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ.

Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. *Proc. 1997 Conf. on Systems, Man, and Cybernetics, Piscataway*, NJ: IEEE Service Center, 4104–4109.

Kennedy, J. and Mendes, R. (2002). Population structure and particle swarm performance. In Evolutionary Computation. *Proceedings of the 2002 Congress*, 1671-1676.

Kennedy, J., Eberhart, R. C and Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA.

Khan, A. and Day, A.J. (2001). A knowledge based design methodology for manufacturing assembly lines. *Computers and Industrial Engineering*, 41(4), 441–467.

Kim, Y. K., Kim, Y. J. and Kim. Y. (1996). Genetic algorithms for assembly line balancing with various objectives. *Computers Industrial Engineering*, 30(3), 397–409.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220, 671-680.

Kriengkorakot, N. and Pianthong, N. (2007). The assembly line balancing problem. *KKU Enginieering Journal*, 34(2), 133-140.

Lapierre, S.D. and Ruiz, A.B. (2004). Balancing assembly lines: An industrial case study. *Journal of the Operational Research Society*, 55(6), 589–597.

Maniezzo, V., Stützle, T. and Voβ, S. (2009). *Matheuristics: Hybridizing metaheuristics and mathematical programming*, Springer.

Martino, L. and Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48(6), 1787–1804

Mayer, D.G., Kinghorn, B.P. and Archer, A.A. (2005). Differential Evolution – An Easy and Efficient Evolutionary Algorithm for Model Optimisation. *Agricultural Systems*, 83, 315-328.

Miltenburg, G. and Wijngaard, J. (1994). The u-line line balancing problem. *Management Science*, 40(10), 1378-1388.

Miltenburg, J. (2001). U-shaped production lines: A review of theory and practice. *International Journal of Production Economics*, 70(3), 201–214.

Miralles, C., García-Sabater, J., Andrés, C. and Carlos, M. (2007). Advantages of assembly lines in sheltered work centres for disabled. A case study. *International Journal of Production Economics*, 110(1-2), 187-197.

Miralles, C., García-Sabater, J., Andrés, C. and Carlos, M. (2008). Branch and bound procedure for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. Discrete *Applied Mathematics*, 156(3), 352-367.

Mohemmed, A., Sahoo, N. and Geok, T. (2008). Solving shortest path problem using particle swarm optimization. *Applied Soft Computing*, 8, 1643-1653.

Montes, M. E., Miranda-Varela, M. E. and Carmen Gomez-Ramon, R., (2010). Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, 180(22), 4223–4262.

Mozdgir, A., Mahdavi, I., Seyedi Badeleh, I. and Solimanpur, M. (2013). Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, 57(1–2), 137–151.

Nearchou, A. C. (2006). Meta-heuristics from Nature for the Loop Layout Design Problem, *International Journal of Production Economics*, 101(2), 312–328.

Nearchou, A. C. (2007). Balancing large assembly lines by a new heuristic based on differential evolution method. *International Journal of Advanced Manufacturing Technology*, 34, 1016–1029.

Nearchou, A. C. (2008). Multi-objective balancing of assembly lines by population heuristics. *International Journal of Production Research*, 46(8), 2275–2297.

Nearchou, A. (2011). Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, 129(2), 242-250.

Nearchou, A.C., Omirou, S.L., (2017). Assembly Line Balancing Using Differential Evolution Models. *Cybernetics and Systems*, 48(5), 436-458.

Nof, S.Y., Wilhelm, W.E. and Warnecke, H.J. (1997). *Industrial Assembly*. Chapman & Hall, New York.

Nourmohammadi, A., Zandieh, M. (2011). Assembly line balancing by a new multiobjective differential evolution algorithm based on TOPSIS. *International Journal of Production Research*, 49(10), 2833–2855.

Özcan, E. and Mohan, C. K. (1999). Particle swarm optimization: surfing the waves. In Evolutionary Computation, 1999. *CEC 99. Proceedings of the 1999 Congress on*, 3, 1944

Özcan, U, ve Toklu, B. (2009). Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. *Computers & Operations Research*, 36(6), 1955-1965.

Özcan, U. ve Toklu, B. (2010). Balancing Two-Sided Assembly Lines with Sequence-Dependent Setup Times. *International Journal of Production Research*, 48(18), 5363-5383.

Özçelik, F., (2018). Basit düz ve U-tipi montaj hattı dengeleme problemleri için diferansiyel evrim algoritması. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 24(1), 130-140.

Pastor, R. (2011). LB-ALBP: the lexicographic bottleneck assembly line balancing problem. *International Journal of Production Research*, 49(8), 2425-2442.

Pastor, R., Ignacio, C. and García-Villoria, A. (2012). A heuristic procedure for solving the Lexicographic Bottleneck Assembly Line Balancing Problem (LB-ALBP). *International Journal of Production Research*, 50(7), 1862-1876.

Petropoulos, D. and Nearchou, A. (2011). A particle swarm optimization algorithm for balancing assembly lines. *Assembly Automation*, 31, 118-129.

Pitakaso, P. and Sethanan, K. (2015). Differential Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types. *Engineering Optimization*, 48(2), 253-271.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1), 33-57.

Qiu, X. and Lau, H. (2014). An AIS-based hybrid algorithm for static job shop scheduling problem. *Journal of Intelligent Manufacturing*, 25, 489-503.

Rabbani, M., Alipour, F., Farrokhi-Asl, H. and Manavizadeh, N. (2018). Using metaheuristic algorithms for solving a mixed model assembly line balancing problem considering express parallel line and learning effect. *Brazilian Journal of Operations & Production Management*. 15, 254-269.

Rabbani, M., Mousavi, Z. and Farrokhi-Asl, H. (2016). Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem. *Journal of Industrial and Production Engineering (JIPE)*, 33, 1-13.

Rachamadugu, R. and Talbot, B. (1991). Improving the equality of workload assignments in assembly lines. *International Journal of Production Research*, 29, 619–633.

Raidl, G. (2006). *A unified view on hybrid metaheuristics*. Hybrid metaheuristics, in Lecture Notes in Computer Science, 4030, Springer,Berlin, Heidelberg, 1-12.

Robinson, J. and Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2), 397-407.

Salman A., Ahmad I. and Al-Madani S. (2002). Particle Swarm Optimization for Task Assignment Problem. *Microprocessors and Microsystems*, 26, 363-371.

Salveson, M. E. (1955). The Assembly Line Balancing Problem. *Journal of Industrial Engineering*, 6(3), 18-25.

Scholl, A. and Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3), 666–693.

Scholl, A. and Klein, R. (1999). ULINO: Optimally balancing U-shaped JIT assembly lines. *International Journal of Production Research*, 37(4), 721–736.

Scholl, A. and Voß, S. (1996). Simple Assembly Line Balancing - Heuristic Approaches. *Journal of Heuristics*, 2, 217-244.

Scholl, A., Boysen, N. and Fliedner, M. (2013). The Assembly Line Balancing and Scheduling Problem with Sequence-Dependent Setup Times: Problem Extension, Model Formulation and Efficient Heuristics. *OR Spectrum*, 35(1), 291-321.

Scholl, A., Klein, R., and Domschke, W. (1998). Pattern based vocabulary building for effectively sequencing mixed-model assembly lines. *Journal of Heuristics*, 4 (4), 359-381.

Sewell, E. and Jacobson, S. (2012). A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing*, 24(3), 433-442.

Seyed-Alagheband, S., Ghomi, S.M.T.F. and Zandieh, M., (2011). A simulated annealing algorithm for balancing theassembly line type II problem with sequence-dependent setup times between tasks. *International Journal of Production Research*, 49, 805-825.

Shi, Y. and Eberhart, R. (1999). Empirical Study of Particle Swarm Optimization. *Proceedings of Congress on Evolutionary Computation*, 3, 1945-1950.

Shi, Y. and Eberhart, R. C. (1998). *Parameter selection in particle swarm optimization*, Springer Berlin Heidelberg, Berlin.

Shtub, A. and Dar-El, E.M. (1989). Methodology for the selection of assembly systems. *International Journal of Production Research*, 27(1), 175–186.

Sivasankaran, P. and Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, 73(9-12), 1665-1694.

Storn, R. and Price, K., (1997). Differential evolution-A simple and efficient heuristic for global optimization over continues spaces. *Journal Global Optimization*, 11, 241–354.

Sun, J., Zhang, Q. and Tsang, E. P. K. (2005). DE/EDA: A new evolutionary algorithm for global optimization. *Information Sciences*, 169(3), 249–262.

Şahin, M. ve Kellegöz, T. (2019). A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations. *Computers & Industrial Engineering*, 133, 107-120.

Tapkan, P., Özbakir, L., ve Baykaşoglu, A. (2011). Bees algorithm for constrained fuzzy multiobjective two-sided assembly line balancing problem. *Optimization Letters*, 38(9), 11947- 11957.

Tempelmeier, H. (2003). Practical considerations in the optimization of flow production systems. *International Journal of Production Research*, 41(1), 149–170.

Toksarı, M. D., İşleyen, S. K. , Güner, E., ve Baykoç, Ö. F. (2008). Simple and U-type assembly line balancing problems with a learning effect. *Applied Mathematical Modelling*, 32, 2954-2961.

Toksarı, M. D., İşleyen, S. K. , Güner, E. ve Baykoç, Ö. F. (2010). Assembly Line Balancing Problem With Deterioration Tasks And Learning Effect. *Expert Systems With Applications*, 37, 1223-1228.

Uğurdağ, H.F., Rachamadugu, R. and Papachristou, C.A. (1997). Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research*, 102(3), 488–501.

Vincent, L. W. H. and Ponnambalam, S. G., (2013). A differential evolution-based algorithm to schedule flexible assembly lines. *IEEE Transactions On Automation Science And Engineering*, 10(4), 1161-1165.

White, W. W. (1961). Comment on a Paper by Bowman. *Operations Research*, 9(2), 274-276.

Yang, B., Chen, W. and Lin, C.. (2017). The Algorithm and Simulation of Multi-Objective Sequence and Balancing Problem for Mixed Mode Assembly Line. *International Journal of Simulation Modelling*, 16, 357-367.

Yolmeh, A. and Kianfar, F. (2011). An Efficient Hybrid Genetic Algorithm to Solve Assembly Line Balancing Problem with Sequence-Dependent Setup Times, *Computers & Industrial Engineering*, 62(4), 936–945.

Yuguang, Z., Bo, A. and Yong, Z. (2016). A PSO algorithm for multi-objective hull assembly line balancing using the stratified optimization strategy. *Computers & Industrial Engineering*, 98, 53-62.

Zhang, H., Yan, Q., Liu, Y. and Jiang, Z., (2016). An integer-coded differential evolution algorithm for simple assembly line balancing problem of type 2. *Assembly Automation*, 36(3), 246-261.

Zülch, G., Braun, W.J. and Schiller, E.F. (1997). Analytical approach of determining job division in manual assembly systems. *International Journal of Production Economics* 51(1- 2), 123–134.

EKLER

Ek 1: Geliştirilmiş Diferansiyel Gelişim Algoritmasi ve Hibrit Algoritma Sonuçları ($\alpha = 1,00$ Veri Seti için)

Problem Adı	Alt Sınır (Lower Bound)	Bulunan En İyi İstasyon Sayısı (GDGA)	Bulunan En İyi İstasyon Sayısı (HA)	CPU Zamanı (sn) (GDGA)	CPU Zamanı (sn) (HA)
Arc111_c=10027	16	22	21	51,96	16,40
Arc111_c=10743	15	21	20	50,71	16,16
Arc111_c=11378	14	19	18	52,54	15,82
Arc111_c=11570	14	19	19	50,03	15,75
Arc111_c=17067	9	13	12	47,05	15,00
Arc111_c=5755	27	29	29	56,67	17,24
Arc111_c=5785	27	29	29	54,89	17,10
Arc111_c=6016	26	30	30	56,90	17,12
Arc111_c=6267	25	32	31	56,89	17,49
Arc111_c=6540	24	32	31	55,96	17,74
Arc111_c=6837	23	33	32	56,83	17,63
Arc111_c=7162	22	32	31	59,85	17,55
Arc111_c=7520	21	31	30	58,45	17,25
Arc111_c=7916	20	29	29	55,41	16,95
Arc111_c=8356	19	27	27	58,29	16,65
Arc111_c=8847	18	25	24	53,46	16,57
Arc111_c=9400	17	24	23	52,61	16,35
Arc83_c=10816	8	10	10	36,24	11,20
Arc83_c=3786	21	23	23	40,70	12,73
Arc83_c=3985	20	24	24	42,11	12,67
Arc83_c=4206	19	24	24	43,57	12,87
Arc83_c=4454	18	25	25	42,05	13,12
Arc83_c=4732	17	25	24	42,52	13,09
Arc83_c=5048	16	23	22	40,82	12,56
Arc83_c=5408	15	21	20	39,90	12,53
Arc83_c=5824	14	19	19	39,45	12,33
Arc83_c=5853	14	19	18	38,72	12,33
Arc83_c=6309	13	18	17	38,87	11,99
Arc83_c=6842	12	16	16	37,71	11,89

Arc83_c=6883	12	16	16	37,78	11,75
Arc83_c=7571	11	15	14	37,46	11,63
Arc83_c=8412	10	13	13	36,84	11,52
Arc83_c=8898	9	12	12	36,10	11,44
Tonge70_c=160	23	25	25	40,57	11,69
Tonge70_c=168	22	26	25	39,27	11,92
Tonge70_c=176	21	26	26	37,68	11,86
Tonge70_c=185	20	27	26	40,98	11,77
Tonge70_c=195	19	27	26	39,77	11,85
Tonge70_c=207	18	28	25	38,46	11,82
Tonge70_c=220	17	25	24	39,62	11,50
Tonge70_c=234	16	23	23	37,22	11,28
Tonge70_c=251	15	21	20	36,06	11,07
Tonge70_c=270	14	19	18	35,39	10,87
Tonge70_c=293	13	17	17	35,78	10,69
Tonge70_c=320	12	16	15	33,20	10,35
Tonge70_c=364	10	14	14	32,26	10,32
Tonge70_c=410	9	12	12	31,62	10,06
Tonge70_c=468	8	11	11	31,74	9,86
Tonge70_c=527	7	10	9	30,56	9,86
barthol2_c=101	45	63	61	100,26	28,17
barthol2_c=104	44	63	61	95,14	27,08
barthol2_c=106	43	63	61	100,63	26,51
barthol2_c=109	42	64	61	97,70	27,11
barthol2_c=112	41	63	60	96,40	26,87
barthol2_c=115	40	62	58	87,61	26,11
barthol2_c=118	39	60	56	88,50	26,10
barthol2_c=121	38	58	54	91,66	26,71
barthol2_c=125	36	55	54	115,69	25,45
barthol2_c=129	35	53	48	93,94	31,40
barthol2_c=133	34	51	48	88,41	28,22
barthol2_c=137	33	50	47	101,53	30,30
barthol2_c=142	32	48	45	89,11	26,01
barthol2_c=146	31	46	44	106,83	34,44
barthol2_c=152	30	43	41	81,16	26,71
barthol2_c=157	29	41	40	87,20	25,78
barthol2_c=163	28	39	38	91,64	25,92
barthol2_c=170	27	38	36	90,64	24,08
barthol2_c=84	53	58	57	101,06	26,91

barthol2_c=85	52	58	56	96,15	27,19
barthol2_c=87	51	59	57	98,85	27,15
barthol2_c=89	50	60	58	106,38	27,52
barthol2_c=91	49	59	59	100,03	27,91
barthol2_c=93	48	60	58	100,48	27,45
barthol2_c=95	47	60	59	95,23	28,08
barthol2_c=97	46	62	60	104,13	28,20
barthol2_c=99	46	62	59	105,91	27,84
barthold_c=403	14	14	14	66,16	21,43
barthold_c=434	13	15	14	68,15	21,79
barthold_c=470	12	13	13	69,41	21,03
barthold_c=513	11	13	12	65,52	21,10
barthold_c=564	10	11	11	64,67	21,08
barthold_c=626	9	10	10	69,55	21,21
barthold_c=705	8	9	9	64,57	20,74
barthold_c=805	7	8	8	63,92	20,79
bowman8_c=20	5	6	6	6,86	3,26
buxey_c=27	14	14	14	17,64	6,32
buxey_c=30	13	15	15	17,98	6,38
buxey_c=33	12	15	15	17,50	6,29
buxey_c=36	11	15	15	17,99	6,66
buxey_c=41	9	12	12	16,50	6,02
buxey_c=47	8	10	10	16,22	5,73
buxey_c=54	7	9	8	15,66	5,49
gunther_c=41	13	14	14	19,90	6,90
gunther_c=44	12	14	14	20,02	6,89
gunther_c=49	11	15	15	20,11	7,06
gunther_c=54	10	15	14	20,35	6,95
gunther_c=61	9	12	12	18,53	6,60
gunther_c=69	8	10	10	17,44	6,42
gunther_c=81	7	9	9	16,92	6,26
hahn_c=2004	8	11	11	24,35	8,70
hahn_c=2338	7	9	9	23,67	8,24
hahn_c=2806	6	8	8	23,27	8,11
hahn_c=3507	5	6	6	22,67	7,91
hahn_c=4676	4	5	5	22,36	8,15
heskia_c=138	8	11	11	15,96	5,91
heskia_c=205	6	7	7	14,18	5,47
heskia_c=216	5	7	7	14,11	5,29

heskia_c=256	5	6	6	13,62	5,38
heskia_c=324	4	5	4	13,12	5,15
heskia_c=342	4	4	4	12,90	5,06
jackson_c=10	7	8	8	8,53	3,68
jackson_c=13	5	6	6	8,00	3,55
jackson_c=14	5	5	5	7,51	3,46
jackson_c=21	3	3	3	6,73	3,22
jackson_c=7	9	9	9	8,89	3,77
jackson_c=9	7	8	8	8,64	3,74
jaeschke_c=10	6	8	8	7,71	3,45
jaeschke_c=18	3	4	4	6,36	3,40
jaeschke_c=6	8	9	9	7,50	3,19
jaeschke_c=7	7	8	8	7,72	3,44
jaeschke_c=8	7	8	8	7,61	3,45
kilbrid_c=110	6	7	7	20,81	7,24
kilbrid_c=111	6	7	7	20,15	6,84
kilbrid_c=138	5	6	6	19,45	6,93
kilbrid_c=184	4	4	4	19,28	6,77
kilbrid_c=56	11	11	11	21,63	7,38
kilbrid_c=57	11	11	11	21,51	7,51
kilbrid_c=62	10	12	11	22,45	7,38
kilbrid_c=69	9	12	11	22,04	7,51
kilbrid_c=79	8	10	10	21,77	7,15
kilbrid_c=92	7	9	8	20,91	6,88
lutz1_c=1414	11	11	11	17,13	6,18
lutz1_c=1572	10	12	12	17,08	6,42
lutz1_c=1768	9	12	12	17,38	6,10
lutz1_c=2020	8	12	12	17,61	6,22
lutz1_c=2357	7	9	9	16,34	5,83
lutz1_c=2828	6	7	8	15,91	5,65
lutz2_c=11	53	57	56	59,57	17,43
lutz2_c=12	49	57	55	55,64	17,52
lutz2_c=13	46	58	55	55,78	17,88
lutz2_c=14	43	56	56	56,15	17,56
lutz2_c=15	41	57	55	56,13	17,57
lutz2_c=16	38	54	53	55,18	17,37
lutz2_c=17	36	50	49	56,04	16,87
lutz2_c=18	34	45	44	52,75	22,76
lutz2_c=19	32	43	41	52,68	20,99

lutz2_c=20	30	38	37	50,76	17,34
lutz2_c=21	28	36	35	49,60	15,41
lutz3_c=103	18	24	23	45,22	13,88
lutz3_c=110	17	22	22	44,20	14,44
lutz3_c=118	15	21	20	42,78	13,49
lutz3_c=127	14	19	19	42,04	13,40
lutz3_c=137	13	17	17	43,09	13,14
lutz3_c=150	12	16	16	40,18	13,44
lutz3_c=75	24	25	25	44,16	14,06
lutz3_c=79	23	25	25	46,26	14,68
lutz3_c=83	22	26	26	46,41	14,73
lutz3_c=87	21	27	26	45,55	14,39
lutz3_c=92	20	27	28	47,48	14,72
lutz3_c=97	19	25	25	46,92	14,48
mansoor_c=48	5	5	5	7,61	3,56
mansoor_c=62	4	5	5	8,06	4,02
mansoor_c=94	3	3	3	6,90	3,47
mertens_c=10	4	6	6	6,41	3,14
mertens_c=15	3	3	3	5,48	2,98
mertens_c=18	2	3	3	5,49	3,00
mertens_c=6	6	6	6	6,35	3,15
mertens_c=7	6	6	6	6,31	3,17
mertens_c=8	5	6	6	6,20	3,24
mitchell_c=14	9	10	9	12,53	5,12
mitchell_c=15	9	10	10	12,48	5,06
mitchell_c=21	7	8	8	12,13	4,82
mitchell_c=26	5	6	6	11,18	4,62
mitchell_c=35	4	5	4	10,88	4,42
mitchell_c=39	4	4	4	10,56	4,48
mukherje_c=176	25	27	27	45,83	14,92
mukherje_c=183	24	27	27	45,96	15,17
mukherje_c=192	23	28	28	46,29	15,91
mukherje_c=201	22	29	28	47,80	17,07
mukherje_c=211	21	29	28	47,52	16,09
mukherje_c=222	20	28	27	47,26	14,91
mukherje_c=234	19	27	26	46,60	14,88
mukherje_c=248	18	25	25	45,66	14,92
mukherje_c=263	17	24	24	45,00	14,47
mukherje_c=281	16	23	22	44,72	14,36

mukherje_c=301	15	20	20	42,98	13,99
mukherje_c=324	14	19	18	42,30	14,20
mukherje_c=351	13	17	17	42,13	13,55
roszieg_c=14	11	11	11	14,22	5,36
roszieg_c=16	10	11	11	14,38	5,42
roszieg_c=18	9	11	11	14,20	5,44
roszieg_c=21	8	9	9	13,66	5,29
roszieg_c=25	7	7	7	12,97	5,47
roszieg_c=32	5	6	6	12,63	5,59
sawyer30_c=25	15	15	15	17,75	6,14
sawyer30_c=27	14	14	14	17,39	6,16
sawyer30_c=30	13	14	14	18,67	6,13
sawyer30_c=33	12	14	14	17,66	6,19
sawyer30_c=36	11	14	14	17,58	6,09
sawyer30_c=41	9	12	12	16,83	5,88
sawyer30_c=47	8	10	10	16,22	5,62
sawyer30_c=54	7	9	9	15,55	5,57
sawyer30_c=75	5	6	6	14,65	5,26
scholl_c=1394	51	54	54	240,41	56,07
scholl_c=1422	50	56	56	219,82	58,16
scholl_c=1452	49	58	58	232,48	58,87
scholl_c=1483	48	59	59	228,20	57,60
scholl_c=1515	47	61	60	215,41	56,86
scholl_c=1548	46	63	61	290,50	57,08
scholl_c=1584	45	64	62	227,35	57,60
scholl_c=1620	44	65	63	248,27	58,93
scholl_c=1659	43	63	61	255,08	56,98
scholl_c=1699	42	62	60	243,17	57,48
scholl_c=1742	41	60	59	264,20	57,46
scholl_c=1787	40	58	57	195,10	57,87
scholl_c=1834	39	56	56	210,76	55,61
scholl_c=1883	38	55	55	187,98	56,61
scholl_c=1935	37	54	52	214,66	55,62
scholl_c=1991	36	52	51	174,05	60,13
scholl_c=2049	35	50	49	187,59	56,11
scholl_c=2111	34	48	47	187,69	55,74
scholl_c=2177	33	47	46	195,36	56,22
scholl_c=2247	32	44	44	185,48	54,80
scholl_c=2322	31	43	42	193,59	54,43

scholl_c=2402	30	42	41	174,98	54,40
scholl_c=2488	29	40	39	219,51	55,25
scholl_c=2580	28	38	38	186,11	54,16
scholl_c=2680	27	37	36	161,33	54,01
scholl_c=2787	26	36	35	162,06	54,36
warnecke_c=104	16	24	23	32,60	10,11
warnecke_c=111	15	22	22	31,60	10,10
warnecke_c=54	31	33	32	35,75	10,89
warnecke_c=56	29	33	32	35,98	11,01
warnecke_c=58	29	33	33	35,72	11,00
warnecke_c=60	28	34	34	36,32	11,23
warnecke_c=62	27	34	34	36,64	11,21
warnecke_c=65	26	35	33	36,70	11,42
warnecke_c=68	25	35	33	36,42	11,57
warnecke_c=71	24	34	34	36,74	11,24
warnecke_c=74	23	34	34	37,14	11,28
warnecke_c=78	22	34	34	37,59	11,35
warnecke_c=82	21	34	32	37,33	11,22
warnecke_c=86	20	31	30	36,18	10,92
warnecke_c=92	19	28	27	38,91	10,55
warnecke_c=97	18	26	25	34,56	10,38
wee-mag_c=28	63	64	64	52,37	16,29
wee-mag_c=29	63	64	63	52,85	15,79
wee-mag_c=30	62	64	63	53,29	16,06
wee-mag_c=31	62	64	63	52,25	15,90
wee-mag_c=32	61	64	63	51,44	15,72
wee-mag_c=33	61	64	64	55,00	15,82
wee-mag_c=34	61	64	64	51,72	16,09
wee-mag_c=35	60	64	64	50,38	16,04
wee-mag_c=36	60	64	63	50,23	15,75
wee-mag_c=37	60	64	64	51,52	15,64
wee-mag_c=38	60	64	64	51,35	15,92
wee-mag_c=39	60	64	64	51,20	15,74
wee-mag_c=40	60	64	64	50,72	15,78
wee-mag_c=41	59	64	63	51,73	15,71
wee-mag_c=42	55	64	63	51,33	15,93
wee-mag_c=43	50	64	64	52,81	15,93
wee-mag_c=45	40	64	64	51,31	16,12
wee-mag_c=46	39	64	64	51,03	15,94

wee-mag_c=47	38	64	63	53,56	16,04
wee-mag_c=49	36	63	63	51,16	15,92
wee-mag_c=50	35	62	62	50,99	15,97
wee-mag_c=52	34	61	61	49,76	15,77
wee-mag_c=54	32	60	60	50,51	15,50
wee-mag_c=56	31	60	60	49,21	15,63

Ek 2: Geliştirilmiş Diferansiyel Gelişim Algoritmasi ve Hibrit Algoritma Sonuçları ($\alpha = 0,75$ Veri Seti için)

Problem Adı	Alt Sınır (Lower Bound)	Bulunan En İyi İstasyon Sayısı (GDGA)	Bulunan En İyi İstasyon Sayısı (HA)	CPU Zamanı (sn) (GDGA)	CPU Zamanı (sn) (HA)
Arc111_c=10027	16	21	20	51,58	15,45
Arc111_c=10743	15	19	19	50,89	15,50
Arc111_c=11378	14	18	17	52,75	15,05
Arc111_c=11570	14	18	17	53,08	15,01
Arc111_c=17067	9	12	12	48,62	14,28
Arc111_c=5755	27	29	29	58,05	16,50
Arc111_c=5785	27	30	29	55,93	16,75
Arc111_c=6016	26	31	30	56,28	16,36
Arc111_c=6267	25	32	31	56,59	17,01
Arc111_c=6540	24	33	31	57,89	17,07
Arc111_c=6837	23	34	32	58,27	16,91
Arc111_c=7162	22	33	31	57,45	16,75
Arc111_c=7520	21	32	30	57,62	16,35
Arc111_c=7916	20	29	28	55,94	16,27
Arc111_c=8356	19	28	26	55,39	16,07
Arc111_c=8847	18	26	24	56,57	15,82
Arc111_c=9400	17	24	23	53,29	15,94
Arc83_c=10816	8	10	10	38,59	10,78
Arc83_c=3786	21	32	31	45,07	13,23
Arc83_c=3985	20	22	21	40,58	12,47
Arc83_c=4206	19	25	24	42,76	12,37
Arc83_c=4454	18	26	25	45,31	12,44
Arc83_c=4732	17	25	24	47,71	12,38
Arc83_c=5048	16	23	23	44,24	12,19
Arc83_c=5408	15	22	20	43,76	11,86
Arc83_c=5824	14	20	19	43,87	11,88
Arc83_c=5853	14	20	19	43,43	11,85
Arc83_c=6309	13	18	17	42,24	11,61

Arc83_c=6842	12	16	16	40,40	11,55
Arc83_c=6883	12	16	16	39,59	11,50
Arc83_c=7571	11	15	14	38,69	11,36
Arc83_c=8412	10	13	13	38,15	11,19
Arc83_c=8898	9	12	12	37,64	11,03
Tonge70_c=160	23	25	25	37,22	11,17
Tonge70_c=168	22	26	25	38,35	11,79
Tonge70_c=176	21	28	25	38,13	11,91
Tonge70_c=185	20	28	26	38,81	11,82
Tonge70_c=195	19	28	26	38,90	12,09
Tonge70_c=207	18	25	24	37,87	11,56
Tonge70_c=220	17	24	23	37,65	11,34
Tonge70_c=234	16	21	21	36,13	11,27
Tonge70_c=251	15	20	19	36,47	10,98
Tonge70_c=270	14	19	18	35,82	10,88
Tonge70_c=293	13	17	16	34,02	10,86
Tonge70_c=320	12	16	15	33,12	10,47
Tonge70_c=364	10	13	13	33,27	10,23
Tonge70_c=410	9	12	11	32,17	10,15
Tonge70_c=468	8	10	10	31,38	9,81
Tonge70_c=527	7	9	9	30,92	9,51
barthol2_c=101	45	65	60	97,03	25,56
barthol2_c=104	44	66	60	92,49	25,63
barthol2_c=106	43	64	58	92,18	25,78
barthol2_c=109	42	62	58	89,72	25,06
barthol2_c=112	40	60	54	89,68	25,47
barthol2_c=115	39	57	53	91,89	25,85
barthol2_c=118	38	56	51	89,17	24,79
barthol2_c=121	37	54	50	91,34	24,67
barthol2_c=125	36	51	48	87,93	24,13
barthol2_c=129	35	50	46	87,84	25,28
barthol2_c=133	34	48	45	87,05	23,73
barthol2_c=137	33	46	43	84,20	23,72
barthol2_c=142	32	45	41	85,91	23,96
barthol2_c=146	31	43	40	82,26	23,15
barthol2_c=152	30	40	38	82,57	23,28
barthol2_c=157	29	39	37	80,63	22,91
barthol2_c=163	28	37	35	80,30	22,63
barthol2_c=170	26	35	34	80,74	22,21

barthol2_c=84	53	60	57	93,49	25,55
barthol2_c=85	52	59	56	90,49	24,77
barthol2_c=87	51	61	57	92,40	25,28
barthol2_c=89	50	61	58	91,32	24,97
barthol2_c=91	49	62	59	92,04	25,50
barthol2_c=93	48	63	59	93,81	25,24
barthol2_c=95	47	63	59	95,08	25,70
barthol2_c=97	46	64	60	103,48	25,59
barthol2_c=99	46	64	60	99,19	25,21
barthold_c=403	14	14	14	68,29	19,71
barthold_c=434	13	14	14	68,63	19,92
barthold_c=470	12	13	12	67,74	20,12
barthold_c=513	11	12	11	67,35	19,52
barthold_c=564	10	11	10	66,99	19,61
barthold_c=626	9	10	9	66,89	19,59
barthold_c=705	8	8	8	66,46	19,56
barthold_c=805	7	7	7	66,73	19,33
bowman8_c=20	5	6	6	6,63	3,08
buxey_c=27	14	14	14	16,77	5,83
buxey_c=30	13	14	15	17,81	5,95
buxey_c=33	12	14	15	17,62	5,83
buxey_c=36	11	13	13	16,84	5,85
buxey_c=41	9	11	11	16,41	5,51
buxey_c=47	8	10	9	15,69	5,26
buxey_c=54	7	8	8	14,98	5,23
gunther_c=41	13	14	14	20,30	6,50
gunther_c=44	12	14	14	20,17	6,47
gunther_c=49	11	15	14	20,40	6,55
gunther_c=54	10	13	13	19,75	6,34
gunther_c=61	9	11	11	19,20	6,10
gunther_c=69	8	10	10	17,52	5,91
gunther_c=81	7	8	8	17,34	5,75
hahn_c=2004	8	11	11	24,98	7,92
hahn_c=2338	7	9	9	24,13	7,71
hahn_c=2806	6	8	8	23,80	7,69
hahn_c=3507	5	6	6	23,17	7,45
hahn_c=4676	4	5	5	22,84	7,38
heskia_c=138	8	11	10	15,47	5,51
heskia_c=205	6	7	7	14,05	5,10

heskia_c=216	5	7	6	13,98	5,09
heskia_c=256	5	6	5	13,44	5,06
heskia_c=324	4	4	4	13,07	4,70
heskia_c=342	4	4	4	12,88	4,75
jackson_c=10	6	7	7	7,96	3,52
jackson_c=13	5	5	5	7,48	3,36
jackson_c=14	4	4	4	7,52	3,21
jackson_c=21	3	3	3	6,86	3,33
jackson_c=7	9	9	9	8,39	3,59
jackson_c=9	7	8	8	8,28	3,62
jaeschke_c=10	6	7	7	7,12	3,30
jaeschke_c=18	3	3	3	6,10	3,04
jaeschke_c=6	8	9	9	7,38	3,02
jaeschke_c=7	7	8	8	7,40	3,25
jaeschke_c=8	7	8	8	7,21	3,25
kilbrid_c=110	6	7	7	20,48	6,76
kilbrid_c=111	6	7	7	19,98	6,54
kilbrid_c=138	5	5	5	19,66	6,31
kilbrid_c=184	4	4	4	19,17	6,25
kilbrid_c=56	11	11	11	21,96	7,04
kilbrid_c=57	11	11	11	21,62	6,99
kilbrid_c=62	10	12	11	22,07	7,13
kilbrid_c=69	9	11	11	21,80	7,10
kilbrid_c=79	8	10	9	21,92	6,84
kilbrid_c=92	7	8	8	21,04	6,66
lutz1_c=1414	11	11	11	16,92	5,82
lutz1_c=1572	10	12	12	17,12	5,90
lutz1_c=1768	9	12	12	16,92	5,81
lutz1_c=2020	8	10	10	16,93	5,62
lutz1_c=2357	7	8	9	16,04	5,40
lutz1_c=2828	6	7	7	15,51	5,36
lutz2_c=11	53	58	55	56,39	16,82
lutz2_c=12	49	57	56	55,73	16,65
lutz2_c=13	46	58	55	56,37	16,70
lutz2_c=14	43	58	56	56,50	16,70
lutz2_c=15	40	54	52	54,93	16,59
lutz2_c=16	37	51	48	54,64	16,03
lutz2_c=17	35	47	45	52,83	15,45
lutz2_c=18	33	44	41	52,10	14,89

lutz2_c=19	31	39	37	50,49	14,78
lutz2_c=20	30	36	34	49,49	14,39
lutz2_c=21	28	34	32	49,31	14,24
lutz3_c=103	18	23	23	44,28	13,12
lutz3_c=110	16	21	21	43,82	12,87
lutz3_c=118	15	19	18	43,38	12,44
lutz3_c=127	14	18	17	42,27	12,35
lutz3_c=137	13	16	16	41,62	12,25
lutz3_c=150	12	15	15	41,18	12,04
lutz3_c=75	24	25	24	45,52	13,38
lutz3_c=79	23	25	26	45,86	13,55
lutz3_c=83	22	26	26	46,06	13,45
lutz3_c=87	21	28	27	46,41	13,60
lutz3_c=92	20	25	25	45,97	13,39
lutz3_c=97	19	24	23	45,07	13,27
mansoor_c=48	5	5	5	7,44	3,39
mansoor_c=62	4	4	4	7,10	3,22
mansoor_c=94	3	3	3	6,80	3,24
mertens_c=10	4	5	5	5,95	3,01
mertens_c=15	3	3	3	5,42	2,88
mertens_c=18	2	2	2	5,12	2,93
mertens_c=6	6	6	6	6,23	3,02
mertens_c=7	6	6	6	6,18	3,06
mertens_c=8	5	6	6	6,15	3,02
mitchell_c=14	9	9	9	12,57	4,63
mitchell_c=15	9	10	10	12,36	4,71
mitchell_c=21	7	7	7	11,51	4,49
mitchell_c=26	5	6	6	11,34	4,38
mitchell_c=35	4	4	4	10,62	4,23
mitchell_c=39	4	4	4	10,60	4,18
mukherje_c=176	25	27	27	48,91	13,93
mukherje_c=183	24	28	27	52,33	14,06
mukherje_c=192	23	28	28	56,78	14,08
mukherje_c=201	22	31	30	56,66	14,27
mukherje_c=211	21	28	27	55,97	14,17
mukherje_c=222	20	27	26	46,74	14,00
mukherje_c=234	19	26	25	46,22	13,78
mukherje_c=248	18	25	24	45,59	13,53
mukherje_c=263	17	23	22	45,23	13,81

mukherje_c=281	16	20	20	44,50	13,23
mukherje_c=301	15	20	19	43,64	12,84
mukherje_c=324	14	18	17	42,96	12,83
mukherje_c=351	13	17	16	42,32	12,67
roszieg_c=14	11	11	11	14,58	5,27
roszieg_c=16	10	11	11	14,59	5,18
roszieg_c=18	9	10	10	14,03	5,05
roszieg_c=21	8	8	8	12,87	4,95
roszieg_c=25	7	7	7	13,24	4,78
roszieg_c=32	5	5	5	12,34	4,58
sawyer30_c=25	15	16	15	17,49	6,02
sawyer30_c=27	14	14	14	17,75	5,96
sawyer30_c=30	13	14	14	18,25	5,89
sawyer30_c=33	12	14	14	18,06	5,98
sawyer30_c=36	11	13	13	17,52	5,78
sawyer30_c=41	9	12	12	17,24	5,73
sawyer30_c=47	8	10	10	17,07	5,56
sawyer30_c=54	7	8	8	15,66	5,31
sawyer30_c=75	5	6	6	14,83	5,05
scholl_c=1394	51	55	54	169,75	55,69
scholl_c=1422	50	57	56	174,24	56,93
scholl_c=1452	49	59	57	183,17	56,94
scholl_c=1483	48	61	59	187,99	58,43
scholl_c=1515	47	62	59	199,84	56,11
scholl_c=1548	46	64	60	177,45	55,96
scholl_c=1584	45	63	60	178,24	56,85
scholl_c=1620	44	62	59	191,00	55,82
scholl_c=1659	43	60	58	173,29	55,94
scholl_c=1699	42	58	56	176,14	55,77
scholl_c=1742	41	57	55	172,68	56,94
scholl_c=1787	40	55	53	170,66	55,03
scholl_c=1834	39	53	52	168,27	55,67
scholl_c=1883	38	52	50	168,37	55,71
scholl_c=1935	37	50	49	174,53	55,76
scholl_c=1991	36	49	46	169,04	55,06
scholl_c=2049	35	47	46	171,21	54,75
scholl_c=2111	34	46	44	171,96	54,39
scholl_c=2177	33	44	43	170,22	54,81
scholl_c=2247	32	43	42	167,02	53,69

scholl_c=2322	31	41	40	166,94	54,83
scholl_c=2402	30	39	38	162,18	54,79
scholl_c=2488	29	38	37	162,37	53,79
scholl_c=2580	28	36	36	171,26	51,91
scholl_c=2680	27	35	34	162,96	53,77
scholl_c=2787	26	33	33	158,99	52,75
warnecke_c=104	16	22	21	31,36	9,64
warnecke_c=111	15	21	19	31,06	9,18
warnecke_c=54	31	33	32	34,83	10,67
warnecke_c=56	29	34	32	34,81	10,63
warnecke_c=58	29	33	33	35,13	10,88
warnecke_c=60	28	34	33	37,05	10,88
warnecke_c=62	27	35	34	35,66	10,99
warnecke_c=65	26	35	34	36,04	10,91
warnecke_c=68	25	35	34	36,84	10,91
warnecke_c=71	24	35	34	36,07	10,69
warnecke_c=74	23	35	34	35,50	10,89
warnecke_c=78	22	33	31	35,07	10,56
warnecke_c=82	21	31	30	34,86	10,48
warnecke_c=86	20	28	27	33,71	10,15
warnecke_c=92	18	26	25	32,66	9,86
warnecke_c=97	17	24	23	32,33	9,75
wee-mag_c=28	63	64	64	49,44	15,26
wee-mag_c=29	63	64	63	49,38	15,28
wee-mag_c=30	62	64	64	49,33	15,30
wee-mag_c=31	62	64	64	51,45	15,79
wee-mag_c=32	61	64	64	49,49	15,06
wee-mag_c=33	61	64	64	49,69	15,13
wee-mag_c=34	61	64	64	49,79	15,06
wee-mag_c=35	60	64	64	49,56	15,27
wee-mag_c=36	60	65	63	49,73	15,03
wee-mag_c=37	60	64	64	50,08	15,27
wee-mag_c=38	60	64	64	49,71	15,39
wee-mag_c=39	60	64	64	49,62	15,09
wee-mag_c=40	60	64	64	50,29	15,25
wee-mag_c=41	59	64	63	50,23	15,11
wee-mag_c=42	55	64	63	50,04	15,13
wee-mag_c=43	50	63	63	51,14	15,10
wee-mag_c=45	38	63	62	49,49	15,08

wee-mag_c=46	37	62	61	48,81	14,88
wee-mag_c=47	37	61	61	49,13	15,04
wee-mag_c=49	35	61	60	48,23	15,07
wee-mag_c=50	34	60	60	48,39	14,88
wee-mag_c=52	33	60	60	47,86	15,21
wee-mag_c=54	31	60	60	47,86	14,97
wee-mag_c=56	30	56	55	48,38	14,93

Ek 3: Geliştirilmiş Diferansiyel Gelişim Algoritmasi ve Hibrit Algoritma Sonuçları ($\alpha = 0,50$ Veri Seti için)

Problem Adı	Alt Sınır (Lower Bound)	Bulunan En İyi İstasyon Sayısı (GDGA)	Bulunan En İyi İstasyon Sayısı (HA)	CPU Zamanı (sn) (GDGA)	CPU Zamanı (sn) (HA)
Arc111_c=10027	16	19	19	56,89	17,31
Arc111_c=10743	15	18	18	56,03	16,48
Arc111_c=11378	14	17	17	53,54	15,68
Arc111_c=11570	14	17	17	52,36	15,74
Arc111_c=17067	9	11	11	50,17	15,10
Arc111_c=5755	27	29	29	59,46	17,58
Arc111_c=5785	27	30	29	59,38	17,60
Arc111_c=6016	26	32	30	59,26	17,82
Arc111_c=6267	25	32	31	60,56	17,99
Arc111_c=6540	24	31	30	60,71	17,77
Arc111_c=6837	23	30	29	59,19	17,42
Arc111_c=7162	22	28	27	57,29	17,25
Arc111_c=7520	21	27	26	57,27	17,06
Arc111_c=7916	20	25	24	56,06	17,08
Arc111_c=8356	19	24	23	57,71	16,83
Arc111_c=8847	18	22	22	55,03	16,46
Arc111_c=9400	17	21	20	55,28	16,23
Arc83_c=10816	8	9	9	40,31	11,41
Arc83_c=3786	21	23	23	48,99	12,87
Arc83_c=3985	20	24	23	47,12	13,03
Arc83_c=4206	19	25	24	46,64	12,91
Arc83_c=4454	18	22	22	46,33	12,79
Arc83_c=4732	17	21	21	45,41	12,68
Arc83_c=5048	16	20	19	45,97	12,47
Arc83_c=5408	15	18	18	43,09	12,59
Arc83_c=5824	14	17	16	41,44	12,32
Arc83_c=5853	14	17	16	41,33	12,10
Arc83_c=6309	13	15	15	40,45	12,02

Arc83_c=6842	12	14	14	39,57	11,94
Arc83_c=6883	12	14	14	39,95	11,92
Arc83_c=7571	11	13	13	39,72	11,82
Arc83_c=8412	10	11	11	39,03	11,53
Arc83_c=8898	9	11	11	38,49	11,51
Tonge70_c=160	23	26	25	39,46	12,31
Tonge70_c=168	22	27	25	39,67	12,63
Tonge70_c=176	21	27	26	40,14	12,59
Tonge70_c=185	20	27	25	40,80	12,50
Tonge70_c=195	19	25	24	39,73	12,26
Tonge70_c=207	18	23	22	38,91	11,95
Tonge70_c=220	17	22	20	38,03	11,84
Tonge70_c=234	16	20	20	37,64	11,85
Tonge70_c=251	15	19	18	36,36	11,30
Tonge70_c=270	14	17	17	35,78	11,08
Tonge70_c=293	13	16	15	35,57	11,03
Tonge70_c=320	12	14	14	34,89	10,95
Tonge70_c=364	10	13	12	33,71	10,62
Tonge70_c=410	9	11	11	34,03	10,08
Tonge70_c=468	8	10	9	33,09	9,81
Tonge70_c=527	7	8	8	32,75	9,75
barthol2_c=101	45	61	56	92,03	26,90
barthol2_c=104	43	59	54	95,08	26,01
barthol2_c=106	42	57	53	91,91	26,60
barthol2_c=109	41	56	52	96,87	26,05
barthol2_c=112	40	54	50	98,81	25,87
barthol2_c=115	39	52	50	88,89	25,51
barthol2_c=118	38	50	48	88,24	26,06
barthol2_c=121	37	49	45	89,43	25,83
barthol2_c=125	36	47	44	93,07	24,77
barthol2_c=129	35	46	43	89,24	24,88
barthol2_c=133	34	44	42	85,32	24,79
barthol2_c=137	33	42	40	86,86	24,36
barthol2_c=142	32	40	38	87,60	23,82
barthol2_c=146	31	39	37	85,24	24,85
barthol2_c=152	29	37	36	83,28	23,77
barthol2_c=157	29	36	34	81,97	23,53
barthol2_c=163	27	34	33	80,81	23,62
barthol2_c=170	26	33	31	84,76	23,22

barthol2_c=84	53	60	56	94,50	26,42
barthol2_c=85	52	59	56	92,17	26,58
barthol2_c=87	51	61	58	102,91	26,92
barthol2_c=89	50	62	58	100,02	26,91
barthol2_c=91	49	63	58	93,69	26,67
barthol2_c=93	48	63	59	96,32	26,52
barthol2_c=95	47	64	58	96,46	27,11
barthol2_c=97	46	64	59	97,59	27,20
barthol2_c=99	45	62	57	96,75	27,15
barthold_c=403	11	14	14	71,05	21,00
barthold_c=434	11	13	13	69,93	20,80
barthold_c=470	10	12	12	69,44	20,82
barthold_c=513	9	11	11	69,71	20,48
barthold_c=564	8	10	10	71,12	20,44
barthold_c=626	7	9	9	69,62	20,66
barthold_c=705	7	8	8	68,24	20,34
barthold_c=805	6	7	7	68,01	20,24
bowman8_c=20	5	6	6	6,90	3,26
buxey_c=27	14	14	14	17,79	6,34
buxey_c=30	13	14	15	18,66	6,34
buxey_c=33	11	13	13	17,72	6,12
buxey_c=36	10	12	12	17,28	6,04
buxey_c=41	9	11	10	16,33	5,76
buxey_c=47	8	9	9	16,28	5,67
buxey_c=54	7	8	7	15,28	5,63
gunther_c=41	13	14	14	21,11	7,00
gunther_c=44	12	15	14	21,83	6,88
gunther_c=49	11	13	13	20,24	6,79
gunther_c=54	10	12	11	20,52	6,56
gunther_c=61	9	10	10	18,79	6,27
gunther_c=69	8	9	9	18,11	6,19
gunther_c=81	7	8	8	17,80	6,15
hahn_c=2004	8	10	10	25,69	8,31
hahn_c=2338	7	8	8	25,20	8,19
hahn_c=2806	6	6	6	24,92	7,94
hahn_c=3507	5	5	5	23,97	7,84
hahn_c=4676	4	4	4	23,59	8,00
heskia_c=138	8	10	10	16,11	5,73
heskia_c=205	6	7	7	14,67	5,35

heskia_c=216	5	6	6	14,45	5,49
heskia_c=256	5	5	5	13,95	5,17
heskia_c=324	4	4	4	13,51	5,03
heskia_c=342	4	4	4	13,39	5,05
jackson_c=10	6	7	7	8,38	3,74
jackson_c=13	5	5	5	7,80	3,57
jackson_c=14	5	5	5	7,56	3,42
jackson_c=21	3	3	3	7,18	3,42
jackson_c=7	9	9	9	8,79	3,82
jackson_c=9	7	8	8	8,55	3,80
jaeschke_c=10	5	6	6	7,26	3,39
jaeschke_c=18	3	3	3	6,36	3,22
jaeschke_c=6	8	9	9	7,80	3,16
jaeschke_c=7	7	8	8	7,71	3,45
jaeschke_c=8	7	8	8	7,73	3,44
kilbrid_c=110	6	6	6	22,07	6,82
kilbrid_c=111	6	6	6	21,34	6,82
kilbrid_c=138	5	5	5	21,33	6,77
kilbrid_c=184	4	4	4	20,68	6,71
kilbrid_c=56	11	11	11	22,99	7,42
kilbrid_c=57	11	11	11	23,11	7,42
kilbrid_c=62	10	11	11	23,09	7,44
kilbrid_c=69	9	10	10	22,52	7,28
kilbrid_c=79	8	9	9	22,05	7,22
kilbrid_c=92	7	8	7	21,45	6,97
lutz1_c=1414	11	11	11	17,98	6,17
lutz1_c=1572	10	12	12	18,25	6,31
lutz1_c=1768	9	11	11	17,85	6,06
lutz1_c=2020	8	10	10	17,30	5,95
lutz1_c=2357	7	8	8	16,52	5,76
lutz1_c=2828	6	7	7	16,03	5,69
lutz2_c=11	53	59	56	59,83	17,74
lutz2_c=12	49	58	55	60,28	18,14
lutz2_c=13	45	53	51	58,71	17,35
lutz2_c=14	42	50	48	57,21	17,13
lutz2_c=15	39	45	43	55,39	16,35
lutz2_c=16	36	43	40	54,04	16,00
lutz2_c=17	34	37	35	52,42	15,24
lutz2_c=18	32	35	34	52,21	15,29

lutz2_c=19	31	32	31	50,55	14,83
lutz2_c=20	29	31	29	50,90	14,57
lutz2_c=21	28	29	28	49,65	14,52
lutz3_c=103	17	22	20	46,04	13,65
lutz3_c=110	16	20	19	45,40	13,35
lutz3_c=118	15	18	18	44,58	13,24
lutz3_c=127	14	17	16	43,65	12,98
lutz3_c=137	13	15	15	43,18	12,81
lutz3_c=150	12	14	14	42,71	12,69
lutz3_c=75	24	25	24	47,86	13,98
lutz3_c=79	23	26	25	49,34	14,15
lutz3_c=83	22	27	26	49,16	14,34
lutz3_c=87	21	25	24	48,62	14,12
lutz3_c=92	20	24	23	46,83	14,09
lutz3_c=97	19	22	22	46,14	13,91
mansoor_c=48	5	5	5	7,75	3,58
mansoor_c=62	4	4	4	7,50	3,42
mansoor_c=94	3	3	3	7,10	3,43
mertens_c=10	4	4	4	6,06	2,98
mertens_c=15	3	3	3	5,66	3,03
mertens_c=18	2	2	2	5,25	2,90
mertens_c=6	6	6	6	6,44	3,24
mertens_c=7	6	6	6	6,46	3,24
mertens_c=8	5	6	6	6,46	3,23
mitchell_c=14	9	10	9	13,16	5,05
mitchell_c=15	9	10	10	13,12	5,03
mitchell_c=21	6	7	6	11,79	4,59
mitchell_c=26	5	5	5	11,30	4,55
mitchell_c=35	4	4	4	10,91	4,45
mitchell_c=39	4	4	4	10,93	4,36
mukherje_c=176	25	27	27	49,65	14,71
mukherje_c=183	24	28	27	50,09	15,06
mukherje_c=192	23	29	28	50,14	15,30
mukherje_c=201	22	28	27	49,86	14,96
mukherje_c=211	21	26	26	48,51	14,66
mukherje_c=222	20	25	25	47,98	14,64
mukherje_c=234	19	24	23	47,82	14,35
mukherje_c=248	18	23	22	47,52	14,32
mukherje_c=263	17	21	20	46,96	13,84

mukherje_c=281	16	20	19	47,25	13,75
mukherje_c=301	15	18	18	46,03	13,60
mukherje_c=324	14	17	16	44,49	13,57
mukherje_c=351	13	15	15	44,45	13,22
roszieg_c=14	11	11	11	15,55	5,45
roszieg_c=16	10	11	11	14,87	5,53
roszieg_c=18	9	9	9	14,32	5,32
roszieg_c=21	8	8	8	13,71	5,08
roszieg_c=25	6	6	6	13,33	4,99
roszieg_c=32	5	5	5	12,78	4,89
sawyer30_c=25	15	16	15	18,42	6,31
sawyer30_c=27	14	15	14	18,26	6,36
sawyer30_c=30	13	15	15	19,13	6,29
sawyer30_c=33	11	13	13	18,21	6,17
sawyer30_c=36	10	12	12	18,01	6,01
sawyer30_c=41	9	11	10	16,92	5,90
sawyer30_c=47	8	9	9	16,51	5,76
sawyer30_c=54	7	8	8	16,17	5,67
sawyer30_c=75	5	6	5	15,70	5,46
scholl_c=1394	51	55	54	184,09	58,58
scholl_c=1422	50	57	56	178,74	59,16
scholl_c=1452	49	59	57	192,92	58,92
scholl_c=1483	48	61	58	192,18	59,56
scholl_c=1515	47	61	58	200,12	60,59
scholl_c=1548	46	60	58	200,18	60,33
scholl_c=1584	45	58	56	181,50	59,13
scholl_c=1620	44	56	55	180,34	58,35
scholl_c=1659	43	55	53	175,12	59,38
scholl_c=1699	42	54	52	180,36	57,72
scholl_c=1742	41	52	50	174,58	57,76
scholl_c=1787	40	51	49	174,67	57,03
scholl_c=1834	39	49	48	172,71	57,74
scholl_c=1883	38	48	47	173,55	57,31
scholl_c=1935	37	47	45	174,49	55,72
scholl_c=1991	36	45	44	174,26	57,25
scholl_c=2049	35	43	42	173,27	56,34
scholl_c=2111	34	42	41	171,73	57,03
scholl_c=2177	33	41	40	169,89	54,91
scholl_c=2247	32	39	39	169,81	56,15

scholl_c=2322	31	38	37	168,38	55,19
scholl_c=2402	30	36	36	171,03	56,10
scholl_c=2488	29	35	35	166,31	54,93
scholl_c=2580	28	34	34	166,05	54,29
scholl_c=2680	27	33	32	166,31	53,90
scholl_c=2787	26	31	31	166,26	55,66
warnecke_c=104	16	21	20	32,97	9,93
warnecke_c=111	15	19	18	33,87	9,62
warnecke_c=54	30	33	32	37,82	11,37
warnecke_c=56	29	33	33	37,82	11,46
warnecke_c=58	29	34	33	36,89	11,47
warnecke_c=60	28	35	33	37,54	11,80
warnecke_c=62	27	35	34	37,57	11,54
warnecke_c=65	26	36	34	38,00	11,56
warnecke_c=68	25	34	33	37,53	11,34
warnecke_c=71	24	33	31	36,70	11,18
warnecke_c=74	23	31	30	36,31	11,18
warnecke_c=78	22	29	28	36,40	10,76
warnecke_c=82	20	27	26	36,11	10,69
warnecke_c=86	20	25	25	35,10	10,62
warnecke_c=92	18	23	22	34,34	10,37
warnecke_c=97	17	22	21	33,63	10,24
wee-mag_c=28	63	64	64	53,48	16,64
wee-mag_c=29	63	64	63	53,71	16,42
wee-mag_c=30	62	64	64	53,07	16,27
wee-mag_c=31	62	65	64	53,23	16,35
wee-mag_c=32	61	64	64	54,93	16,53
wee-mag_c=33	61	64	63	55,89	16,72
wee-mag_c=34	61	64	64	54,24	16,04
wee-mag_c=35	60	64	63	53,04	16,33
wee-mag_c=36	60	65	64	53,51	16,44
wee-mag_c=37	60	64	63	52,64	16,40
wee-mag_c=38	60	64	63	52,70	16,23
wee-mag_c=39	60	63	62	52,54	16,15
wee-mag_c=40	60	62	62	54,47	16,33
wee-mag_c=41	59	62	61	52,54	16,06
wee-mag_c=42	55	61	61	53,15	16,23
wee-mag_c=43	50	61	61	51,59	16,22
wee-mag_c=45	38	60	60	51,40	16,26

wee-mag_c=46	36	60	60	51,04	16,18
wee-mag_c=47	35	60	60	51,22	16,03
wee-mag_c=49	34	60	60	50,80	16,06
wee-mag_c=50	33	59	59	51,34	15,98
wee-mag_c=52	32	52	50	52,13	15,07
wee-mag_c=54	31	45	41	49,83	14,42
wee-mag_c=56	30	40	35	46,35	13,50

Ek 4: Geliştirilmiş Diferansiyel Gelişim Algoritmasi ve Hibrit Algoritma Sonuçları ($\alpha = 0,25$ Veri Seti için)

Problem Adı	Alt Sınır (Lower Bound)	Bulunan En İyi İstasyon Sayısı (GDGA)	Bulunan En İyi İstasyon Sayısı (HA)	CPU Zamanı (sn) (GDGA)	CPU Zamanı (sn) (HA)
Arc111_c=10027	16	18	17	73,32	15,78
Arc111_c=10743	15	17	16	48,00	20,04
Arc111_c=11378	14	15	15	47,81	16,04
Arc111_c=11570	14	15	15	47,66	14,91
Arc111_c=17067	9	10	10	47,13	15,82
Arc111_c=5755	27	29	29	54,18	16,68
Arc111_c=5785	27	30	29	54,57	16,64
Arc111_c=6016	26	32	30	55,35	16,72
Arc111_c=6267	25	30	29	55,03	16,62
Arc111_c=6540	24	28	28	52,96	16,39
Arc111_c=6837	23	27	26	52,95	16,20
Arc111_c=7162	22	25	25	54,93	15,97
Arc111_c=7520	21	24	24	51,73	16,04
Arc111_c=7916	20	23	22	51,06	15,64
Arc111_c=8356	19	21	21	52,79	15,72
Arc111_c=8847	18	20	20	49,59	15,13
Arc111_c=9400	17	19	19	50,75	15,34
Arc83_c=10816	8	8	8	34,94	10,69
Arc83_c=3786	21	23	23	41,95	12,15
Arc83_c=3985	20	23	23	43,15	12,20
Arc83_c=4206	19	21	21	42,15	11,97
Arc83_c=4454	18	21	20	41,47	11,90
Arc83_c=4732	17	19	19	41,56	11,87
Arc83_c=5048	16	18	18	40,19	11,74
Arc83_c=5408	15	16	16	39,83	11,50

Arc83_c=5824	14	15	15	38,67	11,38
Arc83_c=5853	14	15	15	37,70	11,33
Arc83_c=6309	13	14	14	37,30	11,29
Arc83_c=6842	12	13	13	36,78	11,29
Arc83_c=6883	12	13	13	37,01	11,15
Arc83_c=7571	11	12	12	36,21	11,10
Arc83_c=8412	10	10	11	35,88	11,03
Arc83_c=8898	9	10	10	35,82	11,21
Tonge70_c=160	23	26	25	36,31	11,24
Tonge70_c=168	22	24	24	35,40	10,92
Tonge70_c=176	21	23	22	35,43	11,06
Tonge70_c=185	20	22	21	35,49	10,78
Tonge70_c=195	19	20	20	35,32	11,01
Tonge70_c=207	18	20	19	34,14	10,98
Tonge70_c=220	17	18	18	34,08	10,84
Tonge70_c=234	16	17	17	32,93	10,82
Tonge70_c=251	15	16	16	32,60	10,73
Tonge70_c=270	14	15	14	32,09	10,55
Tonge70_c=293	13	13	13	32,11	10,31
Tonge70_c=320	12	12	12	31,51	10,21
Tonge70_c=364	10	11	11	30,80	10,09
Tonge70_c=410	9	9	9	30,51	9,91
Tonge70_c=468	8	8	8	30,01	9,85
Tonge70_c=527	7	7	7	29,59	9,72
barthol2_c=101	44	54	51	82,97	27,23
barthol2_c=104	43	51	49	83,97	24,52
barthol2_c=106	42	51	49	83,96	24,26
barthol2_c=109	41	49	47	81,52	24,08
barthol2_c=112	40	47	46	80,70	24,81
barthol2_c=115	39	46	44	80,27	24,10
barthol2_c=118	38	45	43	82,60	23,88
barthol2_c=121	37	43	41	78,80	23,62
barthol2_c=125	36	42	41	79,31	23,47
barthol2_c=129	35	41	39	80,80	23,21
barthol2_c=133	34	39	38	76,56	23,12
barthol2_c=137	33	37	36	78,43	23,10
barthol2_c=142	31	36	35	77,08	22,92
barthol2_c=146	31	35	34	76,82	22,59
barthol2_c=152	29	33	32	74,70	22,61

barthol2_c=157	28	32	31	73,45	22,03
barthol2_c=163	27	31	31	73,08	22,29
barthol2_c=170	26	30	29	72,27	21,88
barthol2_c=84	53	60	57	88,67	25,24
barthol2_c=85	52	60	56	89,12	25,26
barthol2_c=87	51	61	57	89,36	25,40
barthol2_c=89	50	61	58	90,80	25,60
barthol2_c=91	49	61	57	90,36	25,43
barthol2_c=93	48	59	55	85,19	25,09
barthol2_c=95	47	57	54	84,55	25,80
barthol2_c=97	46	56	53	86,37	25,14
barthol2_c=99	45	55	52	91,95	24,97
barthold_c=403	12	12	12	65,77	20,00
barthold_c=434	11	11	11	65,08	19,61
barthold_c=470	11	11	11	64,79	19,72
barthold_c=513	10	10	10	64,50	19,68
barthold_c=564	9	9	9	64,00	19,83
barthold_c=626	8	8	8	63,84	19,72
barthold_c=705	7	7	7	63,88	19,78
barthold_c=805	6	6	6	63,98	19,69
bowman8_c=20	5	5	5	6,23	3,03
buxey_c=27	14	14	14	16,34	5,95
buxey_c=30	12	13	13	15,71	5,79
buxey_c=33	11	12	12	15,65	5,73
buxey_c=36	10	10	10	15,34	5,58
buxey_c=41	9	9	9	14,95	5,51
buxey_c=47	8	8	8	14,37	5,36
buxey_c=54	7	7	7	13,78	5,20
gunther_c=41	14	14	14	19,53	6,50
gunther_c=44	12	14	14	20,15	6,53
gunther_c=49	11	12	12	18,62	6,22
gunther_c=54	10	10	10	17,74	6,03
gunther_c=61	9	9	9	17,02	5,94
gunther_c=69	8	8	8	16,54	5,99
gunther_c=81	7	7	7	15,96	5,74
hahn_c=2004	8	9	9	23,65	7,81
hahn_c=2338	7	8	8	23,32	7,78
hahn_c=2806	6	6	6	23,13	7,55
hahn_c=3507	5	5	5	22,52	7,57

hahn_c=4676	4	4	4	21,98	7,44
heskia_c=138	8	9	9	14,76	5,36
heskia_c=205	6	6	6	13,35	5,07
heskia_c=216	5	6	6	13,18	5,04
heskia_c=256	5	5	5	12,89	4,99
heskia_c=324	4	4	4	12,59	4,88
heskia_c=342	4	4	4	12,58	4,88
jackson_c=10	6	6	6	7,48	3,50
jackson_c=13	5	5	5	7,32	3,49
jackson_c=14	4	4	4	7,14	3,33
jackson_c=21	3	3	3	6,70	3,32
jackson_c=7	9	9	9	8,32	3,64
jackson_c=9	7	7	7	7,88	3,53
jaeschke_c=10	5	5	5	6,50	3,15
jaeschke_c=18	3	3	3	6,01	3,15
jaeschke_c=6	8	9	9	7,22	2,99
jaeschke_c=7	7	8	8	7,25	3,33
jaeschke_c=8	6	8	8	7,24	3,29
kilbrid_c=110	6	6	6	19,25	6,54
kilbrid_c=111	6	6	6	19,21	6,67
kilbrid_c=138	5	5	5	19,07	6,43
kilbrid_c=184	4	4	4	18,69	6,39
kilbrid_c=56	11	11	11	21,15	7,01
kilbrid_c=57	11	11	11	21,18	7,05
kilbrid_c=62	10	11	10	20,77	7,05
kilbrid_c=69	9	10	9	20,69	6,83
kilbrid_c=79	8	8	8	20,06	6,66
kilbrid_c=92	7	7	7	19,57	6,58
lutz1_c=1414	11	11	11	16,61	5,79
lutz1_c=1572	10	11	11	16,58	5,84
lutz1_c=1768	9	10	10	15,93	5,83
lutz1_c=2020	8	9	9	15,55	5,65
lutz1_c=2357	7	7	7	15,08	5,47
lutz1_c=2828	6	6	6	14,81	5,33
lutz2_c=11	53	58	56	55,02	16,72
lutz2_c=12	48	54	53	53,68	16,54
lutz2_c=13	45	50	48	52,15	16,00
lutz2_c=14	41	47	45	51,26	15,50
lutz2_c=15	39	42	41	49,40	15,11

lutz2_c=16	36	38	36	49,11	14,46
lutz2_c=17	34	35	34	47,51	14,35
lutz2_c=18	32	32	32	47,43	14,45
lutz2_c=19	31	31	31	45,91	14,01
lutz2_c=20	30	29	29	45,32	13,81
lutz2_c=21	28	28	28	44,51	13,86
lutz3_c=103	18	18	17	40,93	12,34
lutz3_c=110	16	17	16	40,40	12,40
lutz3_c=118	15	15	15	40,44	12,17
lutz3_c=127	14	14	14	39,68	12,12
lutz3_c=137	13	13	13	39,48	11,92
lutz3_c=150	12	12	12	38,77	11,87
lutz3_c=75	24	25	25	43,72	13,41
lutz3_c=79	23	23	23	43,07	13,28
lutz3_c=83	22	22	22	43,25	13,08
lutz3_c=87	21	22	21	42,78	12,95
lutz3_c=92	20	20	20	42,28	12,64
lutz3_c=97	19	19	19	42,02	12,59
mansoor_c=48	5	5	5	7,22	3,50
mansoor_c=62	4	4	4	7,09	3,42
mansoor_c=94	3	3	3	6,75	3,35
mertens_c=10	4	4	4	5,66	2,91
mertens_c=15	3	3	3	5,32	2,97
mertens_c=18	2	2	2	5,00	2,81
mertens_c=6	6	6	6	6,05	3,04
mertens_c=7	6	6	6	6,07	3,08
mertens_c=8	5	6	6	6,09	3,06
mitchell_c=14	9	9	9	12,11	4,77
mitchell_c=15	9	9	9	11,95	4,70
mitchell_c=21	7	6	6	11,02	4,46
mitchell_c=26	5	5	5	10,68	4,33
mitchell_c=35	4	4	4	10,31	4,26
mitchell_c=39	4	4	4	10,36	4,14
mukherje_c=176	25	27	27	45,59	13,99
mukherje_c=183	24	27	27	45,39	13,95
mukherje_c=192	23	27	26	44,99	13,79
mukherje_c=201	22	25	25	44,44	14,00
mukherje_c=211	21	24	24	44,13	13,86
mukherje_c=222	20	23	23	43,81	13,70

mukherje_c=234	19	22	22	42,98	13,51
mukherje_c=248	18	21	21	42,91	13,21
mukherje_c=263	17	19	19	42,18	13,09
mukherje_c=281	16	18	18	41,63	12,89
mukherje_c=301	15	17	16	40,79	12,84
mukherje_c=324	14	16	15	40,96	12,62
mukherje_c=351	13	14	14	40,02	12,56
roszieg_c=14	11	11	11	14,10	5,13
roszieg_c=16	10	10	10	13,55	5,12
roszieg_c=18	9	9	8	13,16	4,90
roszieg_c=21	8	7	7	12,86	4,86
roszieg_c=25	7	6	6	12,13	4,82
roszieg_c=32	5	5	5	11,83	4,67
sawyer30_c=25	15	15	15	17,12	6,04
sawyer30_c=27	14	14	14	17,01	5,93
sawyer30_c=30	13	13	13	16,77	5,96
sawyer30_c=33	12	12	12	16,57	5,80
sawyer30_c=36	11	11	10	15,36	5,74
sawyer30_c=41	9	9	9	15,84	5,52
sawyer30_c=47	8	8	8	15,09	5,41
sawyer30_c=54	7	7	7	14,30	5,28
sawyer30_c=75	5	5	5	13,79	5,08
scholl_c=1394	51	55	54	164,21	55,84
scholl_c=1422	50	57	56	165,06	57,24
scholl_c=1452	49	58	56	169,38	55,69
scholl_c=1483	48	56	56	167,84	55,48
scholl_c=1515	47	55	54	167,07	56,36
scholl_c=1548	46	53	53	166,10	54,73
scholl_c=1584	45	52	52	164,38	55,22
scholl_c=1620	44	51	51	163,71	54,07
scholl_c=1659	43	50	48	163,42	54,06
scholl_c=1699	42	48	48	162,70	53,73
scholl_c=1742	41	47	46	160,42	53,98
scholl_c=1787	40	46	45	160,06	54,50
scholl_c=1834	39	45	44	165,82	55,29
scholl_c=1883	38	43	43	160,14	52,89
scholl_c=1935	37	42	41	161,77	53,51
scholl_c=1991	36	41	41	157,10	54,48
scholl_c=2049	35	39	39	162,47	54,46

scholl_c=2111	34	38	38	155,68	52,83
scholl_c=2177	33	37	37	155,14	52,85
scholl_c=2247	32	36	36	155,93	52,61
scholl_c=2322	31	35	35	156,70	52,32
scholl_c=2402	30	33	33	154,32	52,10
scholl_c=2488	29	32	32	153,49	52,74
scholl_c=2580	28	31	31	153,48	52,26
scholl_c=2680	27	30	30	153,25	51,64
scholl_c=2787	26	29	29	151,58	51,57
warnecke_c=104	16	18	18	28,59	9,17
warnecke_c=111	15	16	16	27,66	8,99
warnecke_c=54	31	33	33	34,09	10,72
warnecke_c=56	29	33	32	34,15	10,50
warnecke_c=58	29	34	32	34,09	10,64
warnecke_c=60	28	34	32	34,34	10,62
warnecke_c=62	27	33	31	33,83	10,31
warnecke_c=65	26	31	30	33,17	10,45
warnecke_c=68	25	30	29	33,26	10,17
warnecke_c=71	24	29	28	32,96	10,08
warnecke_c=74	23	26	26	31,63	9,99
warnecke_c=78	22	25	24	30,83	9,82
warnecke_c=82	21	24	22	30,52	9,60
warnecke_c=86	20	22	22	30,44	9,60
warnecke_c=92	18	20	20	29,77	9,39
warnecke_c=97	17	19	19	29,13	9,32
wee-mag_c=28	63	64	64	49,29	15,04
wee-mag_c=29	63	64	63	48,64	14,93
wee-mag_c=30	62	63	63	48,77	15,08
wee-mag_c=31	62	64	64	48,77	14,90
wee-mag_c=32	61	63	63	48,65	14,91
wee-mag_c=33	61	63	63	49,19	14,95
wee-mag_c=34	61	63	62	48,09	14,82
wee-mag_c=35	60	62	62	47,76	14,76
wee-mag_c=36	60	61	61	47,22	14,81
wee-mag_c=37	60	61	61	47,14	14,69
wee-mag_c=38	60	61	61	47,12	14,76
wee-mag_c=39	60	60	60	47,32	14,70
wee-mag_c=40	60	60	60	47,23	14,65
wee-mag_c=41	59	60	60	46,84	14,79

wee-mag_c=42	55	60	60	46,73	14,67
wee-mag_c=43	50	60	60	46,89	14,77
wee-mag_c=45	38	59	59	46,61	14,64
wee-mag_c=46	37	55	55	47,11	14,34
wee-mag_c=47	37	51	50	46,79	14,05
wee-mag_c=49	35	44	40	43,53	13,40
wee-mag_c=50	34	41	38	42,99	12,98
wee-mag_c=52	33	36	34	41,63	12,53
wee-mag_c=54	31	33	32	40,06	12,52
wee-mag_c=56	30	32	32	39,11	12,30