AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Ömür TOSUN

# YAPAY ARI KOLONİSİ ALGORİTMASI ve PERMÜTASYON AKIŞ TİPİ ÇİZELGELEME PROBLEMİNE UYGULANMASI

İşletme Ana Bilim Dalı
Doktora Tezi

Antalya, 2012

AKDENİZ ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

Ömür TOSUN

# YAPAY ARI KOLONİSİ ALGORİTMASI ve PERMÜTASYON AKIŞ TİPİ ÇİZELGELEME PROBLEMİNE UYGULANMASI

Danışman
Yrd.Doç.Dr.Gökhan AKYÜZ

İşletme Ana Bilim Dalı
Doktora Tezi

Antalya, 2012

Akdeniz Üniversitesi
Sosyal Bilimler Enstitüsü Müdürlüğüne,

Ömür TOSUN'un bu çalışması, jürimiz tarafından İşletme Ana Bilim Dalı Doktora Programı tezi olarak kabul edilmiştir.

Başkan : Doç.Dr. Can Deniz KÖKSAL

Üye (Danışmanı) : Yrd.Doç.Dr. Gökhan AKYÜZ

Üye : Doç.Dr. Hüseyin GÖKSU

Üye : Yrd.Doç.Dr. Nuri ÖMÜRBEK

Üye : Yrd.Doç.Dr. Sezgin IRMAK

Tez Konusu: Yapay Arı Kolonisi algoritmasının permütasyon oluşturma tipi sınıflandırma problemine uyarlanması

Onay : Yukarıdaki imzaların, adı geçen öğretim üyelerine ait olduğunu onaylarım.

Tez Savunma Tarihi : 12/09/2012

Mezuniyet Tarihi : 14/09/2012

Doç.Dr.Zekeriya KARADAVUT
Müdür

.....

# İ Ç İ N D E K İ L E R

## İKİNCİ BÖLÜM
## YAPAY ARI KOLONİSİ ALGORİTMASI



## ÜÇÜNCÜ BÖLÜM
## ABC ALGORİTMASININ EN GEÇ TAMAMLANMA ZAMANI KRİTERİNE SAHİP PERMÜTASYON AKIŞ TİPİ ÇİZELGELEME PROBLEMİNE UYARLANMASI

# TABLOLAR LİSTESİ

# ŞEKİLLER LİSTESİ

# KISALTMALAR LİSTESİ

| | |
|---|---|
| ABC | Yapay Arı Kolonisi |
| ACO | Karınca Kolonisi Algoritması |
| ACS | Karınca Kolonisi Sistemi |
| ARE | Ortalama Sapma Değeri |
| BRE | Bilinen En İyi Çözüm Değerinden Sapma Yüzdesi |
| $C_{mak}$ | En Geç Tamamlanma Zamanı |
| CDS | Campbell, Dudek ve Smith |
| DE | Evrimsel Gelişim Algoritması |
| EDA | Dağılım Tahmini Algoritması |
| FM | Ateş Böceği Algoritması |
| GA | Genetik Algoritma |
| GAP | Genelleştirilmiş Atama Problemi |
| GRASP | Aç Gözlü Rastgele Adaptif Arama |
| HDABC | Kesikli ABC Algoritması |
| IIGA | Geliştirilmiş İteratif Aç Gözlü Algoritma |
| ILS | İteratif Yerel Arama |
| LCMST | Yaprak Kısıtlı Minimum Dağılan Ağaç Problemi |
| NEH | Nawaz, Enscore ve Ham |
| NP | ABC Algoritmasındaki Arı Sayısı |
| NP-Zor | Çok Terimli (polinomsal) Zamanda Çözülebilen Problem Sınıfı |
| MICOT | En Düşük Tamamlanma Zamanı Sezgiseli |
| MINIMAX | Johnson Kuralına Dayanan En Düşük Tamamlanma Zamanı Sezgiseli |
| MINIT | En Düşük Boş Zaman Sezgiseli |
| MMAS | Mak-min Karınca Sistemi |
| MRP | Malzeme İhtiyaç Planlama |
| MSXF | Çok Adımlı Çaprazlama |
| PFSP | Permütasyon Akış Tipi Çizelgeleme Problemi |
| PMX | Kısmı Haritalandırılmış Çaprazlama |
| PSO | Parçacık Sürüsü Optimizasyonu |

| | |
|---|---|
| RA | Hızlı Erişim |
| RACS | Yakın Komşu Aramalı Hızlı Erişim Algoritması |
| RAES | Geliştirilmiş Aramalı Hızlı Erişim Algoritması |
| SA | Benzetim Tavlaması |
| SN | Yiyecek Kaynağı |
| SPIRIT | Widmer ve Hertz Sezgiseli |
| SPV | En Küçük Pozisyon Değeri Sezgiseli |
| SS | Dağınık Arama Algoritması |
| TS | Tabu Araması |
| TSP | Gezgin Satıcı Problemi |
| UFLP | Kapasitesiz Fabrika Yerleşim Problemi |
| Q-MST | Kareli Minimum Dağılan Ağaç Problemi |
| VNS | Değişken Komşuluk Araması Sezgiseli |
| VRP | Araç Rotalama Problemi |
| YSA | Yapay Sinir Ağları |

# ÖZET

Akış tipi çizelgeleme problemi endüstri mühendisliği alanında son 50 yılın muhtemelen en çok bilinen problemidir. Bir ve iki makine durumu için kesin çözümler olmakla beraber, makine sayısının üçten fazla olduğu durumlarda problem NP-Zor sınıfına girmekte ve optimum çözümü bulmak zorlaşmaktadır. Bu zorlayıcı ortam, pek çok araştırmacının ilgisini çekmektedir. Yapay arı kolonisi (ABC) algoritması, Karaboğa (2005) tarafından arıların doğadaki davranışları temel alınarak geliştirilen popülasyon tabanlı bir optimizasyon tekniğidir. Öncelikle sayısal optimizasyon problemlerinde kullanılan algoritma, son zamanlarda çeşitli mühendislik problemlerinde de başarıyla uygulanmıştır. Bu tez çalışmasında, ABC algoritması permütasyon akış tipi çizelgeleme problemine uygulanmıştır. Geliştirilen algoritmanın performansı literatürde sıklıkla kullanılan Carlier, Reeves ve Taillard test problemlerine uygulanmış ve sonuçları farklı metasezgisel yöntemlerle karşılaştırılmıştır. Geliştirilen algoritma Carlier ve Reeves problem gruplarında literatürde yer alan pek çok çalışmadan daha başarılı sonuçlar üretmiştir. Reeves23 test problemi için, yeni bir en iyi değer elde edilmiştir. Taillard test problemlerinde ise, karşılaştırma için kullanılan diğer metasezgisel algoritmalarla rekabetçi sonuçlar elde edilmiştir. Bazı problemlerde diğer algoritmalardan daha başarılı değerler bulunurken, kimi problemlerde de küçük farklarla rakip algoritmaların gerisinde kalmıştır.

## SUMMARY

Flow shop scheduling problem (FSSP) is probably one of the most well known industrial engineering problem of the past 50 years. Although there are exact solutions for one or two machines, the problem becomes NP-hard for three or more machines. This challenging environment attracts many researchers to offer more efficient solution algorithms. Artificial Bee Colony (ABC) algorithm is a relatively new swarm intelligence-based algorithm proposed by Karaboğa (2005), inspired by the intelligent foraging behaviors of honeybee swarms. Primarily used for solving numerical optimization problems, it is now used widely in different engineering problem types. In this study, ABC algorithm is applied to permutation flow shop scheduling problem. Performance of the algorithm is tested by using Carlier, Reeves and Taillard test instances and compared against different metaheuristic-based methods. For the Carlier and Reeves instances, the proposed ABC algorithm produced better results than most of the known metaheuristic algorithms in the literature. A new best solution is gained for the Reeves23 test problem. Some competitive results are found for the Taillard test problems against different metaheuristic methods used for comparing. In some instances, ABC algorithm performed better than the other methods, whereas in some instances its solutions are slightly worse than the others.

# ÖNSÖZ

Bugünlere gelmemde ve bu çalışmayı hazırlamamda emeği geçen, sürekli bizi teşvik eden Hocam Prof. Dr. Orhan KURUÜZÜM'e, Orhan Hocam emekli olduktan sonra beni danışmanlık kanatları altına alan ve Araştırma Görevliliğine başladığım ilk günden beni desteklerini esirgemeyen Yrd.Doç.Dr. Gökhan AKYÜZ'e, Tez İzleme Jürimde yer alıp sürekli beni yönlendiren değerli Hocalarıma, odamızda sürekli bir neşe ortamı yaratan Onur DİRLİK ve Duygu AYDIN'a ve diğer tüm Araştırma Görevlisi arkadaşlarıma, son olarak da her zaman en büyük destekçilerim olan sevgili eşim Nedret ve küçük kızım İrem TOSUN'a sonsuz teşekkürleri bir borç bilirim.

Bu tez çalışması, Akdeniz Üniversitesi Bilimsel Araştırmalar Biriminin 2010.03.0107.002 numaralı projesi ile desteklenmiştir.

**Ömür TOSUN**

**Antalya, 2012**

# G İ R İ Ş

Çizelgeleme sorunu, gerek imalat sektöründe gerekse hizmet sektöründe karar vericilerin en sık karşılaştıkları sorunlardan birisidir. Belirli bir dönem içerisinde farklı teknoloji ve kapasite kısıtları altında hem hizmet verenin hem de hizmet alanın ihtiyaçlarının en uygun şekilde karşılanması gerekmektedir. Üretim sistemlerinin doğasında yer alan dinamik yapıdan dolayı, ele alınması gereken sorunlar çoğu zaman oldukça karmaşık bir yapıda ortaya çıkmaktadır. İyi yapılmış bir çizelgeleme çalışması, üretim ve hizmet işletmelerinde verimliliği arttırmakta beraber kaynak kullanımı ve maliyetleri de doğrudan etkileyebilmektedir.

Çizelgeleme sorunun çözümünde Johnson (1954) tarafından geliştirilen ilk çözüm algoritmasından bu yana pek çok çözüm tekniği ortaya çıkmıştır. Yaklaşık 50 yıldır gündemde olan bu probleme olan ilgi günümüzde de artarak devam etmektedir. Johnson algoritması iki ve üç makine durumu için kesin çözümü verebilmektedir. Makine sayısının üçü geçtiği durumlarda ise en iyi çözümü elde etmek için gerekli olan süre üstel olarak artmakta ve karmaşık problemlerde en iyi çözümü bulmak neredeyse imkansız hale gelmektedir. Problemdeki bu zorluk pek çok araştırmacının ilgisini çekmiş ve en kısa sürede optimuma yakın çözümler verecek çeşitli sezgisel ve metasezgisel algoritmalar geliştirilmiştir.

Metasezgisel algoritmalar, çeşitli mühendislik alanlarında yer alan farklı problemlerin çözümünde günümüzde sıklıkla kullanılmaktadırlar. Bu yöntemlerin kullanılmasıyla büyük boyutlu problemlerde daha az hesaplama karmaşıklığı ile klasik yöntemlerle elde edilemeyecek kadar kısa sürede iyi çözümler ortaya çıkabilmektedir. Özellikle endüstri mühendisliği alanında araç rotalama problemi, gezgin satıcı problemi, atölye tipi ve akış tipi iş çizelgeleme problemi, zaman çizelgeleme, personel atama gibi çok çeşitli sorunlarda bu yöntemler başarıyla kullanılabilmektedir.

Doğadaki canlıların davranış biçimlerinden yola çıkarak geliştirilen yapay zeka teknikleri, son yıllarda giderek daha çok kullanılmaktadır. Pek çok araştırmacı tarafından geliştirilen algoritmalar, farklı problem türlerine uygulanmakta ve literatürdeki çeşitli veri setleri kullanılarak performansları ölçülmektedir. Kullanılan bu veri setleri sayesinde algoritmaların

performansı farklı metasezgisel tekniklerle veya aynı algoritmanın farklı yazarlar tarafından geliştirilen sürümleri ile kıyaslanabilmektedir. Bu çalışmada, doğadaki arı kolonilerinin davranışlarını temel alarak geliştirilen yapay arı kolonisi algoritması permütasyon akış tipi çizelgeleme problemine (PFSP) uyarlanmıştır.

Çalışmanın birinci bölümü çizelgeleme fonksiyonuna ilişkin ayrıntılı bilgiler içermektedir. Çizelgelemenin işletmeler için öneminin yanı sıra, akış tipi çizelgeleme problemi ve bu probleme yönelik çeşitli çözüm teknikleriyle ilgili ayrıntılar verilmiştir. İkinci bölümde, problemin çözümü için kullanılacak olan yapay arı kolonisi algoritması ve bu algoritmayla ilgili yapılan literatür taraması yer almaktadır. Son bölümde ise algoritmanın PFSP'ye uyarlanması ve algoritma performansının Carlier, Reeves ve Taillard test problemleri üzerinde denenmesi anlatılmıştır.

# BİRİNCİ BÖLÜM
# ÇİZELGELEME FONKSİYONU

Çizelgeleme problemi ilk ortaya çıkışından bu yana gerek akademik çalışmalarda gerekse de iş dünyasında önemini hiç kaybetmemiştir. İmalat ve hizmet sektöründe yer alan hemen hemen tüm işletmelerde karar vericilerin karşısına çıkan bu sorunun çözümü için, son 50 yılda sayısız çözüm algoritması geliştirilmiştir. Üzerinde sıklıkla çalışılmasına rağmen, günümüzde yeni ve rekabetçi çözüm algoritmaları geliştirilmeye devam edilmektedir.

Bu bölümde, öncelikle çizelgeleme fonksiyonunun işletme içindeki önemi anlatılacaktır. Çizelgeleme problemlerinin ve çözüm tekniklerinin sınıflandırılması anlatıldıktan sonra, en geç tamamlanma zamanı kriteri için PFSP çözümüne yönelik literatür taraması verilecektir.

## 1.1. Çizelgeleme Fonksiyonunun Önemi

Bilim ve teknolojide gerçekleşen sürekli gelişim sonucu tüketicilerin refah düzeyleri ile birlikte beklentileri de artmakta, bu artış firmalar açısından ekonomik bir rekabet ortamı yaratmaktadır. Bu rekabetçi ortamda hayatta kalabilmek için firmalar; ticari anlamda kazanılmış olan pazar ve müşteri portföylerini korumaya çalışmakta, yeni pazarlar aramakta veya yeni müşteriler kazanabilmek için çalışmalar yapmaktadırlar. Rekabet üstünlüğü sağlamak ve sürdürmek için mükemmel bir ürün tasarlamak veya pazarlamaya yoğunlaşmak yetmemektedir. Müşteri hizmetinde kusursuzluğa ulaşmak ve teslimat tarihi için verilen sözleri yerine getirmek en az bunlar kadar önemlidir. Bunun için yapılması gerekenlerin biri; kapasiteyi talebe yanıt verebilecek düzeyde ve beklenmedik durumlara çabuk tepki gösterecek şekilde tutmaktır. Böyle bir üretim sisteminde yapılması gereken ilk çalışmalardan biriside çizelgeleme faaliyetidir.

Çizelgeleme, imalat ve hizmet endüstrilerinde çok önemli role sahip bir karar verme sürecidir. Bir firmada çizelgeleme fonksiyonu, matematiksel teknikler veya sezgisel yöntemler kullanarak sınırlı kaynakların verilen zaman çerçevesi içinde görevlere tahsis edilmesi işlemini gerçekleştirir. Kaynakların uygun olarak atanması ile firmanın bir veya daha fazla amacı optimize ederek hedeflerine en iyi şekilde ulaşması sağlanır (Pinedo, 2008, s.1).

Üretim sisteminde, atölye içindeki kabarık sayıdaki yarı mamul yığınları ve/veya bir kısım tezgahların çalışırken diğerlerinin boş durması gibi durumların gözlemlenmesi çizelgeleme problemlerinin varlığını ortaya koymaktadır. Ayrıca, üretim ortamında ortaya çıkabilecek yüksek seviyede fazla mesai, gecikmiş işlerin varlığı, düşük tezgah/işgücü kullanım oranları gibi istatistikler de çizelgeleme problemlerinin işaretleridir (Güldalı, 1990 alıntılayan: Seçme, 2006, s.4).

Çizelgeleme, belirli bir takım görevleri yerine getirmek üzere kaynakların zaman içinde tahsisi olarak tanımlanabilir. Bu tanımda görevler, içinde bulunulan ortama bağlı olarak değişik şekillerde adlandırılabilir (Pinedo, 2008, s.1). Genel olarak, belirli ürün veya hizmetleri meydana getirmek için gerekli olan faaliyetlerdir. Çizelgelemede üretim, kaynak ve zaman olmak üzere üç temel unsur yer almaktadır. Bu unsurları göz önüne alarak çizelgelemeyi, belirli bir takım işleri yapmak için hangi kaynakların, ne zaman ve nasıl kullanılacaklarının tespit edilmesi şeklinde ifade edebiliriz (Güldalı, 1990 alıntılayan: Seçme, 2006, s.4). Çizelgeleme problemine ait günümüzde karşımıza çıkan çeşitli örnekler aşağıda verilmiştir.

- Proje çizelgeleme
- Bakım çizelgeleme
- Havayollarında uçuş çizelgeleme
- Araç rotalama
- Gezgin satıcı problemi
- Montaj hattı dengeleme
- İş çizelgeleme
- Servis rotalama ve çizelgeleme
- Havayolu ekip çizelgeleme
- Ders ve sınav çizelgeleme
- Üretim çizelgeleme
- İşgücü çizelgeleme

Çizelgeleme, stratejik planlamayla başlayıp artan ayrıntılara sahip planlama faaliyetlerine doğru ilerleyen planlama sürecinin son adımıdır. Çizelgeleme faaliyetine başlamadan önce çizelgelenecek eylemlerle ilgili yeterli verinin olması gerekmektedir. Çizelgelenecek faaliyetle

değişmekle birlikte gereken bazı bilgiler aşağıda verilmiştir (Vonderembse ve White, 1991, s. 522-523):

İşler

- Teslim tarihi
- Rotalar, hazırlık ve işlem süreleri
- Malzeme ihtiyaçları
- Teslim tarihiyle ilgili esneklikler
- Zamanında teslimin önem derecesi

Faaliyetler

- Beklenen süre
- Öncelik ilişkileri
- İstenen tamamlanma süresi

Çalışanlar

- Hazırda bulunabilirlik
- İş kapasiteleri
- Çeşitli işlerdeki etkinlikleri
- Ücret oranları

Ekipman

- Makine/iş merkezi kapasitesi
- Makine/iş merkezi yetenekleri
- Operasyon maliyetleri
- Hazırda bulunabilirlik

Tesisler

- Kapasite
- Kullanım yerleri
- Kullanım maliyetleri
- Hazırda bulunabilirlik

## 1.2. Üretim Çizelgeleme

Üretim çizelgeme, üretim veya hizmet sistemlerinde yapılacak işlem/işlem gruplarının hangi iş istasyonlarında, ne zaman ve nasıl gerçekleştirileceğinin belirlenmesidir. Verimlilik ve etkinlik sağlama açısından önemli bir işletme fonksiyonudur (Pinedo, 2008, s.1).

Üretim çizelgeleme fonksiyonu, üretim sisteminde siparişlerin yani müşteri taleplerinin karşılanması süreci açısından önemlidir. Üretim planlama verileri ve atölye verileri arasında çizelgeleme fonksiyonu köprü vazifesi görür. Üretim planlamadan gelen sipariş bilgileri ile (sipariş miktarı, başlama ve teslim zamanları vb.) mevcut atölye (üretim alanı) durumunu göz önüne alarak en iyi ve etkin çizelgenin belirlenmesi, üretim planlama ile atölye yönetimine geri beslemenin yapılması sağlanır.

Çizelgeleme ile ilgili bulunan çeşitli tanımlardan en yaygın kullanılanları aşağıda belirtilmiştir:

- Çizelgeleme, kıt kaynakların belirli bir zaman boyunca çeşitli işlere tahsis edilmesiyle ilgilidir. Çizelgeleme süreci, bir veya daha fazla amacı en iyileyecek şekilde kıt kaynakların atanması ve bu kaynakların etkinliği ile ilgilenmektedir. Bu süreç, bir veya daha fazla hedefin en iyilenmesini amaçlayan bir karar alma sürecidir (Pinedo, 2008, s.1).
- Çizelgeleme, sistem kaynaklarının çeşitli işlere, görevlere veya faaliyetlere zaman temelinde tahsis edilmesidir.
- Çizelgeleme kaynakların etkin kullanımı ile belirlenen hedeflere ulaşmaktır.
- Faaliyetlerin nerede ve ne zaman gerçekleştirileceğine karar vermek anlamına gelen çizelgeleme, daha çok girdilerle çıktılar arasındaki zamanlamayla ilgilenmektedir (Özkazanç, 1999 alıntılayan: Seçme, 2006, s.4).

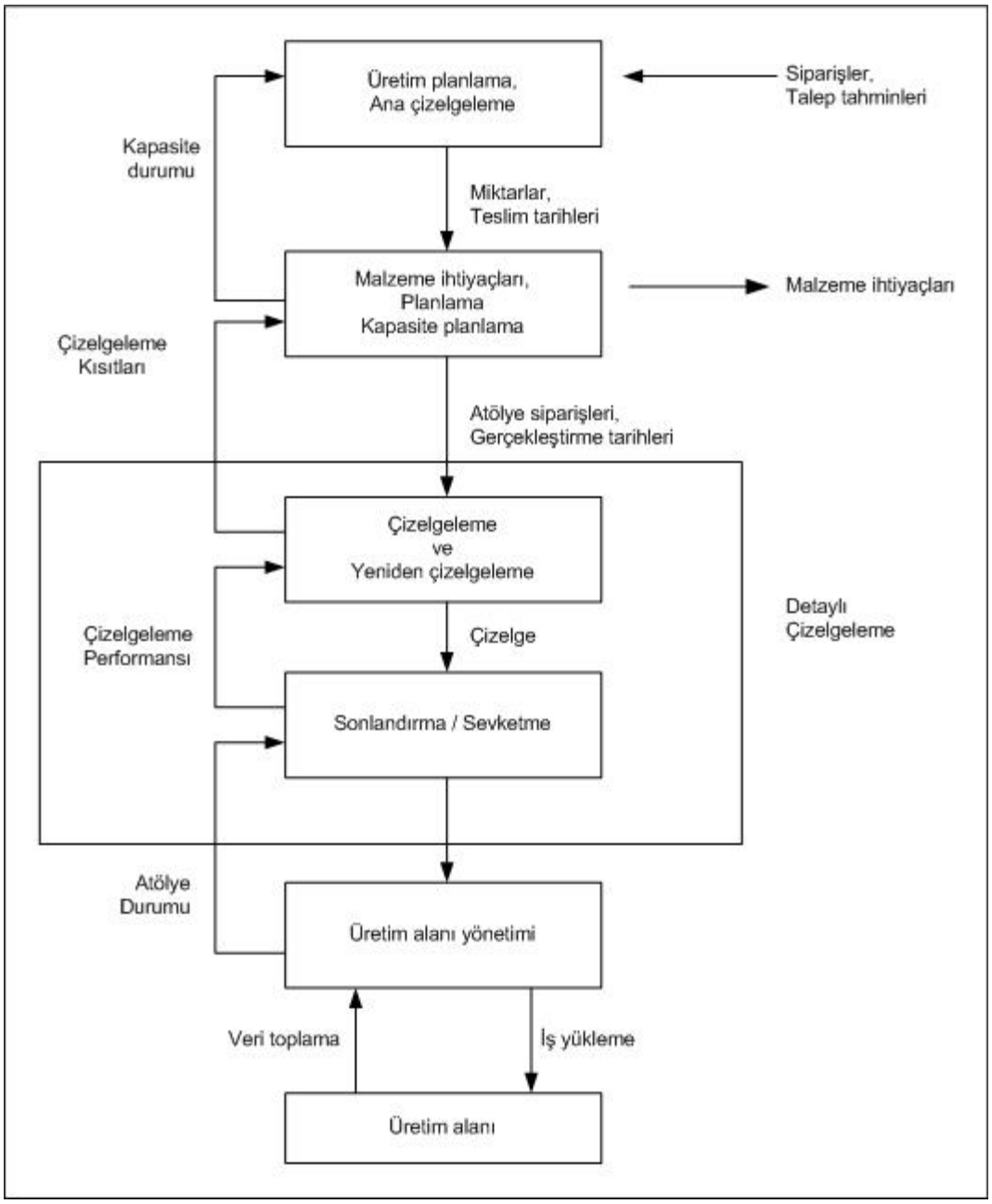
Çizelgeleme, üretim planlama sürecinin önemli bir parçasıdır (Şekil 1.1). Üretim planlama, firmanın genel ürün karmasının ve uzun dönemli kaynak tahsisinin; stok seviyeleri, talep tahminleri ve kaynak ihtiyaçlarına bağlı olarak optimize edilmesine çalışır. Bu üst planlama seviyesinde alınan kararlar çizelgeleme sürecini doğrudan etkilemektedir.

İmalatta, çizelgeleme fonksiyonu fabrikanın diğer karar alma fonksiyonları ile etkileşim içindedir. Birçok fabrika tarafından kullanılan Malzeme İhtiyaç Planlaması (MRP) sistemi bu fonksiyonlara bir örnektir. Çizelge oluşturulduktan sonra, gerekli tüm hammadde ve kaynakların belirlenen zamanda kullanılabilir olması şarttır. Tüm işlerin hazırlanma tarihleri üretim planlama, çizelgeleme sistemi ve MRP sistemince ortaklaşa belirlenir (Pinedo, 2008, s.6).

Üretim sistemi içerisinde çizelgeleme birçok faktörden etkilenmektedir. Bunlardan bazıları; iş öncelikleri, teslim tarihleri, üretim seviyeleri, parti büyüklüğü sınırlamaları, öncelik kurallarının yanı sıra ihtiyaç üretimi, işlem karmaşıklığı, çizelgeleme kriteri, ihtiyaç tanımlama ve çizelgeleme ortamıdır. Çizelgelemeyi ilk önce etkileyen temel unsur, sürecin planlanması işlemidir (Geyik ve Cedimoğlu, 2001, s.53-54).

Çizelgeleme problemleri, yerine getirilmesi gereken bir grup iş ve bu işlerin gerçekleştirilmesinde kullanılacak uygun kaynakları içerirler. Bu iki temel girdinin nitelikleri iyi belirlenmeli ve alacakları değerler olabildiğince kesin ve doğru bir biçimde hesaplanmalıdır. Ancak bu şekilde doğru zaman planları ortaya çıkarılabilir. Kaynaklar belirlendiğinde çizelgeleme probleminin sınırları etkin bir şekilde çizilmiş olmaktadır. Aynı zamanda, bu görevler bütünü arasında her hangi bir teknolojik kısıt varsa belirtilmelidir (Özkazanç, 1999 alıntılayan: Seçme, 2006, s.5).

Bir organizasyondaki çizelgeleme işlevi, Şekil 1.1'de görüldüğü gibi, sadece atölyeden değil, aynı zamanda orta ve uzun dönemli planlamadan sorumlu üretim planlama işlevinden de etkilenir. Üretim planlama işlevi, kaynak ihtiyaçları, talep tahminleri ve stok seviyelerini göz önünde bulundurarak uzun dönemli kaynak tahsisinin yanı sıra firmanın ürün karışımını da en iyilemeyi amaçlar. Bu yüksek planlama seviyesindeki kararlar çizelgeleme işlevini doğrudan etkileyecektir




**Şekil 1.1 Bir Üretim Sisteminde Çizelgeleme Fonksiyonunun Yeri (Pinedo, 2008, s.5)**

Çizelgeleme problemlerinde iki farklı türden uygunluk kısıtlarıyla karşılaşılmaktadır. Bunlardan birincisi makine kapasitelerindeki sınırlamalar, ikincisi ise bazı işlerin işlem sırası üzerindeki teknolojik sınırlamalardır. Çizelgeleme probleminin çözümü, iki gruba ayrılabilen bu kısıtların her hangi bir uygun çözümü ile ifade edilir. Dolayısıyla çizelgeleme problemini çözme işlemi aşağıdaki sorulara cevap arama işlemini kapsamaktadır (Baker, 1997 alıntılayan: Kellegöz, 2006, s.2).

- Hangi kaynak hangi göreve atanacak?
- Her bir görev ne zaman yapılacak?

### 1.3. Çizelgeleme Problemlerinin Sınıflandırılması

Çizelgeleme problemleri üç parametreli veya dört parametreli olmak üzere iki farklı sistemde ifade edilebilir. Bunlar Conway vd. (1967) tarafından geliştirilen $\alpha$ / $\beta$ / $\gamma$ / $\delta$ gösterimi ve Graham vd. (1979) tarafından geliştirilen $\alpha / \beta / \gamma$ şeklinde üç parametreli gösterimdir (Zobolas vd., 2008, s.5; Ruiz ve Maroto, 2005, s.480). Conway tarafından geliştirilen dörtlü gösterimde $\alpha$ parametresi işlerin geliş sürecini (işler arasındaki sürenin olasılık dağılımını) göstermektedir. Sadece *n* ile ifade edildiği durumda ise sonlu sayıda makineye sahip statik bir durumu gösterir. İkinci parametre, $\beta$, makine sayısını göstermektedir. Üçüncü parametre atölyedeki işlerin akış düzenini ifade etmektedir. Bu parametrenin alacağı değer problemin türünü göstermektedir (akış tipi, atölye tipi, …). Son olarak $\delta$ parametresi ise amaç kriterini ifade eder (Conway vd., 1967, s. 7-8).

Üçlü gösterim de ise, $\alpha$ parametresi makine ortamını göstermektedir ve tek bir girdiye sahiptir. $\beta$, işleme özellikleri ve kısıtlarla ilgilidir. Problemin özelliğine bağlı olarak hiçbir girdisi olmayabileceği gibi bir veya birden fazla değer alabilir. Son olarak $\gamma$ ise genellikle tek girdiye sahip olup problemdeki en küçüklenecek amaç göstergesidir (Pinedo, 2008, s.14).

α parametresinin alabileceği bazı değerler (Pinedo, 2008, s.14-15):

- Tek makine (1),
- Benzer özellikli paralel makineler ($P_m$): Benzer özelliğe sahip *m* adet paralel makine vardır. *i* işine ait tek bir operasyon bulunur ve bu operasyon *m* adet makinenin herhangi birisinde yapılabilir.
- Farklı hızlara sahip paralel makineler ($Q_m$): Farklı hızlara sahip *m* adet paralel makine vardır ve *j* makinesinin hızı $v_j$ olarak tanımlanır. $P_i$ işlem zamanına sahip *i* işi *j* makinesinde $P_{ij} = P_i / v_j$ süresini harcar. Eğer tüm makinelerin hızları aynıysa bir önceki durum elde edilir.
- İlişkisiz paralel makineler ($R_m$): Her iş için farklı hızlara sahip *m* adet paralel makine vardır. *j* makinesi *i* işini $v_{ij}$ hızıyla yapabilmekte olup harcanan süre $p_i / v_{ij}$'dir.

- Akış tipi ($F_m$): Seri sıralanmış *m* adet makine söz konusudur. Her bir iş aynı rotayı izleyecek şekilde *m* adet makinenin her birinde işlem görür. Genellikle ilk gelen ilk çıkar kuyruk modeli temel alınır. Bu durumda kuyrukta bekleyen iş, başka bir işi atlayamaz. Bu tür problemler permütasyonlu akış tipi çizelgeleme problemi olarak bilinir.
- Esnek akış tipi ($FF_s$): Akış tipi ve paralel makine ortamının genelleştirilmiş halidir. Toplam *s* adet seri aşama bulunmakta olup her bir aşamada benzer özellikli *m* adet paralel makine vardır. Her bir iş aynı rotayı izleyecek şekilde *s* adet aşamanın her birinde bulunan *m* adet makinenin sadece birinde işlem görür.
- Atölye tipi ($J_m$): *m* adet makineye sahip atölye tipi çizelgelemede her işin kendine ait önceden belirlenmiş bir rotası vardır. Her işin her makineye bir kez uğradığı ve birden çok uğradığı biçiminde iki alt türü bulunmaktadır. İkinci tür problemler *rcrc* (tekrarlı) akış tipi olarak tanımlanabilir.
- Açık tip ($O_m$): Modelde *m* adet makine söz konusu olup her bir iş her bir makinede işlem görür. Bazı işlerin bazı makinelerdeki işlem süreleri sıfır olabileceği gibi farklı işler farklı rotalara da sahip olabilir.

β parametresinin alabileceği bazı değerler (Pinedo, 2008, s.15-17):

- Geliş zamanı ($r_i$): İş *i*'nin işlenmesine $r_i$ geliş zamanından önce başlanamaz. Eğer $\beta$ parametresinde $r_i$ yer almıyorsa *i* işi herhangi bir zamanda başlayabilir.
- Sıra bağımlı hazırlık zamanları ($S_{ik}$): Eğer çizelgede *i* işi *k* işinden önce geliyorsa $S_{ik}$, *k* işin başlayabilmesi için gereken hazırlık zamanını ifade eder. Eğer *i* ve *k* işleri arasındaki hazırlık zamanı makineye de bağımlı ise o zaman $S_{ijk}$ ile ifade edilir.
- Bölünebilme (*prmp*): İşin tamamlanana kadar makinede kalması zorunlu değildir. Her hangi bir zamanda her hangi bir işin işlenmesi durdurularak makineye farklı bir iş yerleştirilebilir. İşlemi yarıda kesilen iş ilgili makineye tekrar konduğunda sadece kalan süre kadar işlem görür.
- Öncelik kısıtları (*prec*): Bazı işlerin işlenmesine başlamadan önce diğer bazı işlerin tamamlaması kısıtıdır.
- Arızalanma (*brkdwn*): Makinelerin tamamı ya da bir kısmı arızalanmalar sebebiyle sürekli olarak işlem yapmaya uygun değildir.

- Permütasyon (*prmu*): Akış tipi makine ortamında karşılaşılan bu parametre, makineler arasındaki kuyruk disiplininin FIFO (ilk gelen ilk işlem görür) olduğunu ifade eder.
- Bloklama (*block*): Akış tipi makine ortamında karşılaşılan bu parametre, bir birini takip eden iki makine arasındaki kuyruğun sınırlı bir kapasiteye sahip ve dolu olduğu zaman önceki makinenin işlemini bitirdiği işi sonraki makineye gönderemeyeceğini ifade eder.
- Beklemesiz (*nwt*): Akış tipi makine ortamında karşılaşılan bu parametre, işlerin birbirini takip eden iki makine arasında bekleyemeyeceğini ifade eder. Bu kısıtlamanın olduğu modelde de kuyruk disiplini FIFO'dur.
- Yeniden dolaşım (*recrc*): Atölye tipi makine ortamında karşılaşılan bu durumda, her hangi bir işin her hangi bir makineyi birden fazla ziyaret edebileceğini ifade eder.

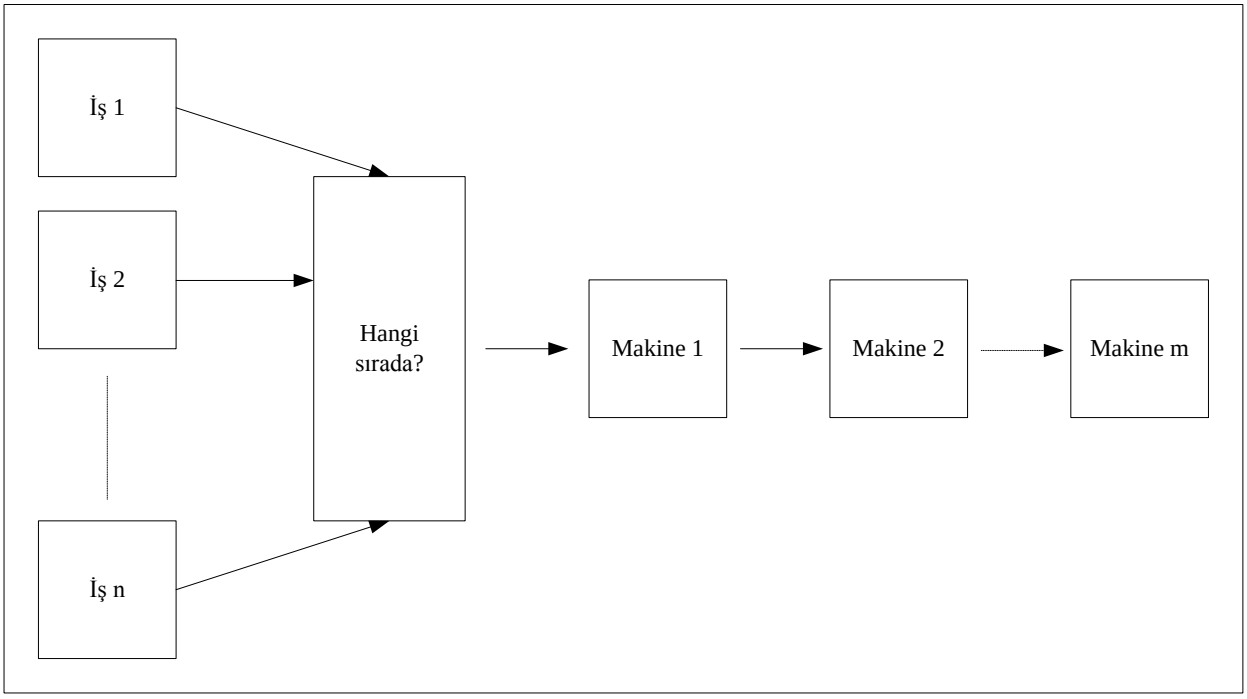
Çizelgeleme problemlerinde genellikle bir performans ölçütü en küçüklenmeye çalışılır. *i* işinin *j* makinesinde tamamlanma zamanı $C_{ij}$ ve sistemde geçirdiği zaman $C_j$ olarak tanımlanır. Performans ölçütü aynı zamanda teslim zamanının ($d_i$) bir fonksiyonu da olabilir. Herhangi bir *i* işinin gecikmesi $L_i = C_i - d_i$, *i* işinin geç bitmesi $T_i = mak(L_i, 0)$ ile gösterilir. Bunun sonucunda $\gamma$ parametresinin alabileceği bazı değerler (Pinedo, 2008, s.18-19; Morton ve Pentico, 1993, s.53-54):

- En geç tamamlanma zamanı ($C_{mak}$): $C_{mak} = mak\ (C_1, \ldots, C_i)$ şeklinde tanımlı olup, en son işin sistemi terk etme zamanını ifade eder. En küçüklenmesi genellikle yüksek makine verimliliği sağlar.
- Maksimum gecikme ($L_{mak}$): $L_{mak} = mak\ (L_1, \ldots, L_i)$ şeklinde tanımlı olup, teslim zamanından sapmaların en büyüğünü ifade eder.
- Akış zamanı ($\sum C_i$): Sistemde işlem gören parçaların, sistemde kaldığı toplam süredir.
- Ağırlıklı akış zamanı ($\sum w_i C_i$): Stok taşıma maliyeti gibi çizelgenin neden olduğu maliyetlerin bir göstergesidir.
- Toplam geç bitirme ($\sum T_i$)
- Ağırlıklı geç bitirme ($\sum w_i T_i$)
- Toplam geciken iş ($\sum U_i$)
- Ağırlıklı geciken iş ($\sum w_i U_i$)

Üretim sistemlerinde; atölye tipi, akış tipi, personel, sipariş ve üretim planlama gibi birçok uygulamada çizelgeleme kullanılmaktadır. Genel olarak süreç odaklı yaklaşımda, üretim çizelgeleme problemleri ***atölye tipi (job shop)*** ve ***akış tipi (flow shop)*** olmak üzere ikiye ayrılmaktadır.

### 1.4. Akış Tipi İş Çizelgeleme Problemi

Geleneksel akış tipi çizelgeleme problemini, *m* makinede *n* adet işin aynı teknolojik sırada işleneceği varsayımı altında, $p_{ij}$ *i* işinin *j* makinesinde işlem süresi olması şartıyla ( $i = 1,2,\ldots,n$; $j = 1,2,\ldots,m$) iyi tanımlanmış bir üretim maliyeti ölçütünü en küçükleyecek ve bu *n* işinin her bir *m* makinesinde işlenmesini sağlayacak çizelgelemeyi bulmak olarak tanımlayabiliriz (Gupta ve Stafford, 2006, s. 701). Akış tipi çizelgeleme problemlerinde makineler seri olarak sıralanmıştır ve her hangi bir makinede operasyonu tamamlanan iş bir sonraki makinenin kuyruğuna katılır. Bu tür çizelgeleme problemleri, işlerin makineden makineye malzeme taşıma sistemleriyle taşındığı üretim sistemlerinde ve montaj hatlarında karşımıza çıkar.




**Şekil 1.2 Akış Tipi Çizelgeleme (Kellegöz, 2006, s. 29)**

Bu tanımda verilen genel durum $(n!)^m$ adet olası çizelgeyi içerir. Akış tipi çizelgelemenin daha genel bir durumu olan permütasyon akış tipi çizelgeleme probleminde (Permutation Flow Shop Problem – PFSP) ise işler tüm makinelerde aynı sırada işlem görür (yani tüm makinelerde kuyruk disiplini ilk gelen ilk işlem görür şeklindedir) ve olası çizelge sayısı (n!) ile ifade edilir.

Akış tipi çizelgeleme problemlerinin temel varsayımları aşağıdaki gibidir (Ruiz ve Maroto, 2005, s. 479-480):

1. *i* işi birim zamanda tek bir *j* makinesinde işlem görebilir.
2. *j* makinesi birim zamanda tek bir *i* işini yapabilir.
3. İşlerin bölünmesine izin verilmez, öyle ki *j* makinesi *i* işini yapmaya başladığı zaman kesintiye uğramaksızın tamamlaması gerekir.
4. Bütün işler birbirinden bağımsızdır ve 0 zamanında hazırdır.
5. İşlerin makinelerdeki hazırlık zamanları önemsizdir ve çizelgeleme sırasında göz ardı edilebilir.
6. Makineler çizelgeleme boyunca çalışır durumdadır.
7. Süreç içi stoka izin verilir. Herhangi bir iş, çizelgenin bir sonraki adımında ihtiyaç duyduğu makine meşgulse, ilgili makinenin kuyruğuna girerek bekleyebilir.

Bu varsayımlar gevşetilerek hibrid (melez) akış tipi, farklı hazırlık süreli, ara kuyruksuz, işlerin bölünebilir olabildiği gibi çeşitli alt problemler türetilmiştir (Gupta ve Stafford, 2006, s. 703).

Akış tipi çizelgeleme problemlerinde en yaygın kullanılan amaç, en geç tamamlanma zamanının (en son makinedeki en son işin tamamlanması – makespan) en küçüklenmesi olarak tanımlanmakta ve çözüm ise bunu sağlayacak iş sırasının bulunmasından oluşmaktadır.

### 1.4.1. Permütasyon Akış Tipi Çizelgeleme Problemi

Her bir $j_i$ *(i=1, 2,…n)* işi, tüm makinelerde aynı sırayı takip ederek m farklı makinede $M_k$ *(k=1, 2,…, m)* işlenecektir. Herhangi bir işin *m.* makinedeki işlem süresi $p_{ik}$ olarak gösterilsin. Problemde temel amaç, her bir makinedeki iş sırasının aynı olduğu ve toplam en geç bitirme zamanı değerini en küçükleyecek iş çizelgesinin bulunmasıdır.

Başka bir ifadeyle, $\pi(i)$, $\pi$ çizelgesinde *i.* pozisyonda yer alan işi ifade etmek üzere, işlerin oluşturduğu her bir çizelge $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ permütasyonuyla tanımlanabilir. *J* iş seti üzerinde bu şekilde oluşturulabilen permütasyonların tümünün kümesi $\Pi$ olsun.

*C(k, $J_i$) $J_i$* işinin *k* makinesindeki tamamlanma zamanı olsun. *N*-iş, *m*-makine durumundaki akış tipi çizelgeleme probleminin tamamlanma zamanı aşağıdaki gibi hesaplanır:

$$C(1,\pi_1) = p_{1,\pi_1} \quad (1)$$

$$C(1,\pi_j) = C(1,\pi_{j-1}) + p_{1,\pi_j}, \quad j = 2,\dots,n \quad (2)$$

$$C(k,\pi_1) = C(k-1,\pi_1) + p_{k,\pi_1}, \quad k = 2,\dots,m \quad (3)$$

$$C(k,\pi_j) = mak\left\{C(k,\pi_{j-1}), C(k-1,\pi_j) + p_{k,\pi_j}\right\}, \quad j = 2,\dots,n;\ k = 2,\dots,m \quad (4)$$

En geç tamamlanma zamanı kriteri;

$$C_{mak}(\pi) = C(m,\pi_n) \quad (5)$$

Dolayısıyla en geç tamamlanma zamanı kriterine sahip PFSP'de istenen amaç bir $\pi^*$ permütasyonu bulmaktır:

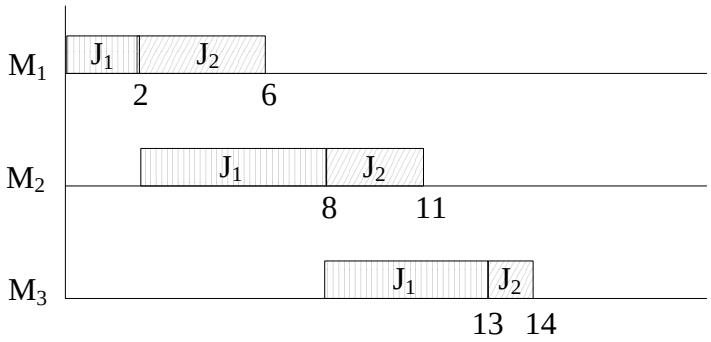
$$C_{mak}(\pi^*) \leq C(m,\pi_n) \quad \forall\pi \in \Pi \quad (6)$$

Aşağıdaki örnekte iki iş, üç makine durumu verilmiştir (Laha, 2008, s. 4). Örnekteki sistem iki farklı işe sahip olduğu için işlere ait 2! = 2 adet permütasyon olacaktır. Bunlar $J_1$-$J_2$ ve $J_2$-$J_1$ iş çizelgeleridir. Bu iki durumu şekil üstünde göstermek istersek:

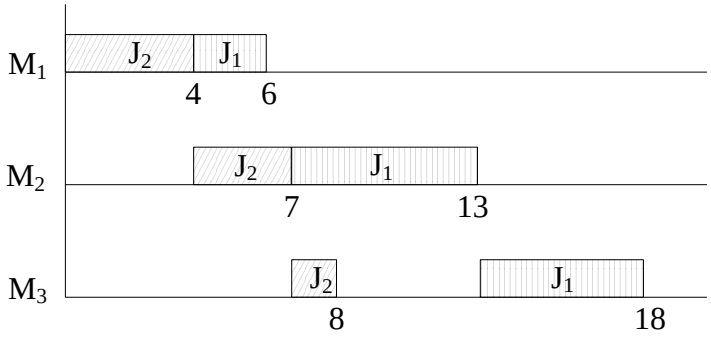
**Tablo 1.1 Örnek Bir Çizelgeleme Sorusu**

| **İş** | **Makine** | | |
|---|---|---|---|
| | $M_1$ | $M_2$ | $M_3$ |
| $J_1$ | 2 | 6 | 5 |
| $J_2$ | 4 | 3 | 1 |

Verilenler doğrultusunda $J_1$-$J_2$ çizelgesinin en geç tamamlanma zamanı 14 iken $J_2$-$J_1$ çizelgesinin ise 18'dir.




**Şekil 1.3 $J_1$-$J_2$ İş Çizelgesi**




**Şekil 1.4 $J_2$-$J_1$ İş Çizelgesi**

### 1.5. Çözüm Yaklaşımları

Akış tipi çizelgeleme problemlerinde özellikle maksimum tamamlanma kriteri için literatürde pek çok çalışma yer almaktadır. Klasik optimizasyon yöntemleriyle (dal-sınır, dinamik programlama) sadece küçük ölçekli problemler çözülebilmektedir. Daha büyük boyutlu problemler ise sezgisel yöntemlerle çözülmektedir.

İki makine n-iş problemleri için optimum çözüm veren ilk çalışma 1954 yılında Johnson tarafından yapılmıştır. Önerilen algoritma Johnson Algoritması olarak bilinmektedir. Makine sayısının ikiden fazla olduğu durumlar için yapılan çalışmalarda problemin NP-Zor grubuna girdiği belirtilmiştir.

Bu zorluk, pek çok araştırmacının dikkatini çekmiştir ve 1960'lardan itibaren kabul edilebilir sürede optimale yakın veya iyi sonuçlar verecek çeşitli sezgisel yöntemler geliştirilmiştir. Özellikle 1990'lı yıllardan sonra yapay zeka tabanlı yöntemlerin çoğalmasıyla, akış tipi çizelgeleme problemleri geliştirilen sezgisel yöntemlerin performanslarının kıyaslanması için en

çok kullanılan problem türlerinden birisi haline gelmiştir. Bunun en önemli sebeplerinden birisi iş çizelgelemenin zorlayıcı ve rekabetçi bir problem türü olmasıdır. Bu özellikten dolayı geliştirilen yeni algoritmaların performanslarının ölçülmeleri ve farklı algoritmalarla kıyaslamalı avantajlarını ortaya koymak için sıklıkla tercih edilmektedir.

Literatürde genellikle farklı amaç değerlerine yönelik sezgisel ve metasezgisel yöntemler geliştirilmiştir. Bu göstergelerden bazıları en geç tamamlanma zamanı, toplam akış süresi, işlerin gecikme süresi ve geciken iş sayısıdır. Bu çalışmada en geç tamamlanma süresi performans kriteri olarak kullanılacağı için takip eden bölümdeki eserler bu amaç değerine yönelik çözüm yaklaşımlarını içermektedir.

Sezgisel yaklaşımların sınıflandırılmasında iki ana yaklaşım dikkat çekmektedir. Birinci yaklaşım sezgisel çözüm yöntemleri geliştirmeyi üç aşamalı bir model olarak tanımlar (Framinan vd., 2004, s. 1244). İkincisi ve en sık kullanılan sınıflandırma yöntemi ise sezgisel yöntemleri iki ana gruba ayırır (Laha, 2008; Ruiz ve Maroto, 2005; Zobolas vd., 2008; Ponnambalam vd., 2001).

Framinan vd. (2004), tarafından yapılan ve PFSP için en geç tamamlanma zamanı minimizasyonuna dayanan yöntemlerin incelendiği literatür taramasında sezgisel çözüm yöntemlerinin yapıcı ve geliştirici olarak iki ana başlıkta sınıflandırılması eleştirilmiş ve model geliştirmeyi üç aşamalı olarak incelemişlerdir. Önerilen 3 aşama:

1. Endeks geliştirme
2. Çözüm oluşturma
3. Çözüm iyileştirme

adımlarını içerir.

Birinci adım olan endeks geliştirmede işler makinelerdeki işlem süreleri gibi belirgin özelliklerine göre sıraya konulur. Başlangıç sıralamasını oluşturmak için iki ana yaklaşım bulunmaktadır, bunlar “örneksel problem” kullanma veya kullanmamadır. Örneksel problem kullanılmayınca işlerin işlem süreleri ve aralarındaki ilişkiler göz önüne alınır. Bu sıralamaya ilk

örnek Palmer'ın eğim endeksi metodudur. Eğim endeksini kullanarak geliştirilen diğer yöntemler ise Hundal ve Rajgopal (1988), Gupta (1971), Gupta (1976), Bonney ve Gundry (1976) ile Koulamas (1998)'dir.

Örneksel problem kullanarak iş sırası türeten literatürde üç farklı yöntem yer almaktadır. Bunlar $F_2//C_{max}$, TSP (Gezgin Satıcı Problemi) ve vektör toplam yöntemleridir.

$F_2//C_{max}$, aynı zamanda makine toplaması yöntemi olarak da bilinir. En yaygın metot ise optimal çözümün bulunabilir olmasından dolayı $m=2$ makine durumuna indirgemektir. Campbell vd. (1970), Dannenbrig (1977), Lai (1996), Rock ve Scmidt (1983), Selim ve Al-Turki (1987) sezgiselleri bu yönteme örnektir.

TSP yönteminde ise işlem sürelerinden oluşturulan bir uzaklık matrisi sayesinde TSP problemi çözülerek başlangıç sıralaması elde edilir. Bu gruba örnekler Gupta (1968), Stinson ve Smith (1982), Widmer ve Hertz (1991) ile Maccellin (1995)'dir.

Son yöntem ise vektör toplamı teorisine dayanmaktadır ve örnekleri Sevast'jonou (1995) ile Lourensa (1996) tarafından verilmiştir.

İkinci aşama olan çözüm oluşturma süreci hangi işin seçileceği ve hangi konuma ekleneceği alt adımlarında incelenebilir. Literatürde pek çok seçim ve ekleme kriteri yer almasına rağmen bunları kısaca tekli seçim/tekli ekleme ve çoklu seçim/çoklu ekleme olarak tanımlayabiliriz.

Üçüncü aşamada ise mevcut başlangıç çözümü, ondan daha iyi veya en azından ona eşit bir çözüm bulununcaya kadar belirli bir süre boyunca geliştirilmeye çalışılır. Çözüm geliştirme yöntemlerini azalan yerel arama (Page (1961), Dannenbring (1977)'nin son aşaması, Aggorwal ve Stafford (1975)'in son aşaması, Ho ve Chang (1991), King ve Spachis (1980)) ve metasezgisel yöntemler (Tabu araması, Genetik algoritma, Karınca kolonisi optimizasyonu, Yapay bağışıklık sistemi, Parçacık sürüsü optimizasyonu vd.) olmak üzere iki gruba ayırabiliriz.

Literatürde en sık kullanılan ikili sınıflandırma da ise çözüm teknikleri yapıcı ve geliştirici sezgiseller olarak sınıflandırılır. Birinci grup yapıcı veya geleneksel yöntemlerden oluşur

(constructive heuristics) ve sıfırdan olası bir çizelge oluşturmaya çalışırlar. İkinci grup ise geliştirmeli yöntemlerdir (improvement methods). Bu yöntemler verilen bir başlangıç çözümüyle başlayarak komşuluk arama esasıyla mevcut çözümü geliştirmeye çalışırlar. Metasezgisel yöntemleri de bu gruba dahil edebiliriz. Örnek yöntemlerden bazıları benzetim tavlaması, yapay sinir ağları, genetik algoritmalar, karınca kolonisi optimizasyonu ve tabu araması yaklaşımıdır.

Çalışmanın takip eden bölümlerinde ele alınacak eserler bu sınıflandırma altında incelenecektir.

### 1.5.1. Sezgisel Yöntemler

#### 1.5.1.1. Yapıcı Sezgiseller

Bu gruba ilk örnek iki makine durumu için en iyi çözümü veren Johnson(1954) algoritmasıdır. Dudek ve Teuton (1964), Johnson algoritmasından faydalanarak son makinedeki toplam boş zamanı en küçükleyecek m-aşama kuralını geliştirmişlerdir. Campbell vd. (1970), Johnson algoritmasının geliştirilmiş bir hali olarak tanımlanabilecek bir algoritma geliştirmiştir. Bu yöntemde çok sayıda çizelge oluşturulup içlerinden en iyisi seçilir. CDS olarak bilinen yöntem, m adet makineyi iki sanal makine altında kümeleyerek *m-1* tane çizelge oluşturur ve bu iki makineli durumu Johnson algoritması ile çözer (Ruiz ve Maroto, 2005, s. 481).

İkincil bir yaklaşım ise her bir işe ağırlık veya endeks değeri atayarak, bu değerlere göre sıralama yapmaktır. Bununla ilgili ilk çalışma Palmer (1965) tarafından yapılmış ve algoritmada her bir iş için bir eğim endeksi hesaplanarak, işlerin bu endeksin azalan sırası ile sıralanması sağlanmıştır. Gupta (1971) ile Banney ve Gundry (1976) Palmer'ın eğim endeksini geliştirerek kendi sezgisel yöntemlerini oluşturmuşlardır. Hundal ve Rajgopal (1988) ise Palmer'ın eğim endeksinin yanı sıra iki farklı eğim endeksi daha hesaplayarak bu üç endeksin oluşturduğu üç farklı çizelgelemeden en iyisini seçmişlerdir (Ruiz ve Maroto, 2005, s. 481).

Dannenbring (1977) tarafından geliştirilen Hızlı Erişim (Rapid Access – RA) yöntemi Johnson algoritması ve Palmer'ın eğim endeksi yöntemlerinin bir karışımı biçimindedir. Modelde CDS'de olduğu gibi sanal iki makine problemi oluşturulur, sonra her bir makine için ağırlıklandırılmış şemalar hesaplanır ve bunlara Johnson algoritması uygulanır. Buradaki ağırlıklandırılmış şemalar, sanal makinelerdeki işlerin işlem süresidir.

Page (1961), PFSP'nin sıralama yöntemlerine olan benzerliğinden yola çıkarak işlerin iyi bir sırasını elde etmeye yarayan ve bu sıralamayı iş değiş-tokuşuyla geliştiren bir sezgisel yöntem geliştirmiştir (Ruiz ve Maroto, 2005, s. 481).

Nawaz vd. (1983) tarafından geliştirilen NEH sezgiseli, PFSP'de maksimum akış zamanının en küçüklenmesi kriteri için pek çok yazar tarafından en iyi sezgisel yöntem olarak kabul edilir. Yöntemin temel dayanağı, tüm makinelerdeki en yüksek işlem süreli işin çizelgelemede mümkün olduğunca en erken sıraya konmasıdır.

Yöntemin işleyişi:

1. $P_i = \sum_{j=1}^{m} p_{ij}$ formülü ile işlerin toplam işlem süresi hesaplanır,
2. İşler $P_i$'nin azalan sırasına göre sıralanır. Daha sonra ilk iki iş (en yüksek $P_i$ sahip ilk iki iş) seçilir ve olası iki çizelge değerlendirilir,
3. $i = 3,4,\ldots,n$ işleri sırayla alınır ve o iş daha önce hazırlanan çizelgedeki mümkün olan her yere konularak elde edilen çizelgeler kıyaslanır. Örneğin, $i=4$ için, eldeki mevcut çizelge ilk üç işi içerdiği için, dört numaralı iş birinci, ikinci, üçüncü ve sonuncu sıraya atanarak elde edilen dört farklı çizelgeden en iyisi bir sonraki iterasyon için seçilir.

Sarin ve Lefoka (1993) son makinedeki boş zamanın minimize edilmesi (son makinedeki boş zamanın artması, tamamlanma zamanının ve maksimum akış zamanının artmasına yol açar) fikrinden yola çıkmışlardır. Önerdikleri yöntemde, her seferinde sadece bir işin çizelgeye eklenmesi ve eklenecek işin seçimi içinse çizelgeye eklendiğinde *m.* makinedeki boş zamana katkısı en az olması prensibinden yararlanmışlardır. Gupta (1972) daha önce bahsedilen fikirlerden yararlanarak minimum boş zaman (MINIT), minimum tamamlanma zamanı (MICOT) ve MINIMAX adında 3 farklı sezgisel yöntem önermiştir. Bu yöntemlerden ilk ikisi iş çiftlerinin değiş-tokuşuna, MINIMAX ise Johnson kuralına dayanmaktadır.

#### 1.5.1.2. Geliştirmeli Sezgiseller

Bu yöntemler daha önceden oluşturulmuş bir başlangıç yerleşimini iyileştirmeyi hedeflemektedirler. Genellikle iki veya üç aşamadan oluşabilen metotların ilk aşamasında yapısal sezgisellerle bir başlangıç çizelgesi belirlenirken, ikinci veya üçüncü aşamalarda elde

edilen bu başlangıç çözümünden daha iyi çizelgeler bulunmaya çalışılır (Ruiz ve Maroto, 2005, s. 482-483; Liu vd., 2007, s. 18).

Dannenbring (1977) iki basit sezgisel yöntem önermiştir. Bunlar RACS (Rapid Access with Close Order Search) ve RAES'dir (Rapid Access with Extensive Search). Her iki yöntemde başlangıç çözümünü RA ile elde eder. RACS'te sıralamadaki tüm komşu iş çiftleri yer değiştirmektedir ve elde edilen *n-1* çizelgeden en iyisi seçilir. RAES'te ise iyileştirmeler bulunduğu müddetçe RACS tekrar edilir.

Ho ve Chang (1991) boşluk olarak tanımladıkları bir işin bir makinede bittiği süreyle bir sonraki işleneceği makinedeki başlama süresi arasındaki zamanı minimize etme fikrini ortaya koymuşlardır. Algoritma mümkün bütün iş ve makine çiftleri için boşlukları hesaplayıp, bazı hesaplamalar ile bu boşluk değerlerine göre sıralamayı değiştirir. Yöntem başlangıç çözümü için CDS'den faydalanır.

Sulimon (2000) tarafından önerilen sezgisel yöntem, CDS ile elde edilen başlangıç çözümünü iş çiftlerinin yer değiştirmeleriyle güncellemeye çalışır. İkili değiş-tokuşun oluşturduğu hesaplama yükünü azaltmak için yönlülük kıstı uygulanmıştır. Örneğin, bir işi ileri doğru taşıyarak elde edilen çizelge daha iyi sonuca sahipse, bundan sonraki iyi işlerin ileri doğru taşımayla elde edileceği varsayılır.

Agarwal vd. (2006) çalışmasında yazarlar, yapay sinir ağlarındaki uyarlanabilir öğrenme stratejisine benzer bir yaklaşımı çizelgeleme problemine uyarlamıştır. İşlerin ağırlıklandırılmış işlem zamanlarını kullanarak çalışan sezgisel yöntemde iş ağırlıkları bir öğrenme stratejisi ile her iterasyonda güncellenmektedir. Eğer belli bir denemeden sonra öğrenme stratejisi çözümü geliştiremezse bu seferde rastgele belirlenen bir sayı ve algoritmada sabit olarak kullanılan bir katsayıyla yeni ağırlıklar belirlenerek modelin kötü sonuçlar üreten çözüm bölgesinden kaçması sağlanır. Modelin başlangıç çözümü ise Palmer, CDS veya NEH ile oluşturulmaktadır. Önerilen algoritma yukarıdaki üç sezgisele uyarlanarak, tek başlarına ürettikleri sonuçlarla literatürde yer alan 172 adet test problemi aracılığıyla karşılaştırılmışlardır. Birlikte kullanıldıklarında oldukça başarılı sonuçlar verdiği ifade edilmiştir, hatta 43 problemde bilinen en iyi değerlerin elde edildiği de belirtilmiştir.

**Tablo 1.2 En Geç Tamamlanma Süresi Kriteri İçin Geliştirilen Sezgisel Teknikler**

| Çalışma | Algoritma | Tür | Notlar |
|---|---|---|---|
| Johnson (1954) | Johnson | Y | İki makine durumu için kesin çözüm |
| Page (1961) | Page | Y | Sıralamaya dayalı |
| Dudek ve Teiton (1961) | - | Y | Johnson kuralına dayalı |
| Palmer (1965) | Palmer | Y | Eğim endeksi |
| Campbell vd. (1970) | CDS | Y | Johnson kuralına dayalı |
| Gupta (1971) | - | Y | Eğim endeksine dayalı |
| Gupta (1972) | MINIT, MICOT, MINMAX | Y | İşlerin yer değişim ve Johnson kuralı |
| Bonney ve Gundry (1976) | - | Y | Eğim endeksine dayalı |
| Dannenbring (1977) | RA | Y | Johnson ve Palmer modellerine dayalı |
| Nawaz vd. (1983) | NEH | Y | İş öncelikleri / yer değiştirme |
| Hundal ve Rajgopal (1988) | - | Y | Palmer eğim endeksine dayalı |
| Sarin ve Lefoka (1993) | - | Y | Son makinedeki boş zamanın minimizasyonu |
| Dannenbring (1977) | RAES, RACS | G | RA ile elde edilen başlangıç çözümünü geliştirme |
| Ho ve Chang (1991) | - | G | Başlangıç çözümü için CDS, İşler arasındaki boşluk minimizasyonu |
| Suliman (2000) | - | G | Başlangıç çözümü için CDS, İş çiftlerinin yer değişimi |
| Agarwal vd. (2006) | - | G | Ağırlıklandırılmış işlem sürelerini kullanır |

**Y:** *Yapıcı sezgisel*, **G:** *Geliştirmeli sezgisel*

### 1.5.2. Metasezgisel Yöntemler

Metasezgisel kelimesi bulmak anlamına gelen "sezgisel" ile daha üst bir seviye anlamına gelen "meta" adındaki iki Yunanca kelimeden gelmektedir. Literatürde yer alan metasezgisel terimine ilişkin çeşitli tanımların ortak noktaları aşağıda verilmiştir (Blum ve Roli, 2003, s. 270-271).

- Metasezgiseller araştırma sürecine rehberlik eden stratejilerdir,
- Amaç optimal/optimala yakın çözümler bulabilmek için araştırma uzayını etkili bir biçimde keşfetmektir,

- Metasezgisel teknikler basit yerel araştırma tekniklerinden karmaşık öğrenme süreçlerine kadar çeşitlilik gösterir,
- Metasezgisel algoritmaları yaklaşık sonuçlar verir ve genellikle deterministik değillerdir,
- Araştırma uzayındaki kimi noktalarda (yerel minimum) tuzağa düşmeyi önleyecek mekanizmalar içerebilirler,
- Problem tabanlı değildirler,
- Araştırmayı kolaylaştırmak için bilgi tabanlı olabilirler (TS'de tabu listesi kullanımı gibi)

Metasezgisel yöntemlerin sınıflandırılmasında kullanılan kriterlerden en yaygın olan ikisi aşağıdaki gibidir (Blum ve Roli, 2003, s. 272-273).

1. Doğadan esinlenilen veya doğadan esinlenmeyen yöntemler. Doğadaki canlıların sosyal birlikteliklerini temel alan bu yöntemlerden bazıları Karınca Kolonisi Optimizasyonu (ACO), Genetik Algoritma (GA), Parçacık Sürüsü Optimizasyonu (PSO) ve Yapay Arı Kolonisi (ABC) algoritmalarıdır. İteratif Yerel Arama (ILS) ve Tabu Araması (TS) doğadan esinlenmeyen yöntemlerden bazılarıdır.
2. Popülasyon tabanlı ve tek nokta araştırması yöntemleri. GA, ABC, PSO, ACO gibi aynı anda araştırma uzayının birden fazla noktasını araştırabilen yöntemler birinci gruba girmektedirler. Yerel araştırma tabanlı yöntemler, TS, Benzetim Tavlaması (SA), ILS ise tek bir çözüm üzerine odaklanmış yörüngesel yöntemlerdir.

Literatürde en geç tamamlanma zamanı kriterine sahip PFSP probleminin çözümünde kullanılan metasezgisel teknikler ve bu tekniklerden yararlanılan çalışmalardan örnekler aşağıda verilmiştir.

#### 1.5.2.1. Benzetim Tavlaması (SA)

Benzetim tavlaması (Simulated Annealing) algoritması, bir metalin soğuyarak ve donarak minimum enerjili kristal yapısına dönüşmesi (tavlama süreci) ile daha genel bir sistemde minimumun araştırılması arasındaki benzerlikten faydalanır. Tavlama sürecinde, malzemenin belirli bir sıcaklığa kadar ısıtılması, bu sıcaklıkta bir süre tutulması ve belirli bir stratejiye göre sıcaklığın oda sıcaklığına kadar azaltılması aşamalarından oluşur (Karaboğa, 2004, s. 21-22; Cura, 2008, s. 43-44).

Metropolis vd. (1958) tarafından geliştirilen tavlama sürecinin optimizasyon problemlerine ilk defa uygulanması Kirkpatrick vd. (1983) tarafından yapılmıştır. Önerilen algoritmada, mevcut *i* halindeki enerji $E_i$ ve bir sonraki *j* halindeki enerji $E_j$ olsun. Buna göre bir sonraki *j* hali, *i* halinin bozulmasıyla elde edilir. Eğer enerji farkı, $E_j - E_i \leq 0$ ise *j* hali mevcut hal olarak kabul edilir. Aksi halde *j* hali reddedilmez, belirli bir olasılığa bağlı olarak kabul edilir. Benzetim tavlamasının kötü çözümleri kabul etme özelliği sayesinde, algoritmanın yerel optimumda sıkışması önlenir (Cura, 2008, s. 46).

SA algoritmasının gerçekleştirilmesi dört adımdan oluşur (Karaboğa, 2004, s. 30):

1. Başlangıç çözümünün üretilmesi için bir metodun bulunması,
2. Komşu üretme mekanizmasının tanımlanması,
3. Komşuluğun nasıl araştırılacağının tanımlanması,
4. Soğutma sürecinin tanımlanması.

SA algoritmasının adımları aşağıda verilmiştir (Karaboğa, 2004, s. 37):

**Adım 1.** Sezgisel olarak veya rastgele bir başlangıç çözümü S üret. Başlangıç sıcaklığı için $T_s$ değerini ve durdurma kriterini belirle.

**Adım 2**. Komşu bir çözüm, $S^a$, üret ve bu üretilen çözümle S çözümün amaç değerlerini arasındaki farkı $\Delta = C(S^a) - C(S)$ hesapla.

**Adım 3.** Şayet;

(i) $S^a$, S'den daha iyi ($\Delta < 0$) ise

veya

(ii) $S^a$, S'den daha kötü ama mevcut sıcaklık T'de $\left(e^{\frac{-\Delta}{T}}\right) > \theta$ (Burada θ, 0< θ<1 aralığında seçilen rastgele bir sayı) olasılığına göre kabul edilir.

S çözümünü $S^a$ ile yer değiştir.

Yoksa, S çözümünü muhafaza et.

**Adım 4.** Belirli kurallar setine göre T sıcaklığı düşür.

**Adım 5.** Durdurma kriteri sağlanıyorsa araştırmayı durdur, yoksa Adım 2'ye git.

Osman ve Potts (1989) değişken ve rassal komşuluğa dayanan basit bir benzetim tavlaması algoritması önermiştir. Ogbu ve Smith (1990), Palmer ve Dannenbring sezgisellerini başlangıç noktası olarak ele alan bir SA modeli kullanmıştır. Wodecki ve Bozejko (2001) paralel SA algoritması oluşturarak, sonuçlarını NEH ile kıyaslayarak, bulunan sonuçların daha iyi olduğunu belirtmiştir (Ruiz ve Maroto, 2005, s. 483).

Ishibuchi vd. (1995) sıcaklık soğuma çizelgeleri farklılaşan iki SA algoritması önermiştir. Zegordi vd. (1995) SA algoritmasını çizelgelemeyle ilgili bilgilerle birleştirerek daha az kontrol parametresi içeren ve çeşitli kurallarla tavlama sürecini basitleştiren bir model geliştirmişlerdir. Bu kurallar İşler için Hareketlilik Tercihi tablosuna dayanmaktadır. Bu algoritmalar genellikle Osman ve Potts (1989) ile kıyaslanmışlardır. Sonuçlarda performans olarak Ogbu ve Smith (1990)'nın çok az geride kaldığı, Ishibuchi vd. (1995)'in başa baş olduğu, Zegordi vd.(1995)'in ise çok az geride kaldığı fakat çok daha hızlı olduğu görülmektedir (Ruiz ve Maroto, 2005, s. 484).

Nearchou (2004), çalışmasında hibrid SA tabanlı bir model önerilmiştir. Modelde GA'da olduğu gibi rassal olarak bir başlangıç popülasyonu oluşturulmakta ve bu popülasyonda öncelikle yerel arama algoritması ile komşu çözümler geliştirilmektedir. Geliştirilen komşu çözümler, orijinallerinden daha iyiyse kabul edilmekte, daha kötü olanlar ise belirli bir olasılığa göre kabul edilebilmektedirler. Oluşturulan çözüm kümesine NEH sezgiselinden esinlenerek düzenlenmiş bir iteratif tepe tırmanışı (iterative hill climbing) algoritması uygulanarak ikincil bir yerel arama süreci daha uygulanmaktadır. Daha sonra SA'daki soğutma süreci uygulanarak model belirli bir tekrar sayısınca çalıştırılmaktadır.

Çalışmada öncelikle yerel arama için çeşitli komşuluk modelleri (iki ardışık noktanın yer değişimi, iki rastgele işin yer değişimi, tekli yer değiştirme, grup olarak kaydırma, bir grubu tersine çevirme, bir grubun yerini değiştirip tersine çevirme) denenerek en başarılı olanın iki rastgele işin yer değişimi olduğu belirlenerek modele dahil edilmiştir.

Nearchou tarafından geliştirilen algoritma GA (Reeves, 1995), SA (Osman ve Potts, 1989; Ogbu ve Smith, 1990; Ishibuchi vd., 1995) ile kıyaslanmıştır. Sonuçlar modelin üstünlüğünü göstermektedir.

### 1.5.2.2. Tabu Araması (TS)

Tabu araması (Tabu Search) algoritması, F. Glover tarafından optimizasyon problemlerinin çözümü için geliştirilmiş iteratif bir araştırma algoritmasıdır. Yöntemde temel yaklaşım, son çözüme götüren adımın, dairesel hareketler yaratmasını engellemek için bir sonraki döngüde tekrarının yasaklanması veya cezalandırılmasıdır (Karaboğa, 2004, s. 49-50; Cura, 2008, s. 65).

Algoritma yerel minimuma doğru hareket ederek başlar. Daha önce yapılmış hareketlere tekrar dönüş yapmayı engellemek için yöntem bir veya daha fazla tabu listesi tutar. Listenin orijinal amacı, önceden yapılmış bir hareketin tekrarından çok tersine dönmesini önlemektir. Tabu listesi kronolojik bir yapıya sahiptir ve tabu arama belleğini biçimlendirir. Belleğin rolü algoritma ilerledikçe değişebilir. Başlangıçta hedef, çözüm uzayında kaba araştırma yapmaktır. Aday konumlar belirlendikçe arama yerel optimum çözümü üretmeye daha fazla odaklanır. (Cura, 2008, s. 65).

Algoritmanın temel adımları aşağıda verilmiştir (Karaboğa, 2004, s. 51):

**Adım 1.** Bir başlangıç çözümü (S) al, başlangıç parametrelerini belirle.
**Adım 2.** Komşu çözümler üret ve bu çözümler arasından en iyi kabul edilebilir olanı ($S^a$) seç. ($S^a$, tabu listesinde olmayan tüm çözümler içinden seçilir).
**Adım 3.** Mevut çözümü (S), $S^a$ ile değiştir ve tabu listesini yenile.
**Adım 4.** Durdurma kriteri sağlanıncaya kadar Adım 2 ve Adım 3'ü tekrarla.

Widmer ve Hertz (1989) SPIRIT adında iki aşamalı bir sezgisel oluşturmuştur. İlk aşamada Açık TSP ile oluşturulan problem, ekleme metoduyla çözülüp başlangıç dizilimi elde edilir. İkinci aşamada çözüm tabu araması ile geliştirilir. Reeves (1993) başlangıç çözümü için NEH sezgiselinden ve ekleme komşuluğundan faydalanacak şekilde SPIRIT algoritmasını değiştirmişlerdir. Yapılan denemelerde Osman ve Potts (1989)'dan daha iyi performans gösterdiği belirtilmiştir.

Taillard (1990), çalışmada öncelikle en yaygın kullanılan sezgisel yöntemler olan Gupta, Johnson, Palmer, RA, CDS ve NEH yöntemlerini maksimum akış zamanının minimizasyonu kriteri için çeşitli büyüklükteki test problemleri kullanarak sonuçları kıyaslamışlardır. Yapılan

karşılaştırmada NEH sezgiselinin en başarılı yöntem olduğu belirtilmiştir. Daha sonra NEH yöntemini hızlandırmak için yeni bir seçim süreci tanımlanmıştır. Aynı problemler yazar tarafından geliştirilen TS tekniği ile çözülüp sonuçlar karşılaştırılmıştır. Yazara göre TS tekniği NEH'den daha iyi sonuçlar vermesine rağmen, problem boyutunun artması gereken işlem süresini de arttırmaktadır. Bu işlem zorluğunun aşılması için ise paralel TS tekniği önerilmektedir.

Moccellin (1995) tarafından SPIRIT sezgiseline dayanan TS tabanlı bir algoritma geliştirilmiştir. Sadece ilk aşamadaki işler arasındaki uzaklığın hesaplanması süreci ve TSP probleminin en uzak eklemeli TSP ile çözümünde farklılaşmıştır. Nowicki ve Smutnicki (1996) komşuluk ilişkisini işlerin tek tek yer değişimi yerine, işlerin kümelenmesi sonucu elde edilen blokların yer değiştirmesi prensibiyle azaltmaya çalıştıkları bir TS algoritması geliştirmiştir. Taillard'ın test problemleri için literatürdeki en iyi sonuçları verdiği ifade edilmiştir.

Ben-Daya ve Al-fawzan (1998), en geç tamamlanma süresinin minimizasyonu için TS tabanlı bir algoritma geliştirmişlerdir. Süreci hızlandırmak için yoğunlaştırma ve ayrıştırma gibi ek özelliklerden yaralanmışlardır. Modelde başlangıç çözümü NEH ile oluşturulmuştur. Komşu çözümleri oluşturmak için ise rassal ekleme, blok ekleme ve rassal yer değiştirme gibi çeşitli yerel arama yöntemlerinden birisi rastgele seçilmiştir. Oluşturulan model Taillard'ın test problemleri kullanılarak Taillard'ın TS tekniği ile Ogbu ve Smith (1990)'da yer alan SA modeli ile kıyaslanmıştır. Sonuçların Ogbu ve Smith'in modelinden daha iyi olduğu, Taillard'ın TS modeline ise çok yakın olduğu fakat ondan daha hızlı olduğu belirtilmiştir.

En geç tamamlanma süresi kriterli PFSP için TS tabanlı bir başka modelde Grabowski ve Wodecki (2004) çalışmasında önerilmiştir. Geliştirilen algoritmanın, akış tipi problemlerin bloklaşma özelliğinden yararlanarak geleneksel TS'ye göre daha hızlı çalışması sağlanmıştır. Ayrıca değişken boyutlu tabu listesi kullanarak yerel minimumdan kaçınılmaya çalışılmıştır. Modelin performansı Nowicki ve Smutnicki (1996) ile Grabowski ve Pempera (2000) TS algoritmaları ile kıyaslanarak onlardan daha iyi sonuçlar verdiği belirtilmiştir.

Ekşioğlu vd. (2008), çalışmalarında 3XTS adında bir TS algoritması geliştirilmiştir. Önerilen modelde komşuluk geliştirme için üç farklı yerel arama metodu (ekleme, iki noktalı değişim -

2EX, üç noktalı değişim - 3EX) kullanılmıştır. Geliştirilen algoritmada yeni komşu çözüm üretmek için öncelikle 3EX uygulanmakta ve belirli bir tekrar süresince daha iyi bir sonuç elde edilemezse 2EX tekniğine geçilmektedir. Bu teknik ile de belirli bir tekrar sayısı boyunca sonuçta herhangi bir gelişme olmazsa ekleme tekniği uygulanmaktadır. Bu üç yerel arama tekniği de çözümü iyileştiremezse o permütasyon tabu listesine alınmaktadır.

Modelin kıyaslaması öncelikle Salimanpur vd. (2004)'te yer alan Neuro-TS algoritması ile yapılmıştır. Adı geçen eserde, yazarlar modellerinin GA (Reeves, 1995) ve SA (Osman ve Potts, 1989; Ogbu ve Smith, 1991) tekniklerinden daha iyi sonuçlar verdiğini belirtmiştir. 3XTS ile yapılan kıyaslamada, başa baş olan sonuçların büyük boyutlu problemlerde 3XTS'den yana olduğu ifade edilmiştir. Ayrıca çalışma hızı olarak 3XTS'nin daha başarılı olduğu belirtilmiştir. Çalışmada model son olarak Ying ve Liao (2004)'te yer alan ACS ile kıyaslanarak, sonuçların daha başarı olduğu tespit edilmiştir.

#### 1.5.2.3. Genetik Algoritma (GA)

Doğal seleksiyonu temel alan evrimsel algoritmaların başında genetik algoritma gelmektedir. GA, doğal evrimin benzetimidir. Bilindiği gibi doğada evrim en uygun olanların hayatta kalmasıyla gerçekleşmektedir. Zayıf bireyler yeni bireyleri ya çok az üretebilecek kadar yaşarlar ya da hiçbir yeni birey oluşturamadan ölürler. Ancak daha güçlü olan bireyler daha uzun süre hayatta kalarak fazla sayıda yeni birey üretebilecektir. GA'da optimize edilecek fonksiyonun parametreleri *kromozomlar* içinde yer alan *genler* şeklinde kodlanır. Başlangıçta rastlantısal olarak belirlenmiş bireylerden (kromozomlar) bir popülasyon havuzu oluşturulur. Daha sonra birey çiftleri fonksiyon optimizasyonundaki performanslarına bağlı olarak bu havuzdan seçilirler. Bu iki birey, ebeveynlerinin karakteristiklerini barındıran yeni bir çocuk oluşturur (*çaprazlama*). Bazen beklenmeyen rastgele değişiklikler çocuk oluşumunda rol oynayabilir, bu duruma *mutasyon* denir. Oluşan çocuklar ebeveynlerinin yerini alırlar ve yeni bir jenerasyon ortaya çıkar. Artık popülasyon havuzunda yeni bireyler bulunmaktadır ve süreç yeniden başlayarak bir sonraki jenerasyon oluşturulur (Cura, 2008, s. 87-88).

Algoritmanın işleyişi aşağıda verilmiştir (Karaboğa, 2004, s. 79):

**Adım 1.** Başlangıç popülasyonunu oluştur.

**Adım 2.** Popülasyondaki her çözümün uygunluk değerini hesapla.

**Adım 3.** Durdurma kriteri sağlanıyorsa araştırmayı durdur. Yoksa, aşağıdaki adımları gerçekleştir:

3.1. Doğal seleksiyon işlemini uygula (uygunluk değeri daha yüksek olan çözümler yeni popülasyonda daha fazla temsil edilirler).

3.2. Çaprazlama işlemini uygula (Mevcut iki çözümden yeni iki yapı üretilir).

3.3. Mutasyon işlemini uygula (Çözümlerden rastgele değişim meydana getirilir).

**Adım 4.** Adım 2'ye git.

Chen vd. (1995) başlangıç çözümünün CDS ve RA sezgiselleri ile bazı bireylerin basitçe yer değiştirmesi ile elde edilen popülasyona dayandığı bir GA modeli önermiştir. Modelde sadece PMX (Partially Mapped Crossover) adındaki çaprazlama operatörü kullanılmış, herhangi bir mutasyona yer verilmemiştir. Reeves (1995) algoritmanın her bir adımında üretilen bireylerin ebeveynlerle değil ortalamanın altında uygunluk değerine sahip jenerasyonlardaki bireylerle değiştirildiği bir GA modeli önermiştir. Çaprazlama için C1 (tek nokta sıralı çaprazlama) operatörü kullanılmıştır. Modelde ayrıca uyarlanabilir mutasyon oranı yer almakta ve mutasyon basit bir şekilde bir işin yerini değiştirmektedir. Daha iyi çözüm üretebilmek için başlangıç popülasyonu NEH sezgiseli ve rastsal olarak oluşturulan bireylerden oluşmaktadır.

Reeves ve Yamada (1998) MSXF (Multi-Step Crossover Fusion) denen ve çaprazlama operatörü ile yerel aramayı birleştiren hibrid bir GA modeli oluşturmuşlardır. MSXF operatörü bir ebeveyni referans alarak diğer ebeveyn doğrultusunda sapmalı (ön yargılı) yerel arama yapar. Murata vd. (1996) elitist strateji içeren, kaydırma mutasyonu (shift mutation – tesadüfi olarak seçilen bir genin, yine tesadüfü olarak belirlenen sağ veya soldaki bir pozisyona kaydırılması) ve iki-noktalı çaprazlama operatörü içeren bir GA modeli önermişlerdir. Belirlenen algoritma TA, SA ve yerel arama algoritmalarından daha kötü sonuçlar vermiştir. Yazarlar bunun üzerine genetik yerel arama ve genetik SA adında iki hibrid GA oluşturmuştur. Geliştirilen algoritmalarda seçim ve çaprazlama öncesinde yerel arama ve SA ile yapılan bir geliştirme fazı

eklenmiştir. Sonuçların ilk önerilen hibrid GA, SA, TS ve yerel arama tekniklerinden daha iyi olduğu belirtilmiştir.

Ponnambalam vd. (2001), çalışmasında maksimum tamamlanma süresi kriteri için GA bazlı bir model geliştirilmiş ve performansı Gupta, Palmer, CDS, RA ve NEH sezgiselleri ile kıyaslanmıştır. Çalışmada öncelikle 21 farklı test problemi Gupta, Palmer, CDS, RA ve NEH ile çözülerek en iyi çözüm değerleri elde edilmiştir. Bu en iyi değerler daha sonra GA ile elde edilen değerlerle kıyaslanmıştır. Ele alınan 21 problemden 11'inde GA'nın daha başarılı olduğu belirtilmiştir.

Wang ve Zheng (2003) çalışmada bir hibrid GA modeli oluşturulmaya çalışılmış. Çalışmada öncelikle basit GA (SGA) modeli ile başlangıç popülasyonunun NEH ile oluşturulduğu SGA ile kıyaslanmıştır. Başlangıç popülasyonunun rastgele oluşturmak yerine NEH ile oluşturulmasının daha kısa sürede daha iyi sonuçlar verdiği tespit edilmiştir.

Ayrıca mutasyon ve çaprazlama operatörlerinin SGA performansı üzerindeki etkisini değerlendirmek için 4 farklı çaprazlama operatörü (C1 – tek noktalı çaprazlama, LOX – doğrusal sıralı çaprazlama, PMX – kısmi haritalandırılmış çaprazlama, NABEL) ile üç farklı mutasyon operatörü (yer değiştirme, tersini alma, ekleme) kullanılmıştır. Yapılan kıyaslama sonucunda performans anlamında istatistiksel bir farklılığın oluşmadığı belirtilmiştir. Bu doğrultuda oluşturulan hibrid GA modelinin aşamaları:

1. Başlangıç popülasyonu için NEH ile çözüm üret ve popülasyonda kalan çözümler rassal olarak üret,
2. Popülasyonu eşit büyüklükte dört alt popülasyona böl ve her birisine çaprazlama için C1, LOX, PMX veya NABEL operatörlerinden birisini uygula,
3. Bireylerin mutasyona uğrama olasılığı SA modelindeki soğutma fonksiyonundan faydalanılarak belirlenir. Daha etkin bir arama süreci için popülasyonun yarısına yer değiştirme, geri kalan kısmının bir yarısına tersini alma, diğer yarısına da ekleme yerel arama metodları uygulanmıştır.

Yukarıdaki algoritmayla oluşturulan hibrid GA'nın sonuçları literatürde yer alan NEH ve en iyi GA sonuçları ile kıyaslandığından onlardan daha başarılı olduğu şeklindedir.

Etiler vd. (2004) CDS, Dannenbring sezgiselleri ve bunlara rastgele yer değiştirme uygulanarak elde edilen çözümlerden oluşan başlangıç popülasyonu içeren bir GA modeli geliştirmiştir. Modelin kıyaslaması iki farklı test problem grubu kullanılarak NEH ve GA (Chen vd. 1995) ile yapılmıştır. NEH ile yapılan karşılaştırmada modelin %76, NEH sezgiselinin ise %40 başarı gösterdiği belirtilmiştir. GA modelleri arasındaki kıyaslamada ise önerilen modelin %78, Chen'in GA modelinin ise %38 düzeyinde başarı gösterdiği, bu farkın özellikle çaprazlamada kullanılan operatörlerin farklı olmasından kaynaklandığı belirtilmiştir.

Ponnambalam vd. (2004), çalışmalarında çok amaçlı PFSP için TSP tabanlı bir GA modeli önermişlerdir. Önerilen GA modelinde başlangıç çözümü için bir uzaklık matrisi oluşturarak buradan TSP yaklaşımı ile başlangıç çözümü elde edilir. Elde edilen çözümde rassal eklemeler yaparak ortaya çıkan çizelgelerle de başlangıç popülasyonu oluşturulur. Çalışmada maksimum tamamlanma süresi, ortalama akış süresi ve makine boş kalma süresi gibi üç farklı amaç değerinin ağırlıklandırılmış toplamları amaç fonksiyonu değeri olarak kullanılmıştır. Çalışmada yazarlar tarafından ağırlık değerleri her seçim sürecinde rassal olarak değişmektedir. Bu sayede problemin doğrusal bir şekilde tek bir yönde gelişmesi engellenmeye çalışılmıştır. Model ile 21 adet test problemi çözülerek, sonuçları verilmiştir.

Iyer ve Saxena (2004), çalışmalarında GA operatörlerinin problem türüne olan etkisini incelemiştir. Temel GA ve problem tabanlı bilgiye sahip bir yerel arama tekniğiyle revize edilmiş GA'nın performanslarını karşılaştırmışlardır.

Ruiz vd. (2006), çalışmasında GA tabanlı iki farklı algoritma önerilmiştir. Önerilen modelde farklı seçim türü ve olasılığı, çaprazlama türü, çaprazlama olasılığı, mutasyon olasılığı gibi GA'nın temel bileşenlerinde en uygun değerler deney tasarımı ile belirlenmiştir. Elde edilen GA modeline yerel arama eklenerek ikinci alternatif model oluşturulmuştur. Oluşturulan modeller NEH (Nawaz vd., 1983; Taillard, 1990), GA (Chen vd., 1995; Murata vd., 1996; Reeves, 1995), SA (Osman ve Potts, 1989), TS (Widmer ve Herts, 1989), ILS (Stützle, 1998a), ACO (Rajendlan

ve Ziegler, 2004) gibi literatürde yer alan 13 farklı alternatif ile kıyaslanmış ve yerel aramayı içeren GA modelinin diğerlerinden daha iyi sonuçlar verdiği belirtilmiştir.

Watase (2007), akış tipi çizelgeleme probleminin GA ile çözüm performansının geliştirilmesi için klasik sezgisel yöntemlerden NEH arama metodundan faydalanmıştır. Çalışmada permütasyonların gösterimi için faktöriyel sayılardan yararlanılmıştır. GA'da yer alan kromozomlar bu faktöriyellerden oluşmuştur. Geliştirilen yöntem literatürdeki çeşitli test problemlerine uygulanarak sonuçları NEH ve farklı çaprazlama operatörleri (tek noktalı sıralı çaprazlama, orta noktalı sıralı çaprazlama, tüm son noktalar sıralı çaprazlama, önerilen faktöriyel çaprazlama) içeren GA modelleri ile kıyaslanmıştır. Sonuçların önerilen yöntemden yana olduğu belirtilmiştir.

Zobolas vd. (2009), üç bileşenli bir hibrid metasezgisel model önermiştir. Modelin ilk aşamasında başlangıç popülasyonu oluşturulmaktadır. Popülasyon NEH, CDS, Palmer ve Gupta sezgisellerinden birer çözümün yanı sıra rassal olarak oluşturulan çözümleri ve NEH sezgiselinin açgözlü rassal arama metoduyla birlikte oluşturulmasıyla elde edilen çözümleri içerir. Bu çözümün geliştirilmesi GA ile yapılırken değişken komşuluk arama yöntemiyle de popülasyonun iyileşmesi sağlanır. Modelin Taillard'ın test problemleri üzerine yapılan denemelerinde kısa sürede başarılı sonuçlar verdiği görülmektedir. Ayrıca ACO (Rajendran ve Ziegler, 2004), Hibrid GA (Ruiz vd., 2006), $PSO_{VNS}$ (Tasgetiren vd., 2007), SA (Osman ve Potts, 1989), TS (Widmer ve Hertz, 1989) ile yapılan kıyaslamalarında hibrid GA, $PSO_{SPV}$ ve önerilen algoritmanın en iyi sonuçları verdiği belirtilmiştir. Test problemlerinin içerdiği 12 farklı boyuttan beşinde en iyi sonuçların elde edildiği ifade edilmiştir.

Tseng ve Lin (2009), en geç tamamlanma ve toplam akış süresi kriterleri için GA tabanlı bir algoritma geliştirilmiştir. GA iki farklı yerel arama tekniğiyle güncellenerek çözüm kalitesi iyileştirilmiştir. Oluşturulan hibrid algoritma, her iki performans kriteri içinde iyi sonuçlar vermekle beraber, özellikle toplam akış süresi kriteri için ele alınan 90 problemden 66'sında en iyi çözümü geliştirmiştir.

### 1.5.2.4. Evrimsel Gelişim Algoritması (DE)

Storn ve Price (1997) tarafından geliştirilen Evrimsel Gelişim (Differential Evolution) algoritması, GA gibi evrimsel gelişim stratejilerini içeren popülasyon tabanlı bir tekniktir. GA'da olduğu gibi çaprazlama, mutasyon ve seçim operatörleri burada da kullanılmaktadır. Farkı ise, her bir operatör tüm popülasyona sırayla uygulanmamaktadır. Kromozomlar tek tek ele alınmakta ve rastgele seçilen üç farklı kromozomda kullanılarak yeni bir birey elde edilmektedir. Bu işlemler sırasında mutasyon ve çaprazlama operatörleri kullanılmaktadır. Mevcut kromozom ve elde edilen yeni kromozomun uygunluk değerleri karşılaştırılmakta, uygunluğu daha iyi olan birey bir sonraki popülasyona aktarılmaktadır (Keskintürk, 2006, s. 87).

Tasgetiren vd. (2004), maksimum tamamlanma süresi kriterine sahip PFSP için evrimsel gelişim tabanlı bir algoritma önermiştir. Storn ve Prize (1997) tarafından doğrusal olmayan sürekli fonksiyonlar için geliştirilen DE algoritması, yazarlar tarafından SPV kuralı yardımıyla kesikli optimizasyona uyarlanmıştır. Oluşturulan $DE_{SPV}$ algoritmasına ayrıca basit bir yerel arama operatörü (rassal değişim) eklenerek performansı GA ve $PSO_{SPV}$ ile kıyaslanmıştır. Yerel arama özellikli $PSO_{SPV}$'nin en iyi sonuçları verdiği belirtilmiştir.

Onwubolu ve Davendra (2006) maksimum tamamlanma süresi, toplam akış zamanı ve geçliği minimize etmek için bir DE algoritması geliştirmiş ve önerilen model klasik genetik algoritma ile karşılaştırılmıştır. Yapılan kıyaslama sonucu, modelin başa baş bir performans gösterdiği ve algoritmanın özel problemlere kolaylıkla uyarlanabilir olduğu belirtilmiştir. DE algoritmasının sürekli sayılarla çalışmasından dolayı, yazarlar bir dönüşüm şeması yardımıyla değişkenleri kesikli yapıya indirgemişlerdir.

Pan vd. (2008b), evrimsel gelişim algoritmasını ve iteratif yerel arama algoritmalarını hem en geç tamamlanma kriteri hem de toplam akış süresi minimizasyonu kriteri için PFSP problemine uygulamışlardır. Yazarlar daha iyi çözümler geliştirebilmek için, her iki yöntemi birleştirmişler ayrıca oluşturulan hibrid algoritmaya kendi geliştirdikleri yeni bir yerel arama tekniği eklemişlerdir. Oluşturulan algoritmanın performansı, her iki kriter için Taillard test problemlerinde denenmiş ve sonuçların başarılı olduğu hatta bazı test problemlerinde bilinen en iyi sonuçlardan daha iyilerinin bulunduğu belirtilmiştir.

### 1.5.2.5. Karınca Kolonisi Optimizasyonu (ACO)

Karınca Kolonisi Optimizasyonu (Ant Colony Optimization) algoritması, gerçek karıncaların beslenme davranışlarından ilham alan bir yöntemdir. Karıncalar, yuvaları ve yemek kaynakları arasındaki en kısa yolu bulma yeteneğine sahiptir. Karınca kolonisinin bu karmaşık davranışı, karıncaların yemek kaynağına seçtikleri bir yoldan giderken feromon (koku) bırakması, başka bir ifadeyle birbirleri ile dolaylı haberleşmesiyle mümkün olmaktadır. Takip eden karıncalar, en kuvvetli feromon içeren yolu tercih etme eğiliminde olurlar. Böylece bir yoldaki mevcut feromon miktarı tazelenir (artar) veya azalır. Kısa bir yol üzerinde karıncalar daha sık gidip geleceğinden, daha fazla feromon izi bırakacaktır. Benzer biçimde, uzun yollar üzerinde feromon miktarı azalacağı için daha az tercih edilir hale gelecektir (Cura, 2008, s. 113-114).

Dorigo vd. (1991) tarafından Karınca Sistemi (Ant System) adında ilk kez optimizasyon problemlerine uyarlanan ACO algoritmasından günümüze kadar Mak-Min Karınca Sistemi, Ant-Q algoritması ve karınca Kolonisi Sistemi gibi çeşitli türevler geliştirilmiştir.

Mak-Min Karınca Sistemi (MMAS) tabanlı bir ACO algoritması Stützle (1998a) tarafından önerilmiştir. Literatürde PFSP için ACO tabanlı ilk çalışmanın olduğu belirtilmiştir. MMAS'de normal Karınca Sisteminden farkı, karıncaların yollarını bulurken faydalandıkları feromen izlerinin belirli bir alt-üst sınır arasında sınırlandırılmasıdır. Bu sayede yeni çözümlerin geliştirilmesiyle eskilerinin korunması arasında bir denge sağlanmaya çalışılmıştır. Elde edilen algoritma NEH, NEH+yerel arama, Osman ve Potts (1989)'daki SA ve rassal başlangıç çözümlü çoklu azalış algoritmaları ile kıyaslanmış ve en iyi sonucu verdiği belirtilmiştir.

Ying ve Liao (2004) Karınca Kolonisi Sistemi (ACS) algoritması kullanan bir çalışmadır. Önerilen model Taillard'ın test problemleri ile literatürde yer alan GA, SA ve komşu arama yerel arama tekniği ile kıyaslanarak onlardan daha üstün sonuçlar verdiği belirtilmiştir.

Rajendran ve Ziegler (2004) tarafından iki farklı ACO algoritması önerilmiştir. İlk modelde Stützle (1998a)'de önerilen MMAS modeli yeni geliştirilen iş endeksi tabanlı yerel arama tekniği ile güncellenmiştir. Geliştirilen ikinci model ise klasik ACO'ya göre başlangıç çözümünün oluşturulmasında, çözümün geliştirilmesinde ve rotanın güncellenmesinde yazarlar tarafından bazı değişiklikler önerilmiştir.

Geliştirilen iki algoritma da MMAS ile maksimum tamamlanma süresi ve toplam akış zamanı kriterleri için kıyaslanmıştır. Maksimum tamamlanma süresi kriteri için her iki algoritma da daha iyi performans göstermiştir. Toplam akış zamanı kriteri için ise Lin ve Reeves (2001) çalışmasında yer alan test problemleri için elde edilen en iyi değerler baz alınmıştır. Geliştirilen her iki modelinde bu sonuçlardan daha iyi değerler ürettiği belirtilmiştir.

Yağmahan ve Yenisey (2006)'da yazarlar karınca kolonisi algoritması ile problemin çözümünde sonuçların kalitesini arttırmak için Taguchi'nin deney tasarımından faydalanarak modele ait beş farklı parametrenin en iyi değerlerini bulmaya çalışmışlardır. Parametrelerin bulunan en iyi, en kötü ve ortalama değerlerini kullanarak Taillard'ın test problemleri üzerinde modelin performansını incelemişlerdir. Elde edilen parametre değerlerinin sonuç performansını olumlu yönde geliştirdiği belirtilmiştir.

Yağmahan ve Yenisey (2008), maksimum tamamlanma süresi, toplam akış zamanı ve toplam makine boş zamanlarını minimize etmeyi amaçlayan çok amaçlı PFSP için ACS temelli bir model geliştirilmiştir. Oluşturulan ACS modeli, NEH ve Ho ve Chang (1991) ile kıyaslanmıştır. Her üç amaç içinde ayrı ayrı test edilen modeller, daha sonra çok amaçlı programlama için $\Delta f = \alpha_1 f_1 + \alpha_2 f_2 + \alpha_3 f_3$ $(\alpha = 0.33)$ eşitliği ile tek amaca indirgenmiştir. Maksimum tamamlanma süresi amacı için en iyi sonucu NEH sezgiseli verirken diğer iki amaç ve çok amaçlı model için ise ACS en başarılı model olmuştur.

Yağmahan ve Yenisey (2010), maksimum tamamlanma süresi ve toplam akış zamanını eş zamanlı minimize etmeyi amaçlayan çok amaçlı ACS modelini önerilmişlerdir. Modelde ayrıca yerel arama için komşu çiftlerin yer değiştirmesi metodu kullanılmıştır. Önerilen modelde elde edilen çizelgelerin maksimum tamamlanma süresi ve toplam akış zamanı değerleri hesaplanarak bunların ağırlıklandırılmış toplamlarıyla da çok amaçlı problemin amaç değeri elde edilmiştir. Çeşitli test problemleri üzerinde denenen modelin sonuçları GA, Rajendron (1995) tarafından önerilen CR(MC), Ravindron vd. (2005) tarafından önerilen HAMC1, HAMC2 ve HAMC3 algoritmaları ile kıyaslanmıştır. Tekil ve çoğul amaçlar için ACS modelinin en iyi sonuçları verdiği belirtilmiştir.

### 1.5.2.6. İteratif Yerel Arama (ILS)

Stützle (1998b) tarafından geliştirilen İteratif Yerel Arama (Iteratif Local Search) tekniği; yerel arama, SA, TS gibi modellerden bileşenler içeren bir metasezgisel modeldir. NEH yöntemi ile elde edilen başlangıç çözümüne sırasıyla yerel arama, değiştirme ve tekrar yerel arama prosedürleri uygulanmaktadır. Elde edilen çözümün kabul edilip edilmemesi SA alınan soğuma fonksiyonu ile belirlenmektedir. Yazar çalışmasında sonuçları Taillard ile Nowicki ve Smutnicki'nin TS algoritmalarından daha iyi olduğunu belirtmiştir.

Pan vd. (2008a)'de yazarlar beklemesiz akış tipi problemi ele almışlardır. Bu tür problemlerde işlemler arasında bekleme yoktur, yani bir iş ilk makinede işleme başlayınca herhangi bir kesintiye uğramadan tüm makinelerden geçmek zorundadır. Gerekirse bir işin ilk makinede başlaması bu şartın sağlanabilmesi için ertelenebilmektedir.

Çalışmada geliştirilen model geliştirilmiş aşamalı açgözlü algoritmaya (improved iterated greedy algorithm) dayanmaktadır. Modelin ilk aşamasında ekleme komşuluğu için hızlandırıcı bir ilave yapılmış, ikinci aşamada başlangıç çözümünü iyileştirmek için geliştirilmiş NEH sezgiseli önerilmiş ve üçüncü aşamada çözüm uzayının etkili bir biçimde araştırılması için ise iteratif açgözlü algoritma kullanılmıştır.

Önerilen algoritmanın performansı parçacık sürüsü optimizasyonu (Pan vd., 2005) ile Grabowski ve Pempera (2005) tarafından önerilen üç farklı TS yöntemiyle kıyaslanmıştır. Önerilen yöntemin daha kısa sürede daha iyi sonuçlar ürettiği belirlenmiştir.

### 1.5.2.7. Parçacık Sürüsü Optimizasyonu (PSO)

Doğadaki kuş ve balık sürülerinin toplu hareketlerinden esinlenilen ve parçacıkların sürü halindeki toplu hareketini temsil eden bir algoritmadır. Bu parçacıkların her biri bir çözüm adayını temsil eder ve bir optimizasyon probleminin mümkün çözüm uzayını araştırmak için kullanılırlar. Her bir parçacık başlangıç aşamasında tesadüfi veya sezgisel olarak belirlenir ve daha sonra serbest harekete bırakılır. Algoritmanın her bir adımında, her parçacık kendi ve çevresindekilerin uygunluğunu ölçer, bu uygunluk değeri optimum değere olan uzaklık olarak düşünülür (Cura, 2008, s. 135).

PSO (Particle Swarm Optimization) algoritması kuş sürülerini taklit etmektedir. Kuşların yerini bilmedikleri yiyeceği aramaları, problem için çözüm aramaya benzetilir. Kuşlar yiyecek ararken yiyeceğe en yakın olan kuşu takip ederler. Her çözüm veya parçacık, çözüm uzayındaki bir kuştur. Parçacık kendi koordinatlarını bir uygunluk fonksiyonuna gönderir ve parçacığın uygunluk değeri bulunduğu konum ile ölçülmüş olur. Her parçacığın koordinatı, hızı, o ana kadar elde ettiği en iyi uygunluk değeri ve bu değeri elde ettiği koordinatları bilmesi gerekir. Çözüm uzayının her boyutundaki hızının ve yönünün çevrimlerdeki değişimi, komşularının en iyi koordinatları ve kendi en iyi koordinatlarının bir birleşimi olacaktır (Cura, 2008, s. 137).

Rameshkumar vd. (2005), PFSP için PSO algoritmasını kullanmışlardır. Yazarlar tarafından önerilen algoritma literatürdeki çeşitli test problemlerine uygulanmış ve sonuçların daha da geliştirilmesi için de çalışmanın ikinci bölümünde yerel arama tekniklerinden faydalanmışlardır.

Lian vd. (2006) çalışmasında PSO algoritmasını PFSP gibi kesikli problemlere uyarlamak için bir dönüşüm metodu önerilmiştir. Oluşturulan model geleneksel GA ile kıyaslanmış ve performansının daha iyi olduğu belirtilmiştir.

Duda (2006)'da yazar GA ve PSO algoritmalarına yerel arama teknikleri ile birleştirdikleri hibrid modelleriyle klasik halleri arasında performans farkının olup olmadığını araştırmaya çalışmıştır. Çalışmada GA modeli NEH sezgiseli ile birleştirilmiş ve elde edilen hibrid model ILS yerel arama tekniği ile kıyaslanmıştır. Taillard'ın test problemleri ile yapılan analizde ortalama değerlere bakılarak hibrid modelin basit yerel arama tekniğinin gerisinde kaldığı belirtilmiştir. Çalışmanın ikinci kısmında $PSO_{SPV}$ algoritması En Küçük Pozisyon Değeri (SPV) yerel arama tekniğiyle kıyaslanmış ve SPV ile elde edilen sonuçların daha başarılı olduğu belirtilmiştir.

Tasgetiren vd. (2007) çalışmasında PSO algoritması yardımıyla akış tipi çizelgeleme problemi maksimum tamamlanma süresi ve toplam akış zamanı kriterleri için çözülmüştür. Sürekli optimizasyon uygulamaları için geliştirilen PSO algoritmasındaki parçacıkların sürekli pozisyon değerleri, yazarlar tarafından SPV sezgiseli yardımıyla kesikli iş permütasyonuna çevrilmiştir. Ayrıca yerel arama için Değişken Komşuluk Araması (SPV) sezgiseli kullanılarak iki farklı PSO ($PSO_{SPV}$, $PSO_{VNS}$) algoritması geliştirilmiştir.

Maksimum tamamlanma süresi kriteri için Taillard'ın test problemleri aracılığıyla GA-$PSO_{SPV}$ ve VNS-$PSO_{VNS}$ karşılaştırmaları yapılmıştır. Sonuçlar doğrultusunda $PSO_{SPV}$'nin GA'dan daha iyi performans gösterdiği fakat daha fazla işlem süresi kullandığı, $PSO_{VNS}$ modelinin ise VNS'den daha üstün sonuçlar verdiği ve ortalama hesap süresinin de düştüğü belirtilmiştir.

Çalışmada $PSO_{VNS}$ algoritması Watson vd. (2002)'de yer alan iki farklı NEH ve iki farklı TS yöntemleri ile yine aynı çalışmada yer alan ve Watson'un geliştirdiği test problemlerine uyarlanmıştır. Yapılan karşılaştırma sonucu $Tabu_{sh}$ metodunun en iyi sonuçları verdiği fakat $PSO_{VNS}$ modelinin problemlerin %66'sında en iyi sonucu çok kısa bir sürede bulduğu belirtilmiştir.

Toplam akış zamanı kriteri için ise Taillard'ın test problemleri kullanılarak $PSO_{VNS}$ algoritması, bu problemler için Liu ve Reeves (2001) çalışmasında literatürdeki çeşitli yöntemlerin karşılaştırılması ile bulunan en iyi değerlerle ve Rajendran ve Ziegler (2004) çalışmasında yer alan iki farklı ACO algoritması ile kıyaslanmıştır. Ele alınan 90 problemden 57 tanesinde en iyi sonucun algoritma tarafından geliştirildiği belirtilmiştir.

Liu vd. (2007) Memetik Algoritma olarak tanımladıkları hibrid PSO tabanlı bir model önermişlerdir. Başlangıç popülasyonu rastgele oluşturulan çözümlerden ve NEH sezgiseli ile elde edilen çözümden oluşan algoritmada, sıra tabanlı değer kuralı ile sürekli parçacıklardan oluşan çözüm değerleri kesikli iş çizelgelerine dönüştürülmüştür. PSO tabanlı arama sürecinin yanı sıra popülasyona NEH ve çeşitli yerel arama teknikleri uygulanarak çözümün geliştirilmesi sağlanmıştır. Modelin performansı $PSO_{VNS}$ (Tasgetiren vd., 2004) ve GA (Wang ve Zheng, 2003) ile yapılmış ve sonuçlar modelin üstünlüğünü ortaya koymuştur.

Jarboui vd. (2008), en geç tamamlanma kriteri için yeni bir PSO tabanlı çözüm algoritması geliştirmiştir. Çalışmada ayrıca PSO algoritmasının çözümlerini geliştirmek için SA algoritmasından yararlanılmış ve elde edilen çözümler hem en geç tamamlanma hem de toplam akış süresi kriterleri için literatürdeki diğer metasezgisel algoritmalarla karşılaştırılmıştır.

Lian vd. (2008), PSO algoritmasını kesikli optimizasyona uyarlamışlardır. Algoritma çeşitli yerel arama teknikleriyle geliştirilerek, performansı GA ile kıyaslanmıştır.

Liu vd. (2011) çalışmasında yazarlar PSO algoritmasını dağılım tahmini algoritması (estimation of distribution algorithm – EDA) ile birleştirmişlerdir. Yazarlar ayrıca kendi geliştirdikleri bir yerel arama tekniğini de modele eklemişlerdir. Oluşturulan algoritma, en geç tamamlanma zamanı kriterli PFSP problemi için Reeves ve Watson test problemlerine uygulanmıştır. Sonuçlar modelin üstünlüğünü vurgulamaktadır.

#### 1.5.2.8. Diğer Algoritmalar

Ruiz ve Maroto (2005) tarafından yapılan literatür taramasında, literatürde yer alan 18 farklı sezgisel yöntemin (14 sezgiselin yanı sıra rastgele çözüm (rastgele çizelgeler üretip en iyisini seçme) ve üç farklı atama kuralı (ilk gelen ilk hizmet alır, en kısa işlem süresine göre atama ve en uzun işlem süresine göre atama) performansları Taillard'ın test problemleri üzerinde karşılaştırılmıştır. Sonuçlar NEH sezgiselinin en iyi sonucu verdiği şeklinde olmuştur. Yazarlar ayrıca yedi farklı metasezgisel yöntemin (SA (Osman ve Potts, 1989; Widmer ve Hertz (1989), GA (Chen vd., 1995; Reeves, 1995; Ponnambalam vd., 2001), ILS, GA-yerel arama (Murata vd., 1996)) performanslarını kıyaslayarak ILS ve Reeves tarafından oluşturulan GA modelinin daha iyi sonuçlar verdiği belirtilmiştir.

Yukarıda bahsedilen ve sıklıkla kullanılan algoritmalar dışında, metasezgisel tabanlı farklı algoritmalara ait çeşitli uygulamalarla da karşılaşılmıştır. Bunlardan bazıları kısaca aşağıda anlatılmıştır.

Nowicki ve Smutnicki (2006) çalışmasında yazarlar dağınık arama yöntemini (Scatter Search - SS) PFSP için uyarlamış ve çeşitli komşuluk özelliklerinin çözüm uzayı üzerindeki etkilerini incelemişlerdir. Kökeni evrimsel algoritmalara dayanan SS algoritması, GA ile kıyaslandığında küçük bir popülasyonla başlayarak çeşitli çaprazlama/mutasyon süreçleriyle yeni çözümler geliştirilir. Oluşturulan algoritma Taillard'ın test problemlerine uygulanmış ve başarılı sonuçlar alınmıştır.

Ruiz ve Stützle (2007) çalışmasında basit bir açgözlü algoritma geliştirilmiştir. Önerilen modelde başlangıç çözümü NEH sezgiseli ile oluşturulmakta ve elde edilen çözümde belli sayıda istasyon çıkartılarak bunlar rastgele çeşitli konumlara atanmaktadır. Elde edilen çeşitli çizelgelerden en iyisi seçilmekte, ayrıca çözüm geliştirilemezse de SA'da yer alan sabit sıcaklı bir kabul fonksiyonu ile daha kötü çözümler seçilebilmektedir. Modelin karşılaştırması literatürde yer alan çeşitli GA, SA, TS, ACO ve ILS algoritmaları ile kıyaslanarak sonuçlarının üstünlüğü belirtilmiştir.

Sayadi vd. (2010) çalışmasında yazarlar doğadaki ateş böceklerinin davranışlarından esinlenerek oluşturulan ateş böceği algoritması (Firefly Metaheuristic – FM) PFSP için uyarlanmıştır. Algoritmanın temeli, ateş böceklerinin yaydıkları ışığın kuvvetine göre birbirlerine doğru hareket etmesi prensibidir. Yöntemde amaç fonksiyonu değeri ateş böceği tarafından yayılan ışığın şiddetini vermektedir. Literatürdeki çeşitli test problemleri için ACO (Ying ve Lin, 2007) ile kıyaslanan yöntemin daha başarılı olduğu belirtilmiştir.

Akış tipi çizelgeleme problemlerinin çözümü için ABC algoritması ile yapılan çalışmalar ikinci bölümde anlatılacaktır.

**Tablo 1.3 PFSP İçin Kullanılan En Geç Tamamlanma Süresi Amaçlı Metasezgisel Yöntemler**

| Çalışma | Algoritma | Notlar |
|---|---|---|
| Osman ve Potts (1989) | SA | |
| Ogbu ve Smith (1990) | SA | Başlangıç çözümü için Palmer ve Dannenbring sezgiselleri |
| Ishibuchi vd. (1995) | SA | |
| Zegordi vd. (1995) | SA | |
| Wodecki ve Bozejko (2001) | Paralel SA | |
| Nearchou (2004) | Hibrit SA | Yerel arama, NEH ve SA |
| Widmer ve Hertz (1989) | TS | Başlangıç çözümü için OTSP |
| Taillard (1990) | TS | |
| Reeves (1993) | TS | Başlangıç çözümü için NEH ve yerel arama |
| Moccellin (1995) | TS | Spirit yöntemine dayanır |
| Nowicki ve Smutnicki (1996) | TS | Bloklar halinde yer değiştirme |
| Ben-Daya ve Al-Fawzan (1998) | TS | Başlangıç çözümü NEH ve yerel arama |
| Grabowski ve Wodecki (2004) | TS | |
| Ekşioğlu vd. (2008) | TS | Çözümü iyileştirmek için üç farklı yerel arama metodu |
| Reeves (1995) | GA | Uyarlanabilir mutasyon oranı |
| Chen vd. (1995) | GA | PMX çaprazlama operatörü |
| Murata vd. (1996) | Hibrit GA | GA+yerel arama, GA+SA |
| Reeves ve Yamada (1998) | GA | MSXF çaprazlama operatörü, Bilgi tabanlı yerel arama |
| Ponnambalam vd. (2001) | GA | |
| Wang ve Zheng (2003) | Hibrit GA | Başlangıç popülasyonu için NEH sezgiseli, C1, LOX, PMX, NABEL çaprazlama operatörleri, mutasyon olasılığı için SA soğuma fonksiyonu |
| Etiler vd. (2004) | GA | Başlangıç popülasyonu CDS, Dannenbring ve rassal olarak oluşturulan çizelgeleri içerir |
| Ponnambalam vd. (2004) | GA | Başlangıç çözümü için TSP ile elde edilen çözümden rastgele çizelgeler oluşturarak elde edilir, çok amaçlı amaç fonksiyonu |

**Tablo 1.3 PFSP İçin Kullanılan En Geç Tamamlanma Süresi Amaçlı Metasezgisel Yöntemler (… devam)**

| Çalışma | Algoritma | Notlar |
|---|---|---|
| Iyer ve Saxena (2004) | GA | |
| Ruiz vd. (2006) | GA | GA parametrelerini deney tasarımı ile bulma, GA + yerel arama |
| Watase (2007) | GA | Faktöriyel gösterim |
| Tseng ve Lin (2009) | Hibrit GA | |
| Zobolas vd. (2009) | Hibrit GA | Üç aşamalı GA tabanlı bir model. Başlangıç çözümü için NEH, CDS, Palmer ve Gupta. Komşuluk oluşturma için NEH ve aç gözlü rassal arama |
| Tasgetiren vd. (2004) | DE | SPV sezgiseli yardımıyla algoritmayı kesikli optimizasyona uyarlama |
| Onwubolu ve Davendra (2006) | DE | En geç tamamlanma süresi, toplam akış zamanı ve geçliğin en küçüklenmesi. Dönüşüm şeması yardımıyla algoritmanın kesikli optimizasyona uyarlanma |
| Pan vd. (2008a) | DE, ILS | DE ve ILS algoritmaları bağımsız olarak ve DE+ILS şeklinde hibrid bir algoritma olarak en geç tamamlanma süresi ve toplam akış zamanı kriterlerine uygulanmıştır |
| Stützle (1998a) | ACO | MMAS |
| Ying ve Liao (2004) | ACO | ACS |
| Rajendran ve Ziegler (2004) | ACO | İki farklı ACO algoritması. En geç tamamlanma süresi ve toplam akış zamanı kriteri için ayrı ayrı denemiştir |
| Yağmahan ve Yenisey (2006) | ACO | Parametreler için deney tasarımı |
| Yağmahan ve Yenisey (2008) | ACO | En geç tamamlanma süresi, toplam akış zamanı ve toplam makine boş zamanı kriterleri için ayrı ayrı ve çok amaçlı ACS sistemi |
| Yağmahan ve Yenisey (2010) | ACO | En geç tamamlanma süresi ve toplam akış zamanı kriterlerinin eş zamanlı min. için ACS sistemi |

**Tablo 1.3 PFSP İçin Kullanılan En Geç Tamamlanma Süresi Amaçlı Metasezgisel Yöntemler (… devam)**

| Çalışma | Algoritma | Notlar |
|---|---|---|
| Stützle (1998b) | ILS | NEH, yerel arama ve SA içerir |
| Ruiz ve Stützle (2007) | IG | Başlangıç çözümü için NEH, çözümü geliştirmek için yerel arama teknikleri ve SA tabanlı kabul fonksiyonu kullanılır. |
| Pan vd. (2008b) | Diğer | Yerel arama, NEH, İteratif aç gözlü algoritma bileşenlerini içerir |
| Rameshkumar vd. (2005) | PSO | |
| Lian vd. (2006) | PSO | Dönüşüm formülleri ile PSO algoritmasını kesikli optimizasyona uyarlama |
| Tasgetiren vd. (2007) | PSO | SPV yerel arama tekniği ile PSO algoritmasını kesikli optimizasyona uyarlama. VNS yerel arama tekniği eklenmiş. En geç tamamlanma süresi ve toplam akış zamanı kriterleri için ayrı ayrı test edilmiş. |
| Liu vd. (2007) | Hibrit PSO | PSO, NEH, çeşitli yerel arama teknikleri ve SA tabanlı yerel arama beraber kullanılmıştır |
| Jarboui vd. (2008) | PSO | PSO ve PSO+SA hibrid algoritması |
| Lian vd. (2008) | PSO | PSO'nun kesikli optimizasyona uyarlanması |
| Liu vd. (2011) | PSO | Dağılım tahmini algoritması ile hibrid PSO |
| Nowicki ve Smutnicki (2006) | SS | |
| Sayadi vd. (2010) | FM | |
| Duda (2006) | GA, PSO | Hibrit GA & GA, Hibrit PSO & PSO kıyaslaması |

**SA:** *Benzetim Tavlaması*, **GA:** *Genetik algoritma*, **TS:** *Tabu Araması*, **ACO:** *Karınca Kolonisi Optimizasyonu*, **ILS:** *İteratif Yerel Arama*, **PSO:** *Parçacık Sürüsü Optimizasyonu*, **DE:** *Evrimsel Gelişim Algoritması*, **SS:** *Dağınık Arama*, **IG**: *İteratif Açgözlü Algoritma*, **FM:** *Ateş Böceği Algoritması*

# İKİNCİ BÖLÜM
# YAPAY ARI KOLONİSİ ALGORİTMASI

Karaboğa tarafından doğadaki arılardan esinlenerek 2005 yılında ortaya konulan Yapay Arı Kolonisi (Artificial Bee Colony - ABC) algoritması, popülasyon temelli bir optimizasyon yöntemidir. Geliştirildikten sonra Karaboğa ve arkadaşları tarafından öncelikle fonksiyon optimizasyonu ve yapay sinir ağlarının eğitilmesi üzerinde kullanılan yöntem, literatürde yer alan çeşitli optimizasyon yöntemleriyle kıyaslandığında görece yeni olması dolayısıyla oldukça dikkat çekicidir. ABC ile ilgili yapılan yayın taramasında çizelgeleme gibi kombinatoral optimizasyon üzerine yapılan fazla çalışmanın olmadığı gözlenmiştir.

Çalışmanın bu bölümünde, öncelikle ABC algoritmasının temelini oluşturan sosyal hayvan topluluklarında gözlemlenen sürü zekası kavramı anlatılacaktır. Karaboğa tarafından geliştirilen ABC algoritmasının işleyişi anlatıldıktan sonra, ABC algoritmasını yönelik geniş bir literatür taraması verilecektir.

## 2.1. Sürü Zekası

Sürü zekası; termitler, arılar, karıncalar, kuşlar, balık sürüleri gibi aralarında etkileşim olan böceklerin veya diğer sosyal hayvanların topluluk halindeki davranışlarını örnek alarak, problemlere çözüm getirmeyi amaçlayan bir yapay zeka tekniğidir. Arı kolonilerinin kovan etrafında dolaşarak birbirlerine bilgi aktarımları, karıncaların geçtikleri yollara kimyasal madde bırakarak diğer karıncalara bilgi aktarımları, kuş ve balık sürülerinin konum ve hızlarını ayarlayarak ilerlemeleri sürü zekasını temel teşkil eden zeki davranışlardır (Karaboğa, 2005, s. 1-2; Karaboğa ve Akay 2009, s. 112-113).

Bir sürüde iki önemli işlev vardır:

**1)** Kendi başına organize olma,
**2)** İş bölümü.

Kendi başına organize olabilme; bir sistemdeki temel birimlerin, diğer birimlerle etkileşimden aldıkları bilgileri kullanarak kendi başlarına işlev görerek sistemin bütününü etkilemeleridir. Sistemin diğer birimleri ile etkileşiminde temel komşuluk bilgilerinden faydalanılır. Bonabeau vd. (1999) kendi başına organize olabilmeyi dört özellik ile karakterize etmişlerdir:

1. Pozitif geri besleme daha uygun yapıların bulunmasını sağlayan davranışlardır. Karıncalardaki kimyasal maddenin bırakılması ve diğer karıncaların bu maddeyi takibi ile daha uygun yolların bulması, arıların dans etmeleri ile zengin nektar kaynakları hakkındaki bilgileri diğer arılara iletmeleri pozitif geri beslemeye örnek verilebilir.
2. Negatif geri besleme ise toplanan bilgilerin kararlı hale gelebilmesi için çalışır. Tüm bireylerin aynılaşarak topluluğun doyuma ulaşmasını engeller.
3. Salınımlar da yeni kaynak keşiflerinin yapılabilmesi için rastgele dolaşımlar olarak düşünülebilir.
4. Çoklu etkileşim ile de bir bireyin diğer bireye ait bilgiyi kullanabilmesi ifade edilir (Karaboğa, 2005, s.2-3; Karaboğa ve Akay 2009, s. 112-113).

İş bölümü, topluluktaki bireylerin eş zamanlı olarak farklı işleri gerçekleştiriyor olmasıdır. Özelleşmiş bireylerin bir arada çalışarak gösterdikleri performans, bu şekilde bir iş bölümüne tabi olmayan bireylerin gösterdikleri performanstan daha etkili olmaktadır ve bu özellik araştırma uzayında değişimlere cevap verebilmeyi sağlamaktadır (Karaboğa, 2005, s.3; Karaboğa ve Akay 2009, s. 112-113).

### 2.2. Yapay Arı Kolonisi Algoritması

Yapay arı koloni algoritmasında arılar bulundukları pozisyonlarla temsil edilir ve her bir arı bir yem/nektar alanına bağlı olarak çalışır. Bu pozisyonlar problem çözümünün parametreleri ile ifade edilir. Dolayısıyla araştırma uzayının boyutunu parametre sayısı belirlemektedir. Sürü içerisinde en iyi pozisyonda bulunan birey, o adımdaki, problemin çözümünü ifade eder. Arıların bulundukları konum uygunluk veya maliyet fonksiyonuyla tanımlanır. Arılar iteratif olarak bir önceki konumunu ve bir önceki adımdaki iyi çözümlerin konumlarını da dikkate alarak yeni konumlarını belirlemek yoluyla problem uzayında optimum çözümü bulmaya çalışırlar (Karaboğa, 2005, s.3-4; Karaboğa ve Akay 2009, s. 113-114).

Yapay arı kolonisi algoritmasında, bir kolonide üç grup arı bulunmaktadır: işçi arılar, gözcü arılar ve kaşif arılar. Yöntemde, koloninin yarısı işçi, yarısı da gözcü arı olarak seçilmiştir. Her bir nektar kaynağı için sadece bir işçi arı bulunmaktadır. Yani işçi arıların sayısı nektar kaynağı sayısına eşittir. Herhangi bir yiyecek kaynağını terk eden işçi arı, kaşif arı olarak çözüm uzayında rastgele araştırma yapabilmektedir. Algoritmanın temel adımları:

Başlangıç parametrelerini belirle
*Tekrarla*

- İşçi arıları kaynaklara gönder ve nektar miktarını hesapla
- Gözcü arıları kaynaklara gönder ve nektar miktarını hesapla
- Rastgele yeni kaynaklar bulmaları için kaşif arıları gönder
- O ana kadarki en iyi kaynağı hafızada tut

*Devam et* (durma kriteri sağlanana kadar)

Algoritmanın en temel halinde üç tane kontrol parametresi bulunmaktadır (Karaboğa ve Akay 2009, s. 114):

**Yem alanı**: Her bir yem alanında sadece bir işçi arı bulunmaktadır. Dolayısıyla bu parametre toplam arı sayısının yarısına eşittir.
**Limit:** İşçi arıların kaşif olması için gereken gelişme olmayan adım sayısı
**Adım sayısı:** Algoritmanın maksimum kaç adım koşacağını belirler.

Her bir çevrim üç adımdan oluşmaktadır: işçi ve gözcü arıların kaynaklara gönderilmesi, gidilen kaynakların nektar miktarının hesaplanması, kaşif arının belirlenerek yeni bir kaynağa rastgele konumlanması. Yiyecek kaynakları optimize edilmeye çalışılan problemin olası çözümlerine karşılık gelmektedir. Bir kaynağa ait nektar miktarı, o kaynakla ifade edilen çözümün kalite değerini ifade eder. İşçi arılar tarafından kaynaklara ait uygunluk değerleri hesaplandıktan sonra, gözcü arılar rulet tekerleği prensibine göre gidecekleri kaynakları belirlemektedir.

Her kolonide rastgele araştırmalar yapan kaşif arılar bulunmaktadır. Bu arılar yiyecek ararken herhangi bir önbilgi kullanmamakta, tamamen rastgele araştırma yapmaktadırlar. Dolayısıyla

arama maliyetleri düşüktür ve de buldukları kaynağın ortalama kalite değeri düşüktür. Zengin nektar kaynağına sahip keşfedilmemiş kaynakları bulma şansları da vardır. ABC algoritmasında işçi arılardan biri seçilerek kaşif arı haline gelmektedir. Bu seçme işlemi *limit* parametresine göre yapılmaktadır. Bir kaynağı ifade eden çözüm belli sayıdaki deneme ile geliştirilememişse bu kaynak terk edilir ve bu kaynağa gidip gelen işçi arı kaşif arı haline gelir. Kaynağın terk edilmesi için belirlenmiş deneme sayısı *limit* parametresi ile belirlenir (Karaboğa, 2005, s.7; Karaboğa ve Akay 2009, s.113-114).

İyi bir arama sürecinde keşif ve keşfedilenden faydalanma aynı anda gerçekleşmelidir. ABC algoritmasında gözcü ve işçi arılar keşfedilen kaynaklardan faydalanma işleminde, kaşif arılar ise keşif sürecinde yer alır (Karaboğa, 2005, s.7; Karaboğa ve Akay 2009, s. 112-114).

Parametre değerleri belirlendikten sonra algoritmada arılar başlangıç konumlarını alırlar. İlk aşamada işçi arılar rastgele araştırma uzayında dağılırlar. Bu adımda işçi arılar ilk kez bulundukları başlangıç pozisyonlarının yem miktarlarını ölçerler, o konuma ait maliyet fonksiyonu elde edilir. Bu fonksiyonun sonucu ise o arının konumunun uygunluk değeridir. Bütün işçi arılar aramayı bitirdikten sonra, kovana dönerek yiyeceğin konumunu ve nektar bilgisini kovandaki gözcü arılarla paylaşırlar. Gözcü arılar bu bilgileri değerlendirerek $p_i$ olasılığı ile bir yiyecek kaynağını seçerler. Kaynağın nektar değeri ne kadar fazla olursa, bu kaynağın bir gözcü arı tarafından seçilme olasılığı o kadar fazla olmaktadır. İşçi arılar ise hafızasında tuttuğu yiyecek konumuyla ilgili değişiklikler yaparak yeni aday konumlar belirler. Eğer bu aday konumun nektar miktarı eski konumdan daha iyiyse yeni konumu hafızasına alır (Karaboğa, 2005, s.6-8).

Gözcü arının yiyecek kaynağını seçme olasılığı:

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_i} \tag{7}$$

formüldeki $fit_i$ $i$ çözümünün uygunluk değerini, SN ise toplam yiyecek sayısını yani toplam işçi arısını ifade etmektedir.

İşçi arının hafızasındaki çözümden aday çözümler aşağıdaki denklemle üretilirler:

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \quad (8)$$

formülde $k \varepsilon \{1, 2,..., SN\}$ ve $j \varepsilon \{1, 2,..., D\}$ rastgele seçilen sayılardır. $D$ optimizasyon parametre sayısıdır. Her ne kadar rastsal da seçilse $k$ $i$'den farklı olmak zorundadır. $\Phi_{ij}$ [-1,1] aralığındaki rastgele bir sayıdır, $x_{ij}$ konumu etrafında rastgele bir komşu yiyecek kaynağının oluşmasını kontrol eder ve iki yiyecek kaynağının kıyaslanmasını ifade eder.

Terk edilen yiyecek kaynaklarının yerine kaşif arılar tarafından yenileri bulunur. Terk edilecek $x_i$ kaynağının yenisi aşağıdaki formülle bulunur:

$$x_i^j = x_{min}^j + rand(0,1)(x_{mak}^j - x_{min}^j) \quad (9)$$

Diğer sosyal yiyecek arayıcıları gibi, arılar E/T değerini yani birim zamanda yuvaya getirilen yiyecek miktarını belirten enerji fonksiyonunu maksimize etmek için çalışırlar (Karaboğa, 2005, s.6-8; Karaboğa ve Akay 2009, s. 113-114).

Bir yem alanı bünyesinde bir tane işçi arı bulunabilirken birden fazla gözcü arı olabilir. Yem alanı ilk önce işçi arı ile geliştirmeye çalışılır. Sonraki adımda bu yem alanını seçmiş gözcü arılar tarafından geliştirilmeye çalışılır. Eğer aynı adımda bir arı yem alanının konumunu değiştirmiş ise diğer arılar bu son konumdan arama işlemine devam ederler. Her adımda önce işçi arılar, sonra gözcü arılar sonra da kaşif arıya dönüşen işçi arılar arama işlemini tamamladıktan sonra en iyi uygunluğa sahip konum o adımın aday çözümünü ifade eder (Karaboğa, 2005, s.6-8).

### 2.3. Yapay Arı Kolonisi Algoritması ile İlgili Literatür Taraması

Sayısal fonksiyon optimizasyonu için geliştirilen ABC algoritması, öncelikle sürekli optimizasyon alanında yaygın bir biçimde kullanılmıştır. Algoritmanın gösterdiği başarıdan dolayı, son yıllarda farklı mühendislik alanlarında kullanıldığı görülmüştür. ABC algoritması ile yapılan çalışmalardan bazıları ve uygulama alanları aşağıda verilmiştir. İncelenen çalışmalar ve uygulama alanları ise Tablo 2.1'de yer almaktadır.

Karaboğa ve Baştürk (2007)'de, ABC algoritması çok değişkenli fonksiyonların optimizasyonu için kullanılmıştır. Algoritmanın performansı GA, PSO ve Parçacık Sürüsünden Etkilenmiş Evrimsel Gelişim algoritmalarıyla kıyaslanmıştır. Çalışmanın sonucunda ABC algoritmasının diğer algoritmalardan daha iyi sonuçlar verdiği belirtilmiştir.

Karaboğa ve Baştürk (2008)'de ise ABC algoritmasının performansı çok boyutlu sayısal problemlerde PSO, DE ve evrimsel algoritma (Evolutionary Algorithm) ile kıyaslanmış, sonuçlarının ele alınan algoritmalardan daha iyi olduğu belirtilmiştir.

Karaboğa ve Akay (2009), çalışmalarında ABC algoritmasının performansını sürekli optimizasyon için çeşitli test problemleri ile ölçmüştür. Çalışmada ABC algoritması, GA, PSO, DE ve evrimsel stratejilerle karşılaştırılmıştır. Sonuçlar ABC algoritmasının üstünlüğünü vurgulamaktadır.

Akay ve Karaboğa (2009)'da tam sayılı programlama için ABC algoritması kullanılmıştır. Algoritmanın performansı PSO ve Dal-Sınır algoritmaları ile kıyaslanmıştır. Sonuçlar ABC algoritmasının üstünlüğünü ifade etmiştir.

Akay ve Karaboğa (2010), büyük ölçekli kısıtsız optimizasyon problemleri için ABC algoritmasını dokuz farklı test problemi üzerinde deneyip sonuçları PSO ve DE ile karşılaştırmışlardır. Yapılan kıyaslamada ABC algoritmasının daha iyi sonuçlar oluşturduğu görülmüştür. Çalışmada ayrıca dört farklı kısıtlı mühendislik problemi de ele alınmış, ABC ile elde edilen sonuçların literatürdeki diğer sonuçlardan daha iyi olduğu belirtilmiştir.

Zhu ve Kwong (2010)'da sayısal optimizasyon için ABC algoritmasının geliştirilmiş bir modelini önermişlerdir. ABC algoritmasının çözüm uzayında daha iyi araştırma yapabilmesi için, modellerinde komşuluk geliştirme süreci iterasyonda bulunan en iyi yiyecek kaynağından elde edilen bilgiden yola çıkılarak yapılmaktadır. Geliştirilen modelin performansı çeşitli fonksiyonlar için klasik ABC ile karşılaştırılmıştır.

**Tablo 2.1 ABC Algoritması ile İlgili Literatür Çalışması**

| Çalışma | Uygulama Alanı |
|---|---|
| Karaboğa ve Baştürk (2007) | Sayısal optimizasyon |
| Karaboğa ve Baştürk (2008) | Sayısal optimizasyon |
| Karaboğa ve Akay (2009) | Sayısal optimizasyon |
| Akay ve Karaboğa (2009) | Tam sayılı programlama |
| Akay ve Karaboğa (2010) | Kısıtlı ve kısıtsız sayısal optimizasyon |
| Zhu ve Kwong (2010) | Sayısal optimizasyon |
| Kang vd. (2011) | Sayısal optimizasyon |
| Karaboğa ve Akay (2011) | Kısıtlı sayısal optimizasyon |
| Sönmez (2011) | Kısıtlı optimizasyon |
| Banharnsakun vd. (2011) | Sayısal optimizasyon, görüntü izdüşümü |
| Gao ve Liu (2012) | Sayısal optimizasyon |
| Li vd. (2012) | Sayısal optimizasyon |
| Karaboğa (2009) | IIR filtre tasarımı |
| Karaboğa ve Akay (2010) | PID kontrolcüsü tasarımı |
| Rao ve Patel (2011) | Soğutma kulesi tasarımı |
| Baykasoğlu vd. (2007) | Genelleştirilmiş atama problemi |
| Singh (2009) | Min. dağılan ağaç problemi |
| Sundar ve Singh (2010) | Karesel min. dağılan ağaç problemi |
| Sundar vd. (2010) | Sırt çantası problemi |
| Safarzadeh vd. (2011) | Enerji tepe noktası minimizasyonu |
| Kashan vd. (2012) | Kapasitesiz fabrika yer seçimi problemi |
| Karaboğa ve Öztürk (2009) | Yapay sinir ağları |
| Özkan vd. (2010) | Yapay sinir ağları, tahmin |
| Irani ve Nasimi (2011) | Yapay sinir ağları |
| Marinakis vd. (2009) | Kümeleme |
| Zhang vd. (2010) | Kümeleme |
| Karaboğa ve Öztürk (2010 ) | Bulanık kümeleme |
| Karaboğa ve Öztürk (2011) | Kümeleme |
| Shukran vd. (2011) | Veri madenciliği |
| Pan vd. (2011) | İşlerin bölünebilir olduğu akış tipi çizelgeleme |
| Taşgetiren vd. (2011) | Toplam akış zamanı kriteri için PFSP |
| Szeto vd. (2011) | Araç rotalama problemi |
| Liu ve Liu (2011) | PFSP |

Kang vd. (2011), Rosenbrock'un dairesel hareket metodunu ABC algoritmasıyla birleştirerek, sayısal optimizasyon problemlerinin daha hassas çözülmesini sağlamaya çalışmışlardır. Çeşitli test fonksiyonları üzerinde yapılan denemeler sonucu önerilen algoritmanın başarı oranı, doğrululuk ve yakınsama hızı gibi kriterlerde klasik ABC algoritmasından daha iyi sonuçlar verdiği belirtilmiştir. Önerilen model aynı zamanda çeşitli GA, PSO, Evrimsel Programlama ve DE algoritmaları ile karşılaştırılmış, sonuçlar önerilen modelin başarısını ortaya koymuştur.

Karaboğa ve Akay (2011)'de öncelikle kısıtsız optimizasyon için geliştirilen ABC algoritmasının kısıtlı optimizasyon problemlerine uygulanması anlatılmıştır. Önerilen algoritma literatürde yer alan 13 farklı test problemine uygulanmış ve modelin parametreleri için deney tasarımı yöntemiyle en iyi parametreler tespit edilmiştir.

Sönmez (2011)'de kafes yapı sistemlerinin optimizasyonu problemini ABC algoritması ile çözmüştür. İnşaat sektöründe karşımıza çıkan kafes sistemlerinde temel amaç, yapısal tasarım ve kurallara uyacak şekilde yapının ağırlığını minimize etmektir. Kısıtlı optimizasyon olarak tanımlanan model, farklı test problemleri için çözülmüştür. Bulunan değerler, literatürde elde edilen en iyi değerlerle kıyaslanmıştır.

Banharnsakun vd. (2011) çalışmasında yazarlar ABC algoritmasının çözüm geliştirme sürecini geliştirmeyi planlamışlardır. Önerilen algoritmada, gözcü arıların komşu çözüm üretmek için o aşamaya kadar bulunan en iyi değerden faydalanması sağlanarak, algoritmanın rastgelelik kısıtı esnetilmiştir. Ayrıca her iterasyon için, yeni komşu çözüm geliştirme sırasında arıların en iyi çözümden uzaklaşmalarını değişken mesafelerle kısıtlamışlardır. Bu kısıtlamayla arılar önce daha uzak mesafeleri araştıracak, daha sonra gittikçe araştırması yapılan yiyecek kaynağına yaklaşacaklardır. Önerilen algoritmanın performansı klasik ABC algoritması ile çeşitli test fonksiyonları aracılığı ile kıyaslanmıştır. Çalışmada ayrıca görüntü izdüşümü tekniğinde de önerilen model ile klasik ABC algoritması karşılaştırılmış, önerilen modelin daha çabuk yakınsadığı ve çok az daha iyi sonuçlar verdiği tespit edilmiştir.

Gao ve Liu (2012), çalışmalarında ABC algoritmasının keşfetme özelliğini geliştirmek için evrimsel algoritmadan esinlenmişlerdir. Geliştirilen algoritmanın performansı çeşitli test problemleri için klasik ABC algoritması ile karşılaştırılmıştır.

Li vd. (2012)'de ABC algoritmasının keşfetme ve keşfedilenden yararlanma sürecinde ortaya çıkan aksaklıklara değinilmiştir. Algoritmanın yakınsama ve çözüm uzayında araştırma hızını arttırmak için geliştirilmiş ABC (Improved ABC, I-ABC) ve hibrid ABC (PS-ABC) adında iki farklı ABC algoritması geliştirilmiştir. I-ABC algoritmasında, en iyi bulunan değer, başlangıç ağırlıkları ve hızlandırma katsayısı gibi yeni değişkenler kullanılmıştır. PS-ABC'de ise I-ABC ve $G_{best}$-ABC (Zhu ve Kwong, 2010) beraber kullanılmıştır. Çeşitli test problemleri için denen algoritmalar klasik ABC algoritmasından daha başarılı sonuçlar üretmiştir. Yakınsama hızı ve bilinen en iyi değerden ortalama sapma kriteri altında PS-ABC algoritması diğer iki ABC algoritmasına üstünlük sağlamıştır.

Karaboğa (2009)'da dijital filtre tasarımı için ABC algoritması kullanılmıştır. Geliştirilen algoritma çeşitli test problemleri için, filtre tasarımında kullanılan geleneksel bir yöntem olan LSQ algoritması ve PSO ile kıyaslanmıştır.

Karaboğa ve Akay (2010) çalışmalarında PID kontrolü tasarımı problemini ele almışlardır. Problemin çözümü için ABC algoritması, arı algoritması ve harmoni araması (Harmony Search) teknikleri kullanılmıştır. Çeşitli test problemleri ile yapılan performans değerlendirmesi sonucunda ABC algoritmasının daha başarılı olduğu ifade edilmiştir.

Rao ve Patel (2011)'de soğutma kulesi tasarımı için ABC algoritması kullanılmıştır. Su/hava kütle oranı, su kütlesi hızı ve hava kütlesi hızı tasarım değişkenleri, gereken ısı ihtiyacının yıllık maliyeti de performans kriteri olarak ele alınmıştır. Geliştirilen model, altı farklı problem üzerinde denemiş ve sonuçları optimal değerleriyle karşılaştırılmıştır.

Baykasoğlu vd. (2007), genelleştirilmiş atama problemi (Generalized Assignment Problem) için ABC algoritmasını kullanmışlardır. Belirli sayıda işi en küçük maliyetle istenen yerlere atanması olarak tanımlanabilecek problemde, ABC algoritmasına çeşitli yerel arama teknikleri eklenmiştir. Geliştirilen modelin performansı literatürden seçilen test problemlerine uygulanarak çeşitli meta-sezgisel algoritmalarla kıyaslanmıştır. ABC algoritmasının diğer algoritmalardan daha iyi sonuçlar verdiği belirtilmiştir.

Singh (2009), yaprak kısıtlı minimum dağılan ağaç (Leaf-constrained minimum spanning tree) olarak adlandırılan kısıtlı optimizasyon problemini ele almıştır. Belli sayıda düğüm noktasını en az saçılmayla birleştirmeyi ele alan bu problemde, yazar ABC algoritmasını kullanarak sonuçlarını literatürde ACO ve TS ile elde edilen en iyi sonuçlarla karşılaştırmıştır. Elde edilen sonuçlar ABC algoritmasının üstünlüğünü vermiştir.

Sundar ve Singh (2010), minimum dağılan ağaç probleminin özel bir versiyonu olan kareli minimum dağılan ağaç problemini ele almışlardır. Bu özel problemde, köşe maliyetlerinin yanı sıra sıralı çiftlerin de maliyetleri göz önüne alınmaktadır. Önerilen ABC algoritması, literatürde yer alan çeşitli GA ile kıyaslanmış ve sonuçların başarılı olduğu tespit edilmiştir.

Sundar vd. (2010) çalışmalarında 0-1 çok boyutlu sırt çantası problemini ele almışlardır. Çeşitli sezgisel operatörler ve yerel arama teknikleriyle güncellenen ABC tabanlı bir çözüm algoritması geliştirmişlerdir. Önerilen algoritma 4 farklı ACO algoritması ile kıyaslanmış ve ABC ile elde edilen sonuçların diğerlerinden daha üstün olduğu belirtilmiştir.

Safarzadeh vd. (2011) çalışmasında yazarlar bir nükleer santraldeki güç tepesi değerini ele almışlardır. Çekirdekteki hücre sayısı, güç yoğunluğu ve enerji çubuklarının yoğunlukları ve içerikleriyle orantı olan bu değer, santralin güvenliği için oldukça önemlidir. Yazarlar güç tepesi değerini veren fonksiyonun minimizasyonu için ABC algoritması kullanılmıştır. Çeşitli test problemleriyle yapılan kıyaslama sonucu önerilen modelin GA ve PSO ile başa baş sonuçlar verdiği ifade edilmiştir.

Kashan vd. (2012) çalışmasında, yazarlar ikili optimizasyon (binary optimization) sorunsalını ele alıp, bir ABC çözüm algoritması geliştirmişlerdir. Geliştirilen algoritma kapasitesiz fabrika yer seçimi problemine (Uncapacitated facility location problem) uygulanmıştır. En basit anlatımıyla, çeşitli müşteriler (talep noktaları) ve aday depo yerlerinden oluşan problemde müşteri taleplerini en düşük maliyetle karşılayacak uygun depo yerlerinin seçimi ele alınmaktadır. Algoritmada yer alan çeşitli parametrelerin en iyi değerleri tespit edildikten sonra, literatürden temin edilen 15 test problemi kullanarak algoritmalarının performansını DE ve PSO ile karşılaştırmışlardır.

Karaboğa ve Öztürk (2009)'da ABC algoritması, yapay sinir ağlarının eğitiminde kullanılmıştır. Yapay sinir ağlarının eğitiminde tahmin edilen gözlem ile gerçek gözlem arasındaki hatanın en küçüklenmesi gerektiğinden yola çıkan yazarlar, çalışmalarında hata fonksiyonunu çok değişkenli bir denklem olarak düşünüp bu denklemi ABC algoritması ile çözmüşlerdir. Elde edilen sonuçlar, literatürdeki diğer eğitme algoritmaları ile karşılaştırılmıştır.

Özkan vd. (2010), çalışmalarında günlük buharlaşma oranı tahmini için ABC ile güncellenmiş yapay sinir ağlarından faydalanmışlardır. Çalışmanın ilk bölümünde ANN-ABC modelinin doğruluğu ortalama hata kare, ortalama mutlak hata ve determinasyon katsayısı yardımıyla Levenberg-Marquardt eğitim algoritması ile kıyaslanıp, ABC'nin yapay sinir ağını eğitmede daha başarılı olduğu tespit edilmiştir. Çalışmanın devamında, geliştirilen yapay sinir ağı günlük buharlaşma katsayısının tahmini için literatürdeki bir veri setine uygulanmıştır.

Irani ve Nasimi (2011), çalışmalarında sondaj sistemlerinde karşılaşılan alt delik basıncı tahmini için yapay sinir ağlarından (YSA) faydalanmışlardır. YSA'nın eğitimi için çalışmada ABC algoritması kullanılmıştır. Yapılan çeşitli karşılaştırmalar sonucu önerilen YSA-ABC modelinin klasik YSA'ya göre daha başarılı olduğu tespit edilmiştir.

Marinakis vd. (2009), çalışmalarında kümeleme için ABC ve aç gözlü rastgele adaptif arama (Greedy randomized adaptive search procedure-GRASP) tabanlı hibrid bir algoritma geliştirmişlerdir. Önerilen algoritma çeşitli veri setleri için denemiş ve sonuçları GA, TS, ACO, PSO, GRASP ve bal arısı eşleştirme optimizasyonu ile karşılaştırılmıştır. Elde edilen sonuçlar hibrid ABC algoritmasının başarısını ortaya çıkarmıştır.

Zhang vd. (2010), kümeleme problemi için ABC algoritmasından yararlanmışlardır. Geliştirilen algoritma, çeşitli veri setleri için denemiş ve elde edilen sonuçlar GA, SA, ACO ve PSO algoritmaları ile kıyaslanmıştır. ABC algoritmasının veriyi kümelere ayırmada ve işlem süresi bazında diğer algoritmalardan daha iyi sonuçlar verdiği belirtilmiştir.

Karaboğa ve Öztürk (2010)'da çeşitli veri setleri için bulanık kümeleme üzerinde çalışılmıştır. Geliştirilen ABC algoritması, bulanık C-ortalama (Fuzzy C-means) algoritması ile

kıyaslanmıştır. Yerel optimum noktalarından kaçınmada önerilen algoritmanın bulanık C-ortalama algoritmasından daha başarılı olduğu görülmüştür.

Karaboğa ve Öztürk (2011)'de ABC algoritması kümeleme için kullanılmıştır. Literatürde yer alan çeşitli veri setleri kullanılarak ABC algoritmasının performansı PSO ve dokuz farklı kümeleme algoritması ile karşılaştırılmıştır. Çalışma sonucunda çok değişkenli kümelemede ABC algoritmasının etkinlikle kullanılabileceği tespit edilmiştir.

Shukran vd. (2011)'de ABC algoritması veri madenciliğinde sınıflandırma amacıyla kullanılmıştır. Klasik ABC algoritmasının yanı sıra yer değiştirme sezgiseliyle güncellenmiş yeni bir ABC algoritması da geliştirilmiştir. Literatürde yer alan çeşitli veri setlerinde denenen modellerin performansı klasik sınıflandırma algoritmalarıyla kıyaslanmış ve ABC algoritmasının daha başarılı sonuçlar verdiği tespit edilmiştir.

Pan vd. (2011), işlerin bölünebilir olduğu akış tipi çizelgeleme problemini ABC algoritması ile çözmüşlerdir. Modelde performans kriteri olarak toplam ağırlıklandırılmış erkenlik ve geçlik ele alınmıştır. Geliştirilen algoritmaya ayrıca yerel arama teknikleri eklenerek revize edilmiştir. Her iki algoritmada GA ve PSO ile çeşitli test problemleri aracılığıyla karşılaştırılmıştır. ABC algoritmasının daha başarılı olduğu ifade edilmiştir.

Taşgetiren vd. (2011)'de toplam akış süresinin en küçüklenmesi kriteri için PFSP problemi çözülmüştür. Geliştirilen ABC tabanlı algoritma, DE ve ILS ile kıyaslanmıştır. Kullanılan 90 test probleminden 44 tanesinde bilinen en iyi sonuçlar geliştirilmiştir.

Szeto vd. (2011), araç rotalama problemi (Vehicle routing problem) için ABC algoritmasını kullanmışlardır. Standart ABC algoritmasının yanı sıra modelin performansını geliştirmek için klasik algoritmada bazı güncellemeler yapmışlardır. Elde edilen her iki algoritmada çeşitli test problemlerinde denenmiştir.

Liu ve Liu (2011)'de ABC algoritması en geç tamamlanma kriterine sahip PFSP için kullanılmıştır. Başlangıç çözümü için NEH ve GRASP sezgisellerinden yararlanan algoritma aynı zamanda değişken komşu arama yerel arama tekniğini de içermektedir. Oluşturulan

algoritmanın performansı Reeves test problemlerinde denenmiş ve $PSO_{VNS}$ (Taşgetiren vd.2004), Hibrid GA (Wang ve Zheng, 2003) ve $PSO_{MA}$ (Liu vd., 2007) ile kıyaslanmış ortalama işlem süresi ve en iyi değerden yüzde sapma gibi kriterlerde diğer 3 alternatiften daha başarılı olduğu belirtilmiştir.

# ÜÇÜNCÜ BÖLÜM
# YAPAY ARI KOLONİSİ ALGORİTMASININ EN GEÇ TAMAMLANMA ZAMANI KRİTERİNE SAHİP PERMÜTASYON AKIŞ TİPİ ÇİZELGELEME PROBLEMİNE UYARLANMASI

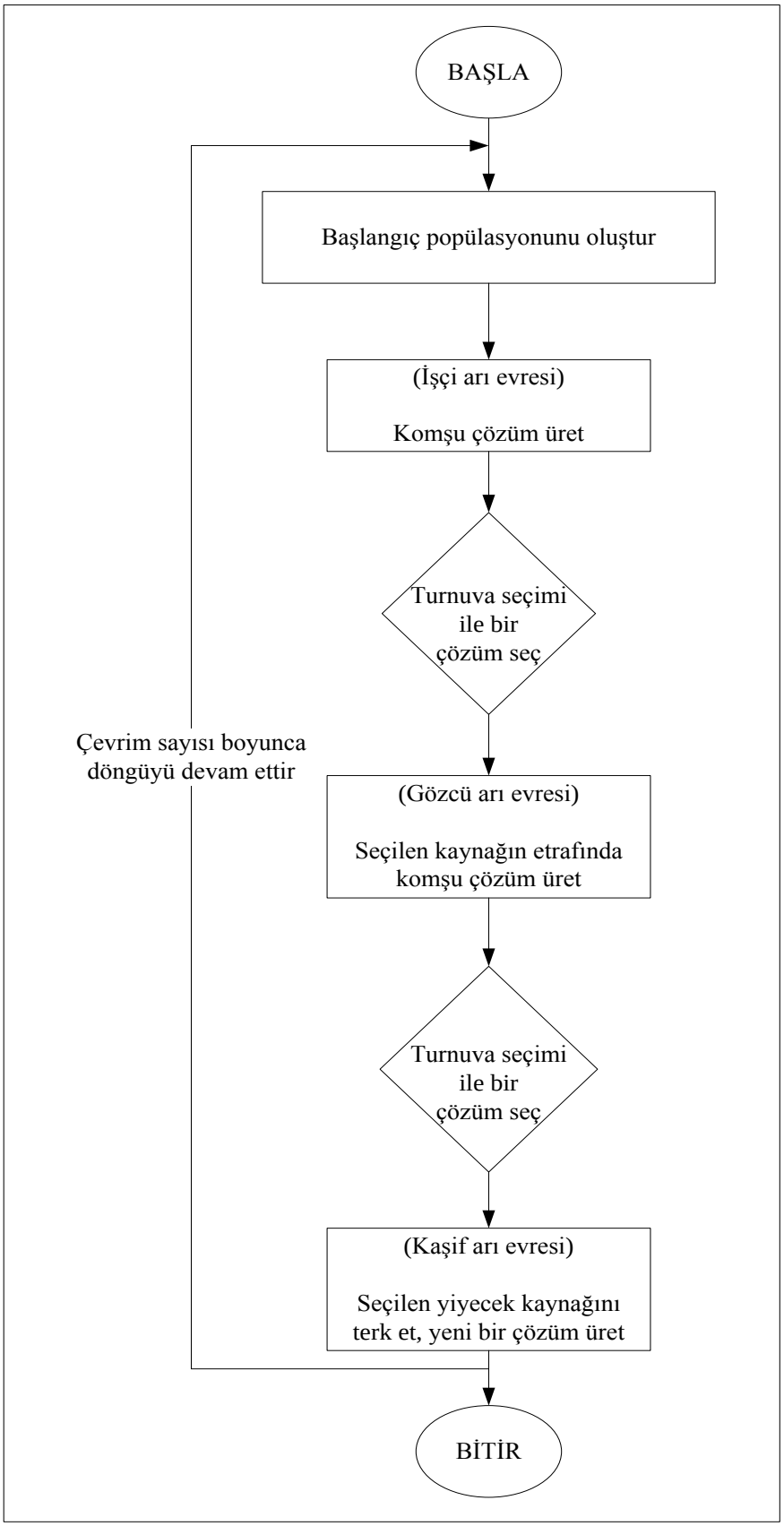
Karaboğa (2005) tarafından çok boyutlu fonksiyonların optimizasyonu için geliştirilen ABC algoritması, PFSP gibi kesikli optimizasyon sorunlarına doğrudan uygulanamaz. Bu bölümde orijinal ABC algoritmasında yapılan değişiklikler anlatılmıştır. Geliştirilen yeni algoritmanın performansı literatürde sıklıkla kullanılan Reeves, Carlier ve Taillard test problemlerine uygulanıp, farklı metasezgisel yöntemlerle elde edilen değerlerle karşılaştırılmıştır.

## 3.1. Önerilen ABC Algoritması

Bu çalışmada üç parametreli gösterim ile $F_m/Permu/C_{mak}$ olarak gösterilen en geç tamamlanma zamanı kriterine sahip permütasyon akış tipi çizelgeleme problemi kullanılmıştır. PFSP'ye uygulanacak ABC algoritmasında, yiyecek kaynakları çizelgelemesi yapılacak olan işlerin bir permütasyonuna karşılık gelmektedir. ABC algoritmasının çalışma adımları Şekil 3.1'de gösterilmiştir. Geliştirilen algoritmanın adımları ise aşağıda verilmiştir.

*Başlangıç*

ABC algoritmasında *NP* adet bireyden oluşan bir sürü kullanılmıştır. Klasik ABC algoritmasında başlangıç popülasyonu rastgele oluşturulmaktadır. Çalışmada önerilen algoritmada ise başlangıç popülasyonun kalitesini arttırmak için NEH sezgiselinden yararlanılmıştır. Literatürde en geç tamamlanma zamanı kriterli PFSP için en iyi yapıcı sezgisel algoritma olarak tanımlanan NEH sezgiseli ile üretilecek bir adet çözümün dışında kalan *(NP-1)* adet başlangıç çözümü rastgele oluşturulmuştur. Daha sonra oluşturulan *NP* adet yiyecek kaynağına ait nektar bilgileri ($C_{mak}$, en geç tamamlanma zamanı) hesaplanmıştır.




**Şekil 3.1 ABC Algoritmasının Çalışma Adımları**

*İşçi Arı Evresi*

Toplam *NP* adet işçi arı kullanılmıştır. Her bir işçi arı, bir yiyecek kaynağına giderek bu kaynağın çevresinde bir komşu çözüm oluşturmaya çalışır. Komşu çözüm geliştirmek için beş farklı yerel arama algoritmasından bir tanesi rastgele olarak seçilmiştir. Bu komşu arama algoritmaları ekleme (tekli veya ikili), yer değiştirme (tekli veya ikili) ile yok etme ve oluşturma süreçlerinden (Ruiz ve Stützle, 2007) oluşan bir yerel arama tekniğini içermektedir.

Eğer elde edilen yeni çözümün uygunluk değeri ($C_{mak}$) eskisinden daha iyiyse yeni çözüm uygunluk değeri olarak kabul edilecektir.

*Gözcü Arı Evresi*

Gözcü arılar işçi arılardan gelen bilgiler ışığında yiyecek kaynaklarının nektar bilgilerinden faydalanırlar. Çalışmada örneklem uzayında daha iyi araştırma yapabilmek için *2NP* adet gözcü arı kullanılmıştır. Gözcü arılar, kaşif arılardan gelen bilgiler ışığında gidecekleri yiyecek kaynağını seçer. Bir yiyecek kaynağını birden fazla gözcü arı seçebilir.

Yiyecek kaynağı seçimi için "turnuva seçimi" yöntemi kullanılmıştır. Çalışmada üçlü turnuva seçimi kullanılmıştır. Turnuva seçiminde rastgele seçilen üç yiyecek kaynağına ait en geç tamamlanma değerlerinden en düşük olanı, yani en iyi yiyecek kaynağı seçilir ve gözcü arı seçilen o kaynağa atanır. Gözcü arılar, yeni komşu çözüm üretmek için tekli/ikili ekleme ve tekli/ikili yer değiştirme tekniklerinden birisini rastgele olarak seçer.

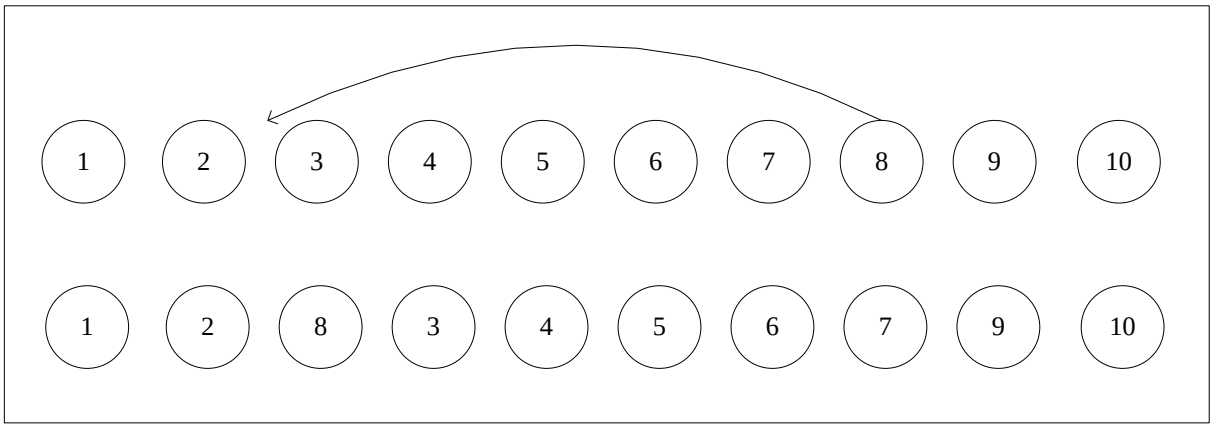
*Kaşif arı evresi*

Klasik ABC algoritmasında belirli bir iterasyon sonucu geliştirilemeyen yiyecek kaynağı terk edilir ve bu kaynağa ait işçi arı kaşif arı olarak seçilir. Bu çalışmada üçlü turnuva seçimi sonucu en kötü $C_{mak}$ değerine sahip olan yiyecek kaynağı seçilerek terk edilir. Algoritmada o ana kadar bulunan en iyi tur seçilir ve bu tura Ruiz ve Stützle (2007) tarafından geliştirilen Aç Gözlü İteratif Algoritmadan (Iterated Greedy Algorithm) esinlenilen bir yerel arama prosedürü (*insert-d*) uygulanır.

Algoritmada kullanılacak komşu arama algoritmaları kısaca aşağıda anlatılmıştır.

### 3.1.1. Ekleme Sezgiseli

Literatürde yer değiştirme tekniğiyle birlikte en çok kullanılan yöntemdir. Basit olmasının yanı sıra, oldukça etkili sonuçlar verdiği bilinmektedir. Ekleme tekniğinde, rastgele seçilen bir iş, yine rastgele seçilen başka bir işin önüne eklenir. Tekniğin temsili işleyişi Şekil 3.2'de gösterilmiştir.




**Şekil 3.2 Ekleme Yerel Arama Tekniği**

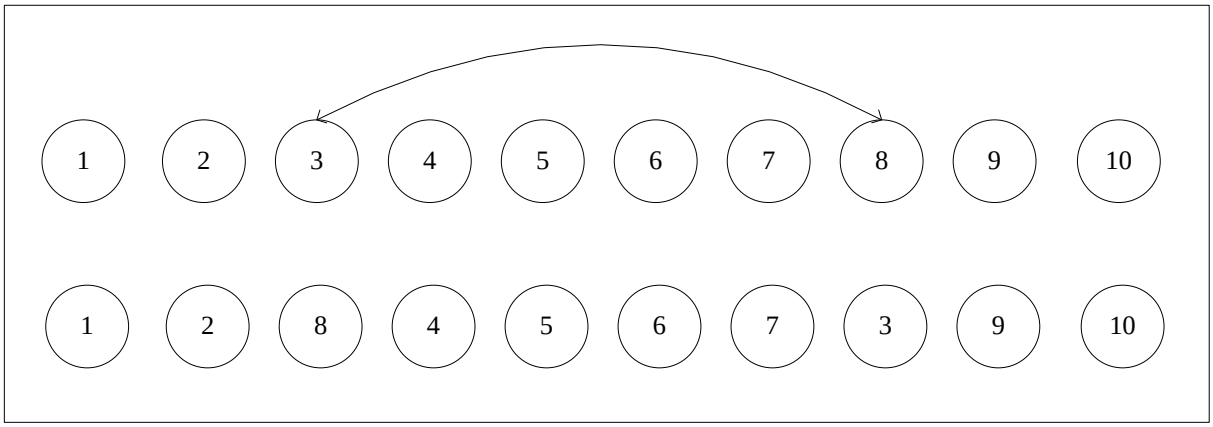
Çalışmada tekli ve ikili olmak üzere iki farklı ekleme yöntemi kullanılmıştır. Tekli eklemede sadece bir iş ötelenirken, ikili ekleme de ise rastgele seçilen iki farklı iş ötelenecektir.

### 3.1.2. Yer Değiştirme Sezgiseli

Yer değiştirme basit olarak rastgele seçilen iki işin yerlerinin değiştirilmesi sonucu yeni bir çözüm üretmektir (Şekil 3.3). Çalışmada tekli ve ikili olmak üzere iki farklı yer değiştirme yöntemi kullanılmıştır.

### 3.1.3. Yok etme – oluşturma Sezgiseli

Yok etme – oluşturma sezgiseli iki aşamalı bir yöntemdir. İlk olarak yok etme süreci gerçekleşir ve seçilen bir $\pi$ permütasyondan $d$ adet iş rastgele seçilerek çıkartılır. Böylelikle $n - d$ adet işten oluşan yeni bir $\pi_D$ permütasyonu elde edilmiş olur. Bu aşamadan sonra, oluşturma süreci başlar. Bu süreçte, çıkartılan $d$ işinden her biri sırasıyla $\pi_D$ permütasyonuna eklenerek mümkün olan tüm yeni olasılıklar ($n - d + 1$ yeni pozisyon) elde edilip $C_{mak}$ değeri en düşük olan yeni çizelge seçilir. Bu oluşturma süreci $d$ işin her biri için tekrarlanır.




**Şekil 3.3 Yer Değiştirme Yerel Arama Tekniği**

### 3.1.4. Insert-d Sezgiseli

Geliştirilen *insert-d* sezgiseli, üç adımdan oluşmaktadır. İlk adımda modifiye edilmiş ekleme (*insertMod*) sezgiseli o aşamaya kadar bulunan en iyi çözüme uygulanır. *InsertMod* tekniğinde, rastgele olarak seçilen bir iş, mümkün olan bütün pozisyonlara atanarak yeni çözümler oluşturulur. Elde edilen aday çözümlerden en iyi $C_{mak}$ değerine sahip olan yeni çözüm seçilir ve buna yok etme – oluşturma sezgiseli uygulanır. Elde edilen yeni çözümün $C_{mak}$ değeri ilk çözüm değerinden daha iyiyse yeni çözüm kabul edilir. Çeşitlendirme sürecini zenginleştirmek ve algoritmanın yerel minimum noktasında sıkışmasını önlemek için *(0,1)* aralığında bir rastgele sayı seçilir ve seçilen bu sayı *0,10*'dan küçükse daha kötü $C_{mak}$ değerine sahip olan çözüm seçilir. Üçüncü aşamada ise seçilen çözüme tekrar i*nsertMod* sezgiseli uygulanır. Algoritmanın işleyişi aşağıda gösterilmiştir.

Yukarıda anlatılan döngü belirli bir iterasyon sayısı boyunca devam eder ve her bir iterasyondaki en iyi $C_{mak}$ değeri ve bu değere karşılık gelen tur bilgisi sistemde saklanır.

$Başla$

$i = 1$

$\pi = \pi_*$

$\pi_1 = InsertMod(\pi_*)$

$\pi_2 = Yok\ etme - oluşturma(\pi_1)$

$Eğer\ \left(f(\pi_2) < f(\pi)\right)$

$\pi = \pi_2$

$Eğer\ rand < 0.1$

$\pi = \pi_2$

$\pi_3 = InsertMod(\pi)$

$Eğer\ \left(f(\pi_3) < f(\pi)\right)$

$\pi = \pi_3$

$i = i + 1$

$i < çevrim\ olduğu\ sürece$

$return\ \pi$

$Bitir$

**Şekil 3.4 Insert-d Sezgiseli Algoritması (Matlab Ortamında Yazılmıştır)**

Geliştirilen algoritmanın adımları aşağıda belirtilmiştir.

**1.** *NP* ve toplam çevrim sayısı (*iter*) parametrelerini belirle

**2.** Başlangıç popülasyonunu oluştur

(*NP-1*) tane rastgele yiyecek kaynağı oluştur, son yiyecek kaynağını NEH sezgiseli ile oluştur.

Yiyecek kaynaklarının uygunluk değerlerini ($C_{mak}$) hesapla.

*i=1,2,..., iter* için aşağıdaki adımları tekrarla

**3.** *İşçi arı evresi*

*i=1,2,…,NP* için aşağıdaki adımları tekrarla
İşçi arıları rastgele yiyecek kaynaklarına gönder
*i*. işçi arı için komşu bir yiyecek kaynağı oluştur
Eğer yeni yiyecek kaynağının $C_{mak}$ değeri eskisinden daha iyiyse, yeni kaynağı seç ve popülasyona ekle

**4.** *Gözcü arı evresi*

*i=1,2,…, 2xNP* için aşağıdaki adımları tekrarla
3 büyüklüğündeki turnuva seçimi yöntemi ile rastgele bir yiyecek kaynağını seç
*i*. gözcü arı için komşu bir yiyecek kaynağı oluştur
Eğer yeni yiyecek kaynağının $C_{mak}$ değeri eskisinden daha iyiyse, yeni kaynağı seç ve popülasyona ekle

**5.** *Kaşif arı evresi*

3 büyüklüğündeki turnuva seçimi ile bir yiyecek kaynağı seç
*insert-d* sezgiseli ile komşu yiyecek kaynağını oluştur ve seçimini yap

**6.** Çevrimdeki en iyi çözümü ve ona karşılık gelen tur bilgisini kaydet.

### 3.2. Önerilen Algoritmanın Uygulanması

Geliştirilen algoritma, Matlab ortamında yazılan uygulamasıyla, literatürdeki çalışmalarda kullanılan Reeves, Carlier ve Taillard test problemlerine uygulanmıştır.

#### 3.2.1. Parametre Tahmini

Önerilen ABC algoritmasında kullanıcı tarafından belirlenen iki genel parametre vardır. Bunlar:

1. Kolonideki arı sayısı,
2. Turnuva seçiminin büyüklüğü.

Literatürdeki benzer çalışmalar incelendiğinde turnuva seçiminin büyüklüğü için genellikle 2 veya 3 değeri kullanılmaktadır. ABC algoritmasını kullanan çalışmalarda ise arı sayısı büyüklüğü için herhangi bir en iyi değer verilmemekte, ele alınan problem türü için farklı değerler kullanılmaktadır. Her iki parametre de deneme yanılma yöntemi ile belirlenmektedir. Bu çalışmada da rastsal davranmamak için öncelikle algoritmanın genel performansını en iyileyecek parametrelerin bulunması gerekmektedir. Parametre tahmini için yapılacak deney tasarımında kullanılacak değerler Tablo 3.1'de verilmiştir.

**Tablo 3.1 Deney Tasarımında Kullanılan Değerler**

| Turnuva Seçiminin Büyüklüğü | Arı Sayısı |
|---|---|
| 2 | 10 |
| 3 | 15 |
| 4 | 20 |
| 5 | 25 |
| | 30 |

Geliştirilen deney tasarımında Tablo 3.1'de verilen değerlerin bütün olası durumlarını içeren 20 farklı ikili durum kullanılmış ve Taillard test problemleri üzerinde denenmiştir. Deney tasarımında Taillard problemlerinden 500 iş grubu dışındaki 11 problem grubunun her birinden rastgele sayılar yardımıyla 1'er adet problem seçilmiştir. Seçilen problemler her bir parametre çifti için 10'ar kez çalıştırılmış (11 * 10 * 20 = 2200 çalışma) ve her problem grubu için elde edilen ortalama değerler hesaplanmıştır. Kullanılan parametre çiftleri arasında seçim yapmak için parametrik olmayan istatistiksel testlerden Friedman testi kullanılmıştır.

Friedman testi (Friedman iki-yönlü sıra bazlı varyans testi) iki yönlü varyans analizinin parametrik olmayan uygulamasıdır. Sıfır hipotezi bütün algoritmaların aynı olduğunu temel alır, dolayısıyla bu hipotezin reddi ile algoritma performansları arasında bir farkın olduğu ortaya çıkmaktadır. Friedman testinde her veri seti için ayrı ayrı en iyi performans gösteren algoritma 1. sırada olacak şekilde sıraya dizilmektedir (Garcia vd., 2009; Luengo vd., 2009; Derrac vd., 2011).

$$H_0: \mu_{(2,10)} = \mu_{(2,15)} = \cdots = \mu_{(5,30)} \quad (10)$$

$$H_1: \mu_{(2,10)} \neq \mu_{(2,15)} \neq \cdots \neq \mu_{(5,30)} \quad (11)$$

Friedman test sonuçları Tablo 3.2'de verilmiştir.

**Tablo 3.2 Friedman Testi ile Elde Edilen Ortalama Sıra Değerleri. Hesaplanan Test İstatistiği ve p-değeri Ayrıca Verilmiştir.**

| Parametre Çifti | Friedman Test Değeri |
|---|---|
| (2, 10) | 9,80 |
| (3, 10) | 14,25 |
| (4, 10) | 12,50 |
| (5, 10) | 15,40 |
| (2, 15) | 12,05 |
| (3, 15) | 9,30 |
| (4, 15) | 11,35 |
| (5, 15) | 14,25 |
| (2, 20) | 9,50 |
| (3, 20) | 11,55 |
| (4, 20) | 8,15 |
| (5, 20) | 10,25 |
| (2, 25) | 10,50 |
| (3, 25) | 7,40 |
| (4, 25) | 7,85 |
| (5, 25) | 12,95 |
| (2, 30) | 7,65 |
| (3, 30) | 9,05 |
| (4, 30) | 7,70 |
| (5, 30) | 8,55 |
| İstatistik | 35,841 |
| *p*-değeri | 0,011 |

Friedman test sonuçlarına göre (3, 25) parametre çifti (turnuva seçimi büyüklüğü:3, kolonideki arı sayısı: 25) 7,40 ortalama sıra değeri ile diğer parametre çiftlerine göre daha iyi bir performans göstermiştir. Hesaplanan *p*-değeri, belirlenen parametre çiftleri arasında anlamlı bir farklılık olduğunu vurgulamaktadır.

Algoritmanın doğasındaki rastgelelikten dolayı farklı problemler ve farklı parametreler için yapılacak denemelerde bulunacak değerler de birbirinden farklı olacaktır. Deney tasarımında sadece algoritmanın ele alınacak tüm problem grupları için geçerli olacak parametre grubu belirlenmiştir. Çalışmanın ilerleyen kısımlarında (3, 25) değerleri kullanılmıştır.

### 3.2.2. Reeves ve Carlier Test Problemleri

Algoritmanın performansı öncelikle Reeves ve Carlier test problemleri üzerinde denenmiştir. Çalışmada sekiz adet Carlier ve 21 adet Reeves test problemi kullanılmıştır. Carlier problemlerinde iş sayısı 8 ve 13 arasında, makine sayısı ise dört ve dokuz arasında değişmektedir. Reeves veri setinde ise işler 20, 30, 50 ve 75 olarak değişirken makine sayısı 5, 10, 15 veya 20 değerini almaktadır.

Uygulamada, her bir test problemi için algoritma yirmi defa çalıştırılmıştır. Bulunan en iyi değerler karşılaştırma tablolarında verilmiştir. Performans kıyaslaması için, her bir test problemi için bilinen en iyi çözüm değerinden sapma (BRE %) ve elde edilen on değer için ortalama sapma (ARE %) değerleri hesaplanmıştır. BRE değeri aşağıdaki formülle hesaplanmıştır:

$$BRE\ \% = \frac{f_{ABC} - f_{min}}{f_{min}} .100 \tag{12}$$

$$ARE\ \% = \frac{\sum_{i=1}^{n} f_{ABC}}{n} \tag{13}$$

Algoritmada kullanılan değişkenlerin aldığı değerler Tablo 3.3'te verilmiştir.

**Tablo 3.3 ABC Algoritmasında Kullanılan Değişkenlerin Aldığı Değerler**

| Değişken | Değer |
|---|---|
| NP | 25 |
| Turnuva seçiminin büyüklüğü | 3 |
| İter | 250 |
| Kaşif arı sayısı | 5 |

Algoritmanın sonuçları Tang vd. (2010), Liu vd. (2011), Liu ve Liu (2011) ile Liu vd. (2007) çalışmalarından alınan sonuçlarla kıyaslanmıştır. Karşılaştırmalar yapılırken yazarlar tarafından geliştirilen algoritmaların yanı sıra, adı geçen eserlerde yazarlar tarafından karşılaştırma amaçlı kullanılan diğer algoritma sonuçlarına da yer verilmiştir. Kullanılan algoritmalar ve kısa tanımları Tablo 3.4'te verilmiştir.

Algoritma ile elde edilen değerler ve performansının karşılaştırılması Tablo 3.5 ve 3.6'da verilmiştir.

Tablo 3.5'te Carlier test problemlerinden elde edilen değerler gösterilmektedir. Geliştirilen algoritma 8 problem içinde literatürde bilinen en iyi değerleri bulabilmiştir. Karşılaştırma için kullanılan algoritmalardan $PSO_{MA}$, $PSO_{VNS}$, HGA, EM ve PSO-EDA_PI algoritmaları benzer bir başarı gösterip 8 problem içinde en iyi değerleri bulmuşlardır.

**Tablo 3.4 ABC Algoritmasıyla Performans Karşılaştırmaları Yapılacak Metasezgisel Yöntemler**

| Çalışma | Geliştirilen Algoritma | Temel Özellik | Karşılaştırma İçin Kullanılan Algoritmalar | Temel Özellik |
|---|---|---|---|---|
| Tang vd. (2010) | PGA | Hibrid GA | HGIA | Hibrid GA |
| | | | $HGA_1$ | Hibrid GA |
| Liu vd. (2007) | $PSO_{MA}$ | Hibrid PSO | $PSO_{VNS}$ | Yerel arama ile güçlendirilmiş PSO |
| | | | SGA | Hibrid GA |
| | | | SGA + NEH | NEH ile güçlendirilmiş hibrid GA |
| | | | $HGA_2$ | Hibrid GA |
| Liu vd. (2011) | PSO-EDA_PI | Dağılım tahmini algoritması ve yerel arama tekniği ile güçlendirilmiş PSO | EM | Elektro-manyetik algoritma |
| | | | $PSO_{MA}$ | Hibrid PSO |
| Liu ve Liu (2011) | HDABC | ABC algoritması | $PSO_{VNS}$ | Yerel arama ile güçlendirilmiş PSO |
| | | | $PSO_{MA}$ | Hibrid PSO |
| | | | $HGA_2$ | Hibrid GA |

Tablo 3.6'da ise Reeves test problemlerine ait değerler yer almaktadır. Geliştirilen algoritmanın performansı 11 farklı algoritma ile kıyaslanmıştır. Ele alınan 21 problemden 13 tanesinde geliştirilen algoritma en iyi sonucu bulmuştur. Bu değerden yola çıkarak algoritmanın

başarı oranının yaklaşık % 61,9 olduğu söylenebilir. Bu başarı oranı, ele alınan diğer algoritmalarda ise 0 ile % 47,6 arasında değişmektedir. En iyi değere ulaşmada önerilen algoritmanın diğerlerinden daha üstün olduğu görülmektedir.

Geliştirilen ABC algoritması Reeves23 test problemi için yeni bir en iyi sonuç bulmuştur.

Bilinen en iyi değerden yüzde sapma kriteri altında algoritmaları incelersek (Tablo 3.6), 14 problemde ABC algoritmasının en düşük sapmayı verdiği, ortalama yüzde hata oranı kriterinde ise 8 problemde algoritmanın daha başarılı olduğu görülmektedir. Bu iki kriter altında da önerilen algoritma ele alınan tüm algoritmalardan daha üstündür.

Tablo 3.6'daki sonuçlar kullanılarak ABC algoritması diğer bir ABC tabanlı algoritma olan HDABC ile kıyaslanırsa, HDABC algoritmasının sadece yedi problemde en iyi değeri bulabildiği, ortalama hata yüzdesi kriteri altında ise ele alınan 21 problem için de ABC algoritmasının HDABC'den daha başarılı olduğu görülmektedir.

Tablo 3.5 ve 3.6 doğrultusunda Reeves ve Carlier test problemleri için mevcut ABC algoritmasının en iyi sonuçları verdiği tespit edilmiştir.

#### 3.2.2.1. ABC Algoritmasının Performansının Değerlendirilmesi

Bu bölümde, geliştirilen ABC algoritmasının performansı parametrik olmayan istatistiksel testler kullanılarak Tablo 3.6'te yer alan diğer algoritmalarla karşılaştırılacaktır.

Parametrik olmayan testler, aynı tekrar sayısına sahip veri setinden elde edilen ortalama değerler ile çalışmaktadır. Bu sayede herhangi bir ön şart veya kısıtlama olmadan olasılıklı ve olasılıklı olmayan yöntemler kullanılabilmektedir. Anlamlı bir karşılaştırma yapabilmek için kullanılacak değerlerin aynı süreçlerle elde edilmesi gerekmektedir (her bir problem ve algoritma için aynı tekrar sayısı). Bu testler ile her bir problem için ortalama değerleri bilinen farklı algoritmaları karşılaştırmak mümkündür (Garcia vd., 2009; Luengo vd., 2009). Çalışmadaki tüm istatistiksel analizler SPSS paket programı aracılığı ile yapılacaktır.

Tablo 3.6'da yer alan algoritmalardan PSO-EDA_PI dışındakiler 20 tekrar ile elde edilen değerleri vermektedir. Dolayısıyla anlamlı bir karşılaştırma yapabilmek için PSO-EDA_PI algoritması istatistiksel analizlerde kullanılmayacaktır.

#### 3.2.2.1.1. İkili Karşılaştırmalar

İkili karşılaştırmalar, deney çalışması çerçevesinde bir araştırmacının yapabileceği en basit istatistiksel analizdir. Ortak bir problem kümesine iki algoritmanın performansını birbirleriyle karşılaştırmak için uygulanırlar. Bu bölümde öncelikle hızlı ve kolay olduğu kadar ikili karşılaştırma için güçlü bir öncü gösterge olabilecek işaret testi, daha sonra ise ikili istatistiksel karşılaştırma yapmak için basit fakat güvenli ve güçlü bir test olan Wilcoxon işaretli sıra testi kullanılacaktır (Derrac vd., 2011).

ABC algoritması ile diğer algoritmalarla yapılan ikili karşılaştırmalar yardımıyla önerilen algoritmanın performansı incelenecektir.

Algoritmaların genel performansını kıyaslamanın popüler bir yöntemi, algoritmanın üstün geldiği vaka sayısını saymaktır. Bu sayı binom dağılımına uygun olarak dağılmaktadır (Derrac vd., 2011).

Tablo 3.7'de $\alpha = 0,05$ ve $\alpha = 0,10$ güven seviyeleri için ulaşılması gereken kritik galibiyet değerleri yer almaktadır (Derrac vd., 2011). Bu çalışmada vaka sayısı 21'dir. Reeves problemleri için bir algoritma eğer o satırdaki vaka sayısından daha fazla sayıda galibiyeti varsa, anlamlı bir biçimde diğerinden daha iyidir.

**Tablo 3.5 Carlier Test Problemleri İçin Performans Karşılaştırma Tablosu**

| **Problem** | **BKS** | **NEH** | **ABC** | | **$PSO_{MA}$** | **$PSO_{VNS}$** | **SGA** | **SGA+NEH** | **$HGA_2$** | **EM** | **PSO-EDA_PI** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (iş x makine) | | | BD | BRE % | BRE % | BRE % | BRE % | BRE % | BRE % | BRE % | BRE % |
| Car1 – 11x5 | 7038 | 7038 | **7038** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| Car2 – 13x4 | 7166 | 7376 | **7166** | **0** | **0** | **0** | **0** | 2,93 | **0** | **0** | **0** |
| Car3 – 12x5 | 7312 | 7399 | **7312** | **0** | **0** | **0** | 1,19 | 0,82 | **0** | **0** | **0** |
| Car4 – 12x4 | 8003 | 8003 | **8003** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| Car5 – 10x6 | 7720 | 7835 | **7720** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| Car6 – 8x9 | 8505 | 8773 | **8505** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| Car7 – 7x7 | 6590 | 6590 | **6590** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| Car8 – 8x8 | 8366 | 8564 | **8366** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**Not:** *BKS (Bilinen en iyi değer), ABC (yazar tarafından geliştirilen algoritma), BD (ABC algoritması ile bulunan en iyi değer). Bulunan en iyi değerler* ***koyu*** *yazılmıştır.*

**Tablo 3.6 Reeves Test Problemleri İçin Performans Karşılaştırma Tablosu**

| **Problem** | **BKS** | **NEH** | **ABC** | | | **$PSO_{MA}$** | | **$PSO_{VNS}$** | | **SGA** | | **SGA+NEH** | | **$HGA_2$** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (iş x makine) | | | BD | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% |
| Rec01 – 20x5 | 1247 | 1303 | **1247** | **0** | 0,120 | **0** | 0,144 | 0,160 | 0,168 | 2,81 | 6,96 | 2,25 | 6,13 | **0** | 0,14 |
| Rec03 – 20x5 | 1109 | 1132 | **1109** | **0** | **0,004** | **0** | 0,189 | **0** | 0,158 | 1,89 | 4,45 | 1,26 | 4,27 | **0** | 0,09 |
| Rec05 – 20x5 | 1242 | 1281 | **1242** | **0** | **0,144** | 0,242 | 0,249 | 0,242 | 0,249 | 1,93 | 3,82 | 2,33 | 2,90 | **0** | 0,29 |
| Rec07 – 20x10 | 1566 | 1626 | **1566** | **0** | 0,057 | **0** | 0,986 | 0,702 | 1,095 | 1,15 | 5,31 | 3,38 | 5,27 | **0** | 0,69 |
| Rec09 – 20x10 | 1537 | 1583 | **1537** | **0** | **0** | **0** | 0,621 | **0** | 0,651 | 3,12 | 4,73 | 0,39 | 2,13 | **0** | 0,64 |
| Rec11 – 20x10 | 1431 | 1550 | **1431** | **0** | **0** | **0** | 0,129 | 0,071 | 1,153 | 3,91 | 7,39 | 1,19 | 3,66 | **0** | 1,10 |
| Rec13 – 20x15 | 1930 | 2002 | **1930** | **0** | **0,140** | 0,259 | 0,893 | 1,036 | 1,790 | 3,68 | 5,97 | 1,92 | 4,41 | 0,36 | 1,68 |
| Rec15 – 20x15 | 1950 | 2013 | **1950** | **0** | 0,118 | 0,051 | 0,628 | 0,769 | 1,487 | 2,21 | 4,29 | 2,87 | 4,02 | 0,56 | 1,12 |
| Rec17 – 20x15 | 1902 | 2019 | **1902** | **0** | **0,092** | **0** | 1,330 | 0,999 | 2,453 | 3,15 | 6,08 | 2,16 | 4,02 | 0,95 | 2,32 |
| Rec19 – 30x10 | 2093 | 2185 | 2099 | **0,286** | 0,655 | 0,430 | 1,313 | 1,529 | 2,099 | 4,01 | 6,07 | 2,05 | 4,35 | 0,62 | 1,32 |
| Rec21 – 30x10 | 2017 | 2131 | 2026 | 0,446 | 1,423 | 1,437 | 1,596 | 1,487 | 1,671 | 3,42 | 6,07 | 3,52 | 3,58 | 1,44 | 1,57 |
| Rec23 – 30x10 | 2011 | 2155 | ***2009*** [*] | - | **0,208** | 0,596 | 1,310 | 1,343 | 2,106 | 3,83 | 7,46 | 3,63 | 5,12 | 0,40 | 0,87 |
| Rec25 – 30x15 | 2513 | 2644 | **2513** | **0** | 0,902 | 0,835 | 2,085 | 2,388 | 3,166 | 4,42 | 7,20 | 3,14 | 4,89 | 1,27 | 2,54 |
| Rec27 – 30x15 | 2373 | 2498 | 2379 | 0,253 | 0,939 | 1,348 | 1,605 | 1,728 | 2,463 | 4,93 | 6,85 | 3,16 | 5,21 | 1,10 | 1,83 |
| Rec29 – 30x15 | 2287 | 2391 | 2297 | 0,437 | **0,839** | 1,442 | 1,888 | 1,968 | 3,109 | 6,21 | 8,48 | 3,32 | 4,93 | 1,40 | 2,70 |
| Rec31 – 50x10 | 3045 | 3173 | 3056 | 0,360 | 1,667 | 1,510 | 2,254 | 2,594 | 3,232 | 6,17 | 8,02 | 5,94 | 6,66 | 0,43 | **1,34** |
| Rec33 – 50x10 | 3114 | 3241 | **3114** | **0** | 0,435 | **0** | 0,645 | 0,835 | 1,007 | 3,08 | 5,12 | 2,70 | 3,38 | **0** | 0,78 |
| Rec35 – 50x10 | 3277 | 3313 | **3277** | **0** | **0** | **0** | **0** | **0** | 0,038 | 1,46 | 3,30 | 1,89 | 2,58 | **0** | **0** |
| Rec37 – 75x20 | 4951 | 5227 | 5068 | 2,363 | 3,288 | 2,101 | 3,537 | 4,383 | 4,949 | 7,89 | 10,07 | 7,14 | 7,94 | 3,75 | 4,90 |
| Rec39 – 75x20 | 5087 | 5308 | 5140 | 1,042 | 1,689 | 1,553 | 2,426 | 2,850 | 3,371 | 7,30 | 8,51 | 6,25 | 7,09 | 2,20 | 2,79 |
| Rec41 – 75x20 | 4960 | 5292 | 5067 | 2,157 | 2,157 | 2,641 | 3,684 | 4,173 | 4,867 | 8,51 | 10,03 | 7,49 | 8,47 | 3,64 | 4,92 |

**Not:** *BKS (Bilinen en iyi değer), ABC (yazar tarafından geliştirilen algoritma), BD (Belirtilen algoritma ile bulunan en iyi değer). Bulunan en iyi değerler* ***koyu*** *yazılmıştır.* [*] *Bilinen en iyi sonuçtan daha iyi bir değer. PSO-EDA_PI algoritmasına ait değerler 10 tekrar ile diğer algoritmaların değerleri 20 tekrar ile bulunmuştur.*

**Tablo 3.6 Reeves Test Problemleri İçin Performans Karşılaştırma Tablosu (… devam)**

| **Problem** | **BKS** | **NEH** | **ABC** | | | **HDABC** | | **PSO-EDA_PI** | | **EM** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| (iş x makine) | | | BD | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% | BRE% | ARE% |
| Rec01 – 20x5 | 1247 | 1303 | **1247** | **0** | 0,120 | **0** | 0,128 | **0** | **0,096** | 0,16 | 4,41 |
| Rec03 – 20x5 | 1109 | 1132 | **1109** | **0** | **0,004** | **0** | 0,041 | **0** | 0,036 | 0,18 | 1,98 |
| Rec05 – 20x5 | 1242 | 1281 | **1242** | **0** | **0,144** | **0** | 0,145 | 0,242 | 0,242 | 0,24 | 2,01 |
| Rec07 – 20x10 | 1566 | 1626 | **1566** | **0** | 0,057 | **0** | 0,934 | **0** | **0** | 1,15 | 3,70 |
| Rec09 – 20x10 | 1537 | 1583 | **1537** | **0** | **0** | **0** | 0,059 | **0** | 0,202 | 0,65 | 3,97 |
| Rec11 – 20x10 | 1431 | 1550 | **1431** | **0** | **0** | **0** | 0,073 | **0** | 0,126 | 0,98 | 4,05 |
| Rec13 – 20x15 | 1930 | 2002 | **1930** | **0** | **0,140** | 0,104 | 0,474 | 0,104 | 0,223 | 2,23 | 4,87 |
| Rec15 – 20x15 | 1950 | 2013 | **1950** | **0** | 0,118 | 0,667 | 0,956 | **0** | 0,303 | 1,64 | 3,79 |
| Rec17 – 20x15 | 1902 | 2019 | **1902** | **0** | **0,092** | 0,789 | 1,669 | **0** | 0,289 | 2,63 | 5,57 |
| Rec19 – 30x10 | 2093 | 2185 | 2099 | **0,286** | 0,655 | 0,382 | 1,247 | 0,287 | **0,612** | 2,34 | 5,97 |
| Rec21 – 30x10 | 2017 | 2131 | 2026 | 0,446 | 1,423 | 1,437 | 1,448 | 1,140 | 1,408 | 1,64 | 3,22 |
| Rec23 – 30x10 | 2011 | 2155 | ***2009**** | - | **0,208** | 0,447 | 1,235 | 0,398 | 0,597 | 2,49 | 5,97 |
| Rec25 – 30x15 | 2513 | 2644 | **2513** | **0** | 0,902 | 1,154 | 2,085 | 0,279 | 1,894 | 2,55 | 5,93 |
| Rec27 – 30x15 | 2373 | 2498 | 2379 | 0,253 | 0,939 | 0,801 | 1,483 | 0,969 | 1,584 | 2,49 | 5,90 |
| Rec29 – 30x15 | 2287 | 2391 | 2297 | 0,437 | **0,839** | 1,006 | 1,650 | **0,350** | 1,045 | 3,15 | 7,08 |
| Rec31 – 50x10 | 3045 | 3173 | 3056 | 0,360 | 1,667 | 1,839 | 2,274 | **0,263** | **0,430** | 4,01 | 6,40 |
| Rec33 – 50x10 | 3114 | 3241 | **3114** | **0** | 0,435 | 0,385 | 0,759 | **0** | 0,469 | 0,96 | 3,76 |
| Rec35 – 50x10 | 3277 | 3313 | **3277** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | 2,35 |
| Rec37 – 75x20 | 4951 | 5227 | 5068 | 2,363 | 3,288 | 3,090 | 3,700 | **1,838** | **2,725** | 5,43 | 8,22 |
| Rec39 – 75x20 | 5087 | 5308 | 5140 | 1,042 | 1,689 | 1,828 | 2,349 | **0,924** | **1,409** | 3,97 | 7,23 |
| Rec41 – 75x20 | 4960 | 5292 | 5067 | 2,157 | 2,157 | 2,943 | 3,773 | **1,815** | **2,506** | 6,21 | 8,43 |

**Not:** *BKS (Bilinen en iyi değer), ABC (yazar tarafından geliştirilen algoritma), BD (Belirtilen algoritma ile bulunan en iyi değer).*
*Bulunan en iyi değerler* **koyu** *yazılmıştır.* * *Bilinen en iyi sonuçtan daha iyi bir değer. PSO-EDA_PI algoritmasına ait değerler 10 tekrar ile diğer algoritmaların değerleri 20 tekrar ile bulunmuştur.*

**Tablo 3.6 Reeves Test Problemleri İçin Performans Karşılaştırma Tablosu (… devam)**

| **Problem** | **BKS** | **NEH** | **ABC** | | | **PGA** | | | **HGIA** | | | **$HGA_1$** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (iş x makine) | | | BD | BRE% | ARE% | BD | BRE% | ARE% | BD | BRE% | ARE% | BD | BRE% | ARE% |
| Rec01 – 20x5 | 1247 | 1303 | **1247** | **0** | 0,120 | **1247** | **0** | 0,14 | **1247** | **0** | 0,58 | **1247** | **0** | 1,36 |
| Rec03 – 20x5 | 1109 | 1132 | **1109** | **0** | **0,004** | **1109** | **0** | 0,06 | **1109** | **0** | 0,15 | **1109** | **0** | 1,35 |
| Rec05 – 20x5 | 1242 | 1281 | **1242** | **0** | **0,144** | **1245** | 0,24 | 0,24 | 1245 | 0,24 | 0,26 | **1242** | **0** | 0,40 |
| Rec07 – 20x10 | 1566 | 1626 | **1566** | **0** | 0,057 | **1566** | **0** | 0,36 | **1566** | **0** | 0,68 | **1566** | **0** | 1,91 |
| Rec09 – 20x10 | 1537 | 1583 | **1537** | **0** | **0** | **1537** | **0** | 0,40 | **1537** | **0** | 0,56 | 1549 | 0,78 | 2,26 |
| Rec11 – 20x10 | 1431 | 1550 | **1431** | **0** | **0** | **1431** | **0** | 0,37 | **1431** | **0** | 0,40 | **1431** | **0** | 3,09 |
| Rec13 – 20x15 | 1930 | 2002 | **1930** | **0** | **0,140** | **1930** | **0** | 0,72 | 1933 | 0,16 | 1,13 | 1944 | 0,73 | 2,08 |
| Rec15 – 20x15 | 1950 | 2013 | **1950** | **0** | 0,118 | **1950** | **0** | **0,07** | 1951 | 0,01 | 1,13 | 1966 | 0,82 | 1,66 |
| Rec17 – 20x15 | 1902 | 2019 | **1902** | **0** | **0,092** | **1902** | **0** | 0,82 | 1920 | 0,94 | 0,92 | 1928 | 1,37 | 3,36 |
| Rec19 – 30x10 | 2093 | 2185 | 2099 | **0,286** | 0,655 | 2101 | 0,38 | 0,69 | 2104 | 0,53 | 1,86 | 2120 | 1,29 | 2,85 |
| Rec21 – 30x10 | 2017 | 2131 | 2026 | 0,446 | 1,423 | 2021 | **0,20** | **1,29** | 2032 | 0,75 | 1,52 | 2046 | 1,44 | 2,50 |
| Rec23 – 30x10 | 2011 | 2155 | ***2009**** | - | **0,208** | 2019 | 0,40 | 0,74 | 2021 | 0,50 | 1,74 | 2039 | 1,39 | 3,47 |
| Rec25 – 30x15 | 2513 | 2644 | **2513** | **0** | 0,902 | 2520 | 0,28 | **0,68** | 2523 | 0,40 | 2,30 | 2576 | 2,51 | 3,69 |
| Rec27 – 30x15 | 2373 | 2498 | 2379 | 0,253 | 0,939 | 2379 | **0,25** | **0,86** | 2402 | 1,22 | 2,02 | 2406 | 1,39 | 2,80 |
| Rec29 – 30x15 | 2287 | 2391 | 2297 | 0,437 | **0,839** | 2307 | 0,87 | 1,42 | 2307 | 0,87 | 2,41 | 2358 | 3,10 | 3,92 |
| Rec31 – 50x10 | 3045 | 3173 | 3056 | 0,360 | 1,667 | 3058 | 0,43 | 1,63 | 3106 | 2,00 | 3,01 | 3142 | 3,19 | 3,88 |
| Rec33 – 50x10 | 3114 | 3241 | **3114** | **0** | 0,435 | **3114** | **0** | **0,16** | 3121 | 0,22 | 1,10 | 3140 | 0,83 | 2,08 |
| Rec35 – 50x10 | 3277 | 3313 | **3277** | **0** | **0** | **3277** | **0** | **0** | **3277** | **0** | **0** | **3277** | **0** | 0,21 |
| Rec37 – 75x20 | 4951 | 5227 | 5068 | 2,363 | 3,288 | 5087 | 1,94 | 3,25 | 5104 | 2,28 | 4,73 | 5139 | 3,80 | 4,77 |
| Rec39 – 75x20 | 5087 | 5308 | 5140 | 1,042 | 1,689 | 5188 | 1,98 | 2,17 | 5195 | 2,12 | 3,31 | 5237 | 2,95 | 3,62 |
| Rec41 – 75x20 | 4960 | 5292 | 5067 | 2,157 | 2,157 | 5148 | 3,38 | 4,86 | 5098 | 2,78 | 4,34 | 5204 | 4,92 | 5,53 |

**Not:** *BKS (Bilinen en iyi değer), ABC (yazar tarafından geliştirilen algoritma), BD (Belirtilen algoritma ile bulunan en iyi değer). Bulunan en iyi değerler **koyu** yazılmıştır. * Bilinen en iyi sonuçtan daha iyi bir değer. PSO-EDA_PI algoritmasına ait değerler 10 tekrar ile diğer algoritmaların değerleri 20 tekrar ile bulunmuştur.*

**Tablo 3.7 İki Yönlü İşaret Testi İçin α = 0,05 ve α = 0,10 Değerlerine ait Kritik Değerler**

| Vaka sayısı | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | **21** | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| α= 0.05 | 5 | 6 | 7 | 7 | 8 | 9 | 9 | 10 | 10 | 11 | 12 | 12 | 13 | 13 | 14 | 15 | **15** | 16 | 17 | 18 | 18 |
| α= 0.10 | 5 | 6 | 6 | 7 | 7 | 8 | 9 | 9 | 10 | 10 | 11 | 12 | 12 | 13 | 13 | 14 | **14** | 15 | 16 | 16 | 17 |

Reeves test problemleri için ABC veya karşılaştırılan algoritmanın aldığı galibiyet sayıları sayılarak işaret testi uygulanabilir. Bu değerler, Tablo 3.7'de yer alan kritik değerlerle karşılaştırılarak anlamlı bir farklılık olup olmadığı tespit edilir. Elde edilen değerler, Tablo 3.8'de yer almaktadır. ABC algoritması PGA dışındaki diğer tüm algoritmalara göre α = 0,05 anlamlılık seviyesinde bir ilerleme göstermektedir.

**Tablo 3.8 İkili Kıyaslamalar İçin İşaret Testi Sonuçları**

| **ABC** | **$PSO_{MA}$** | **$PSO_{VNS}$** | **SGA** | **SGA+NEH** | **$HGA_1$** | **EM** | **HDABC** | **PGA** | **HGIA** | **$HGA_2$** |
|---|---|---|---|---|---|---|---|---|---|---|
| Galibiyet (+) | 20 | 21 | 21 | 21 | 19 | 21 | 20 | 13 | 20 | 21 |
| Kayıp (-) | - | - | - | - | 1 | - | - | 7 | - | - |
| Anlamlılık seviyesi | α = 0.05 | α = 0.05 | α = 0.05 | α = 0.05 | α = 0.05 | α = 0.05 | α = 0.05 | - | α = 0.05 | α = 0.05 |

Wilcoxon işaretli sıra testi "ele alınan iki örnek iki farklı popülasyona mı ait?" sorusuna cevap aramaktadır. İki bağımlı örnek testlerinde hipotez testi için kullanılmaktadır. Eşleştirilmiş t testinin parametrik olmayan türü olup iki örnek ortalaması (iki algoritmanın davranışı) arasında anlamlı bir fark bulunup bulunmadığının tespitine yönelik kullanılmaktadır (Derrac vd., 2011).

Tek bir bağımlı gruptan iki farklı algoritma ile ölçüm yapılıp sonuçlar karşılaştırıldığı ve veri setindeki problemlerin sırası her algoritmada aynı olduğu (sıra bağımlılık) için Wilcoxon işaretli sıra testi kullanılmıştır.

Yapılan testte $R^+$ birinci algoritmanın ikinci algoritmandan daha iyi performans gösterdiği sıraların toplamı, $R^-$ değeri ise ikinci algoritmanın birinci algoritmadan daha iyi performans gösterdiği sıraların toplamıdır. Elde edilen test sonuçları Tablo 3.9'da verilmiştir.

Tablo 3.9'da yer alan *p* değerleri dikkate alındığında ABC algoritması PGA algoritmasına göre α = 0,05 anlamlılık seviyesinde; PSOMA, PSOVNS, SGA, SGA+NEH, $HGA_1$, EM, HDABC, HGIA ve $HGA_2$ algoritmalarına göre de α = 0,01 anlamlılık seviyesinde gelişim

göstermektedir. ABC algoritması ile bulunan değerler ele alınan bütün algoritmalara göre gelişim göstermektedir.

**Tablo 3.9 Wilcoxon İşaretli Sıra Testi Sonuçları**

| Karşılaştırma | $R^+$ | $R^-$ | *p*-değeri | Karşılaştırma | $R^+$ | $R^-$ | *p*-değeri |
|---|---|---|---|---|---|---|---|
| ABC vs $PSO_{MA}$ | 210 | - | 0.00 | ABC vs EM | 231 | - | 0.00 |
| ABC vs $PSO_{VNS}$ | 231 | - | 0.00 | ABC vs HDABC | 210 | - | 0.00 |
| ABC vs SGA | 231 | - | 0.00 | ABC vs PGA | 161 | 49 | 0.04 |
| ABC vs SGA+NEH | 231 | - | 0.00 | ABC vs HGIA | 210 | - | 0.00 |
| ABC vs HGA1 | 205 | 5 | 0.00 | ABC vs HGA2 | 231 | - | 0.00 |

### 3.2.2.1.2. Friedman Testi

Bölüm 3.2.1'de ayrıntılı olarak açıklanan Friedman testi için kullanılan hipotezler aşağıda verilmiştir.

$$H_0: \mu_{ABC} = \mu_{PSO_{MA}} = \cdots = \mu_{HGA2} \quad (14)$$

$$H_1: \mu_{ABC} \neq \mu_{PSO_{MA}} \neq \cdots \neq \mu_{HGA2} \quad (15)$$

Friedman test sonuçları Tablo 3.10'da verilmiştir.

**Tablo 3.10 Friedman Testi ile Elde Edilen Ortalama Sıra Değerleri. Hesaplanan Test İstatistiği ve *p*-değeri Ayrıca Verilmiştir.**

| Algoritma | Friedman |
|---|---|
| ABC | 1,50 |
| $PSO_{MA}$ | 4,17 |
| $PSO_{VNS}$ | 6,74 |
| SGA | 11,00 |
| SGA+NEH | 9,38 |
| HGA1 | 5,05 |
| EM | 9,57 |
| HDABC | 3,24 |
| PGA | 2,29 |
| HGIA | 5,12 |
| HGA2 | 7,95 |
| *İstatistik* | *192,994* |
| *p-değeri* | *0.00* |

Friedman test sonuçlarına göre ABC algoritması ortalama 1,50 sıra değeri ile en iyi performans gösteren algoritmadır. Hesaplanan *p*-değeri, karşılaştırma yapılan diğer algoritmalara göre anlamlı bir farklılık olduğunu güçlü bir şekilde vurgulamaktadır.

### 3.2.3. Taillard Test problemleri

Çizelgeleme problemlerinde literatürdeki hemen hemen tüm çalışmalarda performans kıyaslaması için kullanılan test problemlerinden oluşan Taillard veri seti, değişik iş ve makine sayısına sahip toplam 120 adet problemden oluşmaktadır. 20, 50 ve 100 iş sayısına sahip veri setinin her biri için 5, 10 ve 20 makine sayısından oluşan üçer alt veri seti bulunmaktadır. Bu 9 veri setinin dışında 200 iş için 10 ve 20 makineden, son olarak da 500 iş 20 makineden oluşan toplam 12 farklı veri seti ve her bir veri setinde ise 10'ar problem yer almaktadır. Çalışmada her bir problem için algoritma on kez çalıştırılmış, bulunan en iyi değer, en iyi değerden sapma yüzdesi (BRE) ve on deneme için ortalama sapma değeri (ARE) verilmiştir.

Taillard problemlerinin çözümünde kaşif arı sayısı *5* ve algoritmanın çevrim sayısı *250* (200 iş problemleri için *150*) olarak alınmıştır. Algoritma ile elde edilen sonuçlar Tablo 3.11'de verilmiştir.

Tablo 3.11 incelendiğinde 20x5, 20x10, 20x20, 50x5 ve 100x5 problem gruplarındaki beşi hariç tüm veri seti için en iyi sonucu bulduğu gözükmektedir. Diğer problem grupları incelendiğinde özellikle 100x10 ve 200x10 problem grupları için en iyi değerden sapma ve ortalama sapma değerlerinin % 1'in altında olduğu gözükmektedir.

20 makineye sahip problemlerde, algoritmanın başarısı diğer veri setlerinden zayıf kalmakta fakat bu gruplar için dahi en iyi değerden sapma değeri % 1-3 arasında değişmektedir.

Algoritmanın performansı öncelikle diğer bir ABC temelli algoritma olan HDABC ile kıyaslanacaktır. Bunun dışında literatürde sıklıkla kullanılan ACS (Ying ve Liao, 2004), GA (Reeves, 1995), S-GA (self guided GA) (Chen vd., 2012), evrimsel gelişim algoritması DDE (Pan vd., 2008b), $PSO_{VNS}$ ve $PSO_{SPV}$ (Taşgetiren vd., 2007) algoritmaları ile karşılaştırmalar yapılacaktır. Bu değerler Tablo 3.12'de verilmiştir.

Geliştirilen ABC algoritmasının performansı öncelikle HDABC (Liu ve Liu, 2011) ile kıyaslanmıştır. HDABC ile yapılan çalışmada, yazarlar Taillard test problemlerinden sadece sekiz tanesini kullanmışlardır. İki test grubunda önerilen ABC algoritmasının daha başarılı, diğer gruplar için ise HDABC'nin küçük farklarla daha üstün sonuçlar verdiği görülmektedir.

Literatürde sıklıkla kullanılan GA ve ACS algoritmaları ile kıyaslama yapılırsa, ABC algoritmasının performansı her iki algoritmadan daha üstündür, sadece 100x20 ve 200x20 problem gruplarında elde edilen ortalama değerlerde ACS algoritması daha başarılıdır. PSO ile yerel arama tekniklerinin birleştirilmelerinden oluşan $PSO_{SPV}$ ve $PSO_{VNS}$ algoritmalarıyla yapılan kıyaslamada, algoritmanın $PSO_{SPV}$'den daha başarılı olduğu fakat $PSO_{VNS}$'nin önerilen ABC algoritmasından daha iyi sonuçlar verdiği görülmektedir. Dağılım tahmini algoritması (EDA) ise güçlendirilmiş GA'ya dayanan S-GA ile yapılan kıyaslamada, 200x20 problem grubu dışındakilerde ABC algoritması daha iyi değerler vermiştir. Son olarak DDE ile yapılan karşılaştırma, 200x20 problem grubu dışında ABC algoritmasının performans üstünlüğünü ortaya koymaktadır.

500x20 test grubu için yapılan ön çalışmada (çevrim sayısı:250, kaşif arı sayısı:5), algoritmanın işlem süresinin çok uzun olduğu görülmüştür. Bu yüzden, bu grup için çevrim sayısı olarak 100, kaşif arı sayısı da iki olarak alınmış ve daha kısa sürede benzer sonuçlar elde edilmiştir. Tablo 3.12'de yer aldığı gibi sadece DDE, ACS ve GA çalışmalarında 500x20 problemleri kullanılmıştır. Yapılan kıyaslamada, ABC algoritmasının GA'dan daha iyi fakat DDE ve ACS algoritmalarından küçük bir farkla düşük performans gösterdiği görülmektedir.

Bu çalışmada, literatürde yer alan pek çok eserde olduğu gibi, geliştirilen algoritmanın performansının ölçülmesi için her bir problem türünde birbirinden bağımsız on tekrar yapılmış ve elde edilen sonuçlar raporlanmıştır. Algoritmanın doğasında barındırdığı rastgelelik unsurunun performansa olan etkisinin ölçülebilmesi amacıyla, Taillard test problemleri kullanılarak her bir problem için algoritma 40'ar kez çalıştırılmıştır. Elde edilen değerler ile algoritmanın onar kez çalıştırılmasıyla elde edilen değerlerin karşılaştırılması Tablo 3.13'de verilmiştir.

Elde edilen en iyi değerler karşılaştırıldığında, tekrar sayısının artması sonucu dört problemde (t023, t026, t064, t094) bilinen en iyi değerin bulunduğu fakat t020 probleminde on tekrar sırasında bulunan en iyi değerin bu sefer bulunamadığı tespit edilmiştir. Tekrar sayısının artması sonucu, bazı problemlerde bulunan en iyi değerlerin daha iyi olduğu, bazı problemlerde ise on tekrar sayısı ile elde edilen değerlerden daha kötü olduğu görülmektedir. Artan tekrar sayısı sonucu 44 problemde daha iyi bir çözüm değeri bulunurken, 45 problemde ise bulunan en iyi çözüm değeri on tekrar ile bulunandan daha kötüdür. Ortalama sapma değeri kriteri incelenirse, 40 tekrar ile 24 problemde ARE değeri düşerken (ortalama % 8,33), 90 problemde bu değerde artış görülmüştür (ortalama %15,13). Tekrar sayısının artmasıyla ortaya çıkacak performans değişimi ile artan işlem süresi arasındaki ilişkinin dikkatlice değerlendirilmesi gerekmektedir.

**Tablo 3.11 Taillard Test Problemleri İçin Algoritmanın Performans Değerleri**

| **Problem grubu** | **Üst sınır** | | | | **Problem grubu** | **Üst sınır** | | | |
|---|---|---|---|---|---|---|---|---|---|
| *iş x makine* | | **BD** | **BRE %** | **ARE %** | *iş x makine* | | **BD** | **BRE %** | **ARE %** |
| *20 x 5* | | | | | *50 x 5* | | | | |
| 1 | 1278 | **1278** | **0** | **0** | 1 | 2724 | **2724** | **0** | **0** |
| 2 | 1359 | **1359** | **0** | **0** | 2 | 2834 | **2834** | **0** | 0,112 |
| 3 | 1081 | **1081** | **0** | **0** | 3 | 2621 | **2621** | **0** | **0** |
| 4 | 1293 | **1293** | **0** | 0,061 | 4 | 2751 | **2751** | **0** | 0,156 |
| 5 | 1235 | **1235** | **0** | **0** | 5 | 2863 | **2863** | **0** | **0** |
| 6 | 1195 | **1195** | **0** | **0** | 6 | 2829 | **2829** | **0** | 0,007 |
| 7 | 1234 | 1239 | 0,405 | 0,405 | 7 | 2725 | **2725** | **0** | **0** |
| 8 | 1206 | **1206** | **0** | **0** | 8 | 2683 | **2683** | **0** | **0** |
| 9 | 1230 | **1230** | **0** | **0** | 9 | 2552 | 2554 | 0,078 | 0,325 |
| 10 | 1108 | **1108** | **0** | **0** | 10 | 2782 | **2782** | **0** | **0** |
| *Ortalama* | | | **0,040** | **0,046** | *Ortalama* | | | **0,007** | **0,06** |
| *20 x 10* | | | | | *50 x 10* | | | | |
| 1 | 1582 | **1582** | **0** | 0,037 | 1 | 2991 | 3039 | 1,604 | 1,912 |
| 2 | 1659 | **1659** | **0** | 0,006 | 2 | 2867 | 2911 | 1,534 | 2,054 |
| 3 | 1496 | **1496** | **0** | 0,187 | 3 | 2839 | 2871 | 1,127 | 1,895 |
| 4 | 1377 | **1377** | **0** | 0,145 | 4 | 3063 | 3071 | 0,261 | 0,277 |
| 5 | 1419 | **1419** | **0** | 0,098 | 5 | 2976 | 3015 | 1,310 | 1,747 |
| 6 | 1397 | **1397** | **0** | 0,057 | 6 | 3006 | 3024 | 0,598 | 1,497 |
| 7 | 1484 | **1484** | **0** | 0 | 7 | 3093 | 3124 | 1,002 | 1,891 |
| 8 | 1538 | **1538** | **0** | 0,312 | 8 | 3037 | 3046 | 0,296 | 0,675 |
| 9 | 1593 | **1593** | **0** | 0 | 9 | 2897 | 2917 | 0,690 | 1,007 |
| 10 | 1591 | **1591** | **0** | 0,282 | 10 | 3065 | 3120 | 1,794 | 2,202 |
| *Ortalama* | | | **0** | **0,112** | *Ortalama* | | | **1,021** | **1,515** |
| *20 x 20* | | | | | *50 x 20* | | | | |
| 1 | 2297 | **2297** | **0** | 0,047 | 1 | 3847 | 3917 | 1,819 | 2,446 |
| 2 | 2099 | **2099** | **0** | 0,066 | 2 | 3704 | 3746 | 1,133 | 2,356 |
| 3 | 2326 | 2328 | 0,085 | 0,189 | 3 | 3640 | 3714 | 2,032 | 2,593 |
| 4 | 2223 | **2223** | **0** | 0,017 | 4 | 3719 | 3787 | 1,828 | 2,465 |
| 5 | 2291 | **2291** | **0** | 0,192 | 5 | 3610 | 3684 | 2,049 | 2,454 |
| 6 | 2226 | 2229 | 0,134 | 0,139 | 6 | 3679 | 3751 | 1,957 | 2,671 |
| 7 | 2273 | **2273** | **0** | 0,136 | 7 | 3704 | 3769 | 1,754 | 2,473 |
| 8 | 2200 | **2200** | **0** | 0,354 | 8 | 3691 | 3770 | 2,140 | 3,419 |
| 9 | 2237 | **2237** | **0** | 0,156 | 9 | 3741 | 3808 | 1,790 | 2,245 |
| 10 | 2178 | **2178** | **0** | 0,068 | 10 | 3756 | 3814 | 1,544 | 2,404 |
| *Ortalama* | | | **0,021** | **0,136** | *Ortalama* | | | **1,804** | **2,552** |

**Not:** *BD (Belirtilen algoritma ile bulunan en iyi değer). Bulunan en iyi değerler* ***koyu*** *yazılmıştır.*

**Tablo 3.11 Taillard Test Problemleri İçin Algoritmanın Performans Değerleri (… devam)**

| **Problem grubu** | **Üst sınır** | | | | **Problem grubu** | **Üst sınır** | | | |
|---|---|---|---|---|---|---|---|---|---|
| *iş x makine* | | **BD** | **BRE %** | **ARE %** | *iş x makine* | | **BD** | **BRE %** | **ARE %** |
| *100 x 5* | | | | | *200 x 10* | | | | |
| 1 | 5493 | **5493** | **0** | **0** | 1 | 10862 | 10872 | 0,092 | 0,187 |
| 2 | 5268 | **5268** | **0** | **0** | 2 | 10480 | 10555 | 0,715 | 0,811 |
| 3 | 5175 | **5175** | **0** | **0** | 3 | 10922 | 10965 | 0,393 | 0,932 |
| 4 | 5014 | 5017 | 0,059 | 0,075 | 4 | 10889 | 10893 | 0,036 | 0,125 |
| 5 | 5250 | **5250** | **0** | 0,034 | 5 | 10524 | 10537 | 0,123 | 0,315 |
| 6 | 5135 | **5135** | **0** | **0** | 6 | 10329 | 10350 | 0,203 | 0,488 |
| 7 | 5246 | **5246** | **0** | 0,036 | 7 | 10854 | 10882 | 0,257 | 0,616 |
| 8 | 5094 | **5094** | **0** | 0,125 | 8 | 10730 | 10754 | 0,223 | 0,514 |
| 9 | 5448 | **5448** | **0** | 0,062 | 9 | 10438 | 10465 | 0,258 | 0,377 |
| 10 | 5322 | **5322** | **0** | 0,078 | 10 | 10675 | 10727 | 0,487 | 0,517 |
| *Ortalama* | | | **0,005** | **0,041** | *Ortalama* | | | **0,278** | **0,488** |
| *100 x 10* | | | | | *200 x 20* | | | | |
| 1 | 5770 | 5783 | 0,225 | 0,509 | 1 | 11181 | 11440 | 2,316 | 2,648 |
| 2 | 5349 | 5387 | 0,710 | 0,837 | 2 | 11203 | 11489 | 2,552 | 3,178 |
| 3 | 5676 | 5679 | 0,052 | 0,292 | 3 | 11281 | 11631 | 3,102 | 3,546 |
| 4 | 5781 | 5808 | 0,467 | 0,989 | 4 | 11275 | 11590 | 2,793 | 3,113 |
| 5 | 5467 | 5502 | 0,640 | 1,068 | 5 | 11259 | 11503 | 2,167 | 2,530 |
| 6 | 5303 | 5308 | 0,094 | 0,147 | 6 | 11176 | 11455 | 2,496 | 2,841 |
| 7 | 5595 | 5608 | 0,232 | 0,689 | 7 | 11360 | 11610 | 1,200 | 2,840 |
| 8 | 5617 | 5650 | 0,587 | 1,114 | 8 | 11334 | 11602 | 2,364 | 2,635 |
| 9 | 5871 | 5880 | 0,153 | 0,571 | 9 | 11192 | 11454 | 2,341 | 2,831 |
| 10 | 5845 | 5903 | 0,992 | 0,992 | 10 | 11288 | 11613 | 2,879 | 3,333 |
| *Ortalama* | | | **0,415** | **0,720** | *Ortalama* | | | **2,421** | **2,949** |
| *100 x 20* | | | | | *500 x 20* | | | | |
| 1 | 6202 | 6383 | 2,918 | 3,286 | 1 | 26059 | 26498 | 1,684 | 1,955 |
| 2 | 6183 | 6298 | 1,859 | 2,796 | 2 | 26520 | 27019 | 1,881 | 2,018 |
| 3 | 6271 | 6408 | 2,184 | 2,924 | 3 | 26371 | 26803 | 1,638 | 1,802 |
| 4 | 6269 | 6388 | 1,898 | 2,249 | 4 | 26456 | 26850 | 1,489 | 1,654 |
| 5 | 6314 | 6476 | 2,565 | 3,202 | 5 | 26334 | 26552 | 0,827 | 0,995 |
| 6 | 6364 | 6508 | 2,626 | 2,685 | 6 | 26477 | 26842 | 1,378 | 1,781 |
| 7 | 6268 | 6410 | 2,265 | 3,026 | 7 | 26389 | 26661 | 1,030 | 1,077 |
| 8 | 6401 | 6552 | 2,359 | 3,071 | 8 | 26560 | 26923 | 1,366 | 1,663 |
| 9 | 6275 | 6443 | 2,677 | 3,101 | 9 | 26005 | 26350 | 1,326 | 1,614 |
| 10 | 6434 | 6554 | 1,865 | 2,202 | 10 | 26457 | 26775 | 1,201 | 1,391 |
| *Ortalama* | | | **2,321** | **2,854** | *Ortalama* | | | **1,382** | **1,595** |

**Not:** *BD (Belirtilen algoritma ile bulunan en iyi değer). Bulunan en iyi değerler* ***koyu*** *yazılmıştır.*

**Tablo 3.12 Taillard Test Problemleri İçin Algoritmanın Performans Karşılaştırması**

| **Problem boyutu** *(iş x makine)* | **ABC** | **HDABC** | **DDE** | **$PSO_{SPV}$** | **$PSO_{VNS}$** | **GA** | **ACS** | **S-GA** |
|---|---|---|---|---|---|---|---|---|
| | **ARE %** | **ARE %** | **ARE %** | **ARE %** | **ARE %** | **ARE %** | **ARE %** | **ARE %** |
| 20, 5 | 0,046 | **0** | 0,46 | 1,75 | 0,03 | 0,65 | 1,19 | 1,10 |
| 20, 10 | 0,112 | 0,15 | 0,93 | 3,25 | **0,03** | 1,87 | 1,70 | 1,90 |
| 20, 20 | 0,136 | 0,20 | 0,79 | 2,82 | **0,05** | 1,34 | 1,60 | 1,60 |
| 50, 5 | 0,060 | 0,02 | 0,17 | 1,14 | **0** | 0,28 | 0,43 | 0,52 |
| 50, 10 | 1,515 | 0,99 | 2,26 | 5,29 | **0,57** | 2,76 | 1,89 | 2,74 |
| 50, 20 | 2,552 | 1,98 | 3,11 | 7,21 | **1,36** | 4,04 | 2,71 | 3,94 |
| 100, 5 | 0,041 | **0** | 0,08 | 0,63 | **0** | 0,19 | 0,22 | 0,38 |
| 100, 10 | 0,720 | 0,52 | 0,94 | 3,27 | **0,18** | 1,19 | 1,22 | 1,60 |
| 100, 20 | 2,854 | | 3,24 | 8,25 | **1,45** | 3,34 | 2,22 | 3,51 |
| 200, 10 | 0,488 | | 0,55 | 2,47 | **0,18** | 0,61 | 0,64 | 0,80 |
| 200, 20 | 2,949 | | 2,61 | 8,05 | 1,35 | 3,39 | **1,30** | 1,85 |
| 500, 20 | 1,595 | | **1,43** | | | 1,79 | 1,68 | |

**Not:** *Bulunan en iyi değerler* ***koyu*** *yazılmıştır.*

**Tablo 3.13 Tekrar Sayısının Algoritma Performansı Üzerindeki Etkisi**

| | **10 tekrar** | | **40 tekrar** | | | **10 tekrar** | | **40 tekrar** | |
|---|---|---|---|---|---|---|---|---|---|
| Problem | BRE % | ARE % | BRE % | ARE % | Problem | BRE % | ARE % | BRE % | ARE % |
| *t001* | **0** | 0 | **0** | 0 | *t031* | **0** | 0 | **0** | 0 |
| *t002* | **0** | 0 | **0** | 0 | *t032* | **0** | 0,112 | **0** | 0,132 |
| *t003* | **0** | 0 | **0** | 0,049 | *t033* | **0** | 0 | **0** | 0,003 |
| *t004* | **0** | 0,061 | **0** | 0,275 | *t034* | **0** | 0,156 | **0** | 0,263 |
| *t005* | **0** | 0 | **0** | 0,018 | *t035* | **0** | 0 | **0** | 0,004 |
| *t006* | **0** | 0 | **0** | 0,073 | *t036* | **0** | 0,007 | **0** | 0,026 |
| *t007* | **0,405** | 0,405 | **0,405** | 0,669 | *t037* | **0** | 0 | **0** | 0,013 |
| *t008* | **0** | 0 | **0** | 0 | *t038* | **0** | 0 | **0** | 0,017 |
| *t009* | **0** | 0 | **0** | 0 | *t039* | **0,078** | 0,325 | **0,078** | 0,345 |
| *t010* | **0** | 0 | **0** | 0,002 | *t040* | **0** | 0 | **0** | 0,001 |
| *t011* | **0** | 0,037 | **0** | 0,117 | *t041* | 1,604 | 1,912 | **1,471** | 2,289 |
| *t012* | **0** | 0,006 | **0** | 0,351 | *t042* | **1,534** | 2,054 | 1,849 | 2,495 |
| *t013* | **0** | 0,187 | **0** | 0,491 | *t043* | **1,127** | 1,895 | **1,127** | 1,941 |
| *t014* | **0** | 0,145 | **0** | 0,448 | *t044* | **0,261** | 0,277 | **0,261** | 0,697 |
| *t015* | **0** | 0,098 | **0** | 0,386 | *t045* | 1,310 | 1,747 | **0,370** | 1,807 |
| *t016* | **0** | 0,057 | **0** | 0,433 | *t046* | **0,598** | 1,497 | 0,798 | 1,949 |
| *t017* | **0** | 0 | **0** | 0,049 | *t047* | **1,002** | 1,891 | **1,002** | 2,011 |
| *t018* | **0** | 0,312 | **0** | 0,762 | *t048* | 0,296 | 0,675 | **0,165** | 0,757 |
| *t019* | **0** | 0 | **0** | 0,113 | *t049* | 0,690 | 1,007 | **0,483** | 1,235 |
| *t020* | **0** | 0,282 | 0,189 | 0,742 | *t050* | 1,794 | 2,202 | **1,175** | 2,339 |
| *t021* | **0** | 0,047 | **0** | 0,308 | *t051* | **1,819** | 2,446 | 2,002 | 2,786 |
| *t022* | **0** | 0,066 | **0** | 0,166 | *t052* | **1,133** | 2,356 | 1,620 | 2,480 |
| *t023* | 0,085 | 0,189 | **0** | 0,506 | *t053* | **2,032** | 2,593 | 2,088 | 3,087 |
| *t024* | **0** | 0,017 | **0** | 0,074 | *t054* | 1,828 | 2,465 | **1,667** | 2,616 |
| *t025* | **0** | 0,192 | **0** | 0,427 | *t055* | 2,049 | 2,454 | **1,357** | 2,557 |
| *t026* | 0,134 | 0,139 | **0** | 0,338 | *t056* | **1,957** | 2,671 | 2,039 | 2,770 |
| *t027* | **0** | 0,136 | **0** | 0,339 | *t057* | 1,754 | 2,473 | **1,404** | 2,561 |
| *t028* | **0** | 0,354 | **0** | 0,451 | *t058* | 2,140 | 3,419 | **1,951** | 3,546 |
| *t029* | **0** | 0,156 | **0** | 0,190 | *t059* | **1,790** | 2,245 | 1,951 | 2,515 |
| *t030* | **0** | 0,068 | **0** | 0,333 | *t060* | 1,544 | 2,404 | **1,464** | 2,750 |

**Not:** *Bulunan en iyi değerler **koyu** yazılmıştır.*

**Tablo 3.13 Tekrar Sayısının Algoritma Performansı Üzerindeki Etkisi (… devam)**

| | **10 tekrar** | | **40 tekrar** | | | **10 tekrar** | | **40 tekrar** | |
|---|---|---|---|---|---|---|---|---|---|
| Problem | BRE % | ARE % | BRE % | ARE % | Problem | BRE % | ARE % | BRE % | ARE % |
| *t061* | **0** | 0 | **0** | 0 | *t091* | **0.092** | 0.187 | **0.092** | 0.193 |
| *t062* | **0** | 0 | **0** | 0.138 | *t092* | 0.715 | 0.811 | **0.534** | 0.767 |
| *t063* | **0** | 0 | **0** | 0.034 | *t093* | 0.393 | 0.932 | **0.238** | 0.876 |
| *t064* | 0.059 | 0.075 | **0** | 0.078 | *t094* | 0.036 | 0.125 | **0** | 0.060 |
| *t065* | **0** | 0.034 | **0** | 0.057 | *t095* | **0.123** | 0.315 | 0.124 | 0.344 |
| *t066* | **0** | 0 | **0** | 0.005 | *t096* | **0.203** | 0.488 | 0.232 | 0.490 |
| *t067* | **0** | 0.036 | **0** | 0.013 | *t097* | 0.257 | 0.616 | **0.184** | 0.609 |
| *t068* | **0** | 0.125 | **0** | 0.168 | *t098* | **0.223** | 0.514 | 0.224 | 0.564 |
| *t069* | **0** | 0.062 | **0** | 0.107 | *t099* | **0.258** | 0.377 | 0.259 | 0.424 |
| *t070* | **0** | 0.078 | **0** | 0.104 | *t100* | 0.487 | 0.517 | **0.094** | 0.490 |
| *t071* | **0.225** | 0.509 | 0.260 | 0.532 | *t101* | 2.316 | 2.648 | **2.209** | 2.703 |
| *t072* | 0.710 | 0.837 | **0.355** | 0.852 | *t102* | **2.552** | 3.178 | 2.624 | 3.115 |
| *t073* | **0.052** | 0.292 | 0.053 | 0.251 | *t103* | 3.102 | 3.546 | **2.828** | 3.482 |
| *t074* | **0.467** | 0.989 | 0.519 | 1.016 | *t104* | 2.793 | 3.113 | **2.262** | 2.956 |
| *t075* | 0.640 | 1.068 | **0.530** | 1.046 | *t105* | 2.167 | 2.530 | **1.803** | 2.242 |
| *t076* | **0.094** | 0.147 | **0.094** | 0.183 | *t106* | 2.496 | 2.841 | **2.380** | 2.860 |
| *t077* | 0.232 | 0.689 | **0.125** | 0.744 | *t107* | **1.200** | 2.840 | 2.192 | 2.667 |
| *t078* | **0.587** | 1.114 | 0.588 | 1.092 | *t108* | 2.364 | 2.635 | **2.162** | 2.528 |
| *t079* | **0.153** | 0.571 | 0.341 | 0.789 | *t109* | **2.341** | 2.831 | 2.493 | 2.846 |
| *t080* | 0.992 | 0.992 | **0.051** | 0.945 | *t110* | **2.879** | 3.333 | **2.879** | 3.289 |
| *t081* | 2.918 | 3.286 | **2.854** | 3.594 | *t111* | 1.684 | 1.955 | **1.558** | 1.841 |
| *t082* | **1.859** | 2.796 | 2.297 | 3.221 | *t112* | 1.881 | 2.018 | **1.610** | 1.940 |
| *t083* | **2.184** | 2.924 | 2.217 | 3.068 | *t113* | 1.638 | 1.802 | **1.323** | 1.696 |
| *t084* | 1.898 | 2.249 | **1.595** | 2.515 | *t114* | 1.489 | 1.654 | **1.191** | 1.487 |
| *t085* | 2.565 | 3.202 | **2.407** | 3.295 | *t115* | 0.827 | 0.995 | **0.782** | 1.017 |
| *t086* | 2.626 | 2.685 | **2.184** | 3.019 | *t116* | 1.378 | 1.781 | **1.144** | 1.588 |
| *t087* | **2.265** | 3.026 | 2.505 | 3.476 | *t117* | 1.030 | 1.077 | **0.947** | 1.067 |
| *t088* | **2.359** | 3.071 | 2.671 | 3.517 | *t118* | 1.366 | 1.663 | **0.877** | 1.579 |
| *t089* | 2.677 | 3.101 | **2.534** | 3.279 | *t119* | 1.326 | 1.614 | **1.238** | 1.658 |
| *t090* | 1.865 | 2.202 | **1.818** | 2.290 | *t120* | 1.202 | 1.392 | **1.134** | 1.439 |

**Not:** *Bulunan en iyi değerler **koyu** yazılmıştır.*

# SONUÇ

Çizelgeleme problemi günümüz işletmelerinin karar alma süreçlerinde en sık karşılaştıkları problemlerden birisidir. Çizelgeleme probleminin özel bir alt türü olan iş çizelgeleme, işlerin makinelere atanması olarak özetlenebilir. İki ve üç makine durumunda kesin çözüm kolaylıkla elde edilebilmektedir. Makine sayısının üçü geçtiği durumlarda problem karmaşıklaşmakta ve en iyi çözümü elde etmek için gerekli olan zaman üstel bir biçimde artmaktadır. İşlem süresindeki bu artış sonucunda, kesin çözüm bulmayı hedefleyen matematiksel tekniklerin yerini çeşitli sezgisel teknikler almaya başlamıştır.

Bu çalışmada literatürde oldukça yeni bir metasezgisel yöntem olan yapay arı kolonisi algoritması permütasyon akış tipi çizelgeleme problemine uygulanmıştır. Klasik çizelgeleme probleminin genelleştirilmiş bir durumu olan PFSP'de, işler tüm makinelerde aynı sırada işlem görmektedir. Geliştirilen algoritma ile ele alınan performans kriterini en iyileyecek iş çizelgesi oluşturulmaya çalışılmış ve diğer algoritmalarla performans karşılaştırılması yapılmıştır.

Literatürde yapılan analizde, akış tipi çizelgeleme problemlerinde başta en geç tamamlanma zamanı, toplam akış süresi ve toplam gecikme süresi olmak üzere çeşitli performans göstergelerinin kullanıldığı tespit edilmiştir. Bu çalışmada performans kriteri olarak, birçok çalışmada sıklıkla kullanılan en geç tamamlanma zamanı kullanılmıştır.

Geliştirilen algoritmanın performansı literatürde yer alan çeşitli problemler (Carlier, Reeves ve Taillard) üzerinde denenmiştir. Carlier test problemlerinde, tüm problem grupları için bilinen en iyi değerlere ulaşılmıştır. Reeves test problemlerinde ise 21 problemden 13 tanesinde bilinen en iyi değere ulaşılmış, bunun yanı sıra Rec23 problemi için yeni bir en iyi değer bulunmuştur. Reeves problem grubu için algoritmanın performansı 11 farklı metasezgisel algoritma ile kıyaslanmış, 5 problemde PSO-EDA_PI algoritması, 2 problemde ise PGA algoritması ABC algoritmasından daha iyi değerler bulmuştur. ABC tabanlı diğer bir algoritma olan HDABC ile yapılan karşılaştırmada ise geliştirilen ABC algoritmasının 21 problem içinde hem en iyi

değerden sapma yüzdesi (BRE) hem de ortalama sapma yüzdesi (ARE) kriterlerinde daha başarılı olduğu tespit edilmiştir.

Geliştirilen algoritmanın performansını ölçmek için parametrik olmayan istatistiksel testler kullanılmıştır. Anlamlı bir karşılaştırma yapabilmek için ele alınan algoritmaların aynı tekrar sayısına sahip olması gerekmektedir, bu yüzden analizlerde PSO-EDA_PI algoritması kullanılmamıştır. İşaret testi ve Wilcoxon işaretli sıra testi kullanılarak ABC algoritması ikili kıyaslamalar aracılığıyla diğer algoritmalarla karşılaştırılmıştır. Elde edilen sonuçlar ABC algoritmasının üstünlüğünü göstermektedir. Son olarak Friedman testi kullanılarak yapılan analizde ABC algoritmasının en iyi sıra değerine sahip olduğu görülmektedir.

Taillard test problemleri için algoritma 120 problemden 45 tanesinde en iyi değeri bulmuştur. Algoritma ile beş ve on makine problem grupları için oldukça başarılı sonuçlar elde edilmiştir. Geliştirilen algoritmanın Taillard 20 makine problemlerine uygulanması sonucu elde edilen değerler ile bilinen en iyi değerler arasındaki sapmada beş ve on makine problemlerine göre artış gözlenmiştir.

Literatürde yer alan HDABC algoritması, Taillard test problemlerinden sadece ilk sekiz gruba uygulanmıştır. Elde edilen sonuçlar ABC algoritması ile kıyaslanırsa; iki problem grubunda önerilen algoritmanın, diğer altı problem grubunda ise HDABC algoritmasının daha başarılı değerler verdiği görülmektedir. Her iki algoritmada küçük farklarla birbirlerine üstünlük sağlamıştır. Geliştirilen algoritma ile elde edilen ortalama değerler, beş farklı algoritma ile kıyaslanmıştır. VNS yerel arama tekniğini içeren PSO algoritması dışında ele alınan tüm algoritmalardan daha başarılı sonuçlar elde edilmiştir.

Performans ölçümü sırasında işlem süresine yönelik herhangi bir karşılaştırma yapılmamıştır. Karşılaştırma amacıyla ele alınan algoritmaların, farklı yıllarda farklı donanımlara sahip bilgisayarlarla yapılmaları ve yazarlar tarafından adı geçen eserlerde uygulama kodlarının verilmemesinden dolayı karşılaştırma yapılmamıştır.

Permütasyon akış tipi çizelgeleme problemine uyarlanan ABC algoritmasının, yapılacak bazı değişikliklerle farklı varsayımlar içeren (atölye tipi çizelgeleme, işlerin bölünebilir olduğu çizelgeleme problemi, ara stoklara izin verildiği çizelgeleme problemi) ve farklı performans kriterlerine sahip (toplam gecikme süresi, toplam akış süresi) çizelgeleme problemlerine de uygulanabileceği düşünülmektedir.

Özellikle Taillard test problemlerinde makine sayısının artmasıyla ortaya çıkan performans azalmasının algoritma üzerinde yapılacak gelecekteki çalışmalarda giderilmesi planlanmaktadır.

Algoritmanın başarılı performansı, onun çeşitli problemlere de kolaylıkla uygulanabileceği fikrini vermektedir. Bu açıdan PFSP için geliştirilen bu algoritmanın, diğer araştırmacılara yol göstermesi beklenmektedir.

# KAYNAKÇA

Agarwal A., Colak S., Eryarsoy E., "Improvement Heuristic for the Flow-shop Scheduling Problem: An Adaptive Learning Approach", European Journal of Operational Research, Vol. 169, (2006), 801-815.

Akay B., Karaboğa D., "Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm", AI*IA 2009: Emergent Perspectives in Artificial Intelligence, Lecture Notes in Computer Science, Volume 5883, 355-364, (2009), DOI: 10.1007/978-3-642-10291-2_36.

Akay B., Karaboğa D., "Artificial Bee Colony Algorithm for Large-Scale Problems and Engineering Design Optimization", Journal of Intelligent Manufacturing, (2010), DOI: 10.1007/s10845-010-0393-4.

Banharnsakun A., Achalakul T., Sirinaovakul B. "The best-so-far selection in Artificial Bee Colony algorithm", Applied Soft Computing, Vol. 11, No. 2, (2011), 2888-2901.

Baykasoğlu A., Özbakır L., Tapkan P., "Artificial bee colony algorithm and its application to generalized assignment problem." Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, der. Chan F.T.S., Tiwari M.K., 113-144, Itech Education and Publishing, Vienna, Austria, 2007.

Ben-Daya M., Al-Fawzan M., "A Tabu Search Approach for the Flow Shop Scheduling Problem", European Journal of Operational Research, Vol. 109, (1998), 88-95.

Blum C., Roli A., "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparision", ACM computing surveys, Vol. 35, No. 3, (2003), 268-308.

Chen C.L., Vempati V.S., Aljaber N., "An application of genetic algorithms for flow shop problems", European Journal of Operational Research, Vol. 80, (1995), 389-396.

Chen S.H., Chang P.C., Cheng T.C.E., Zhang Q., "A self-guided genetic algorithm for permutation flowshop scheduling problems", Computers and Operations Research, Vol. 39, No. 7, (2012), 1450-1457.

Conway R.W., Maxwell W.L., Miller L.W., Theory of Scheduling, Addison-Wesley, USA, 1967.

Cura T., Modern Sezgisel Teknikler ve Uygulamaları, Papatya Yayıncılık, İstanbul, 2008.

Dannenbring, D.G., "An evaluation of flow shop sequencing heuristics", Management Sciences, Vol. 23, No. 11, (1977), 1174-1182.

Derrac J., Garcia S., Molina D., Herrera F., "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", Swarm and Evolutionary Computation Vol. 1, (2011), 3-18.

Dorigo M., Maniezzo V., Colorni A., "Positive feedback as a search strategy", Dipartiment di Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.

Duda J., "Local Search and Nature Based Metaheuristics: A Case of Flowshop Scheduling Problem", Proceedings of the International Multiconference on Computer Science and Information Technology, (2006), 17-24.

Ekşioğlu B., Ekşioğlu S.D., Jain P., "A Tabu Search Algorithm for the Flow Shop Scheduling Problem with Changing Neighborhoods", Computers & Industrial Engineering, Vol. 54, No. 1, (2008), 1-11.

Etiler O., Toklu B., Atak M., Wilson J., "A Genetic Algortihm for Flow Shop Scheduling Problems", Journal of the Operational Research Society, Vol. 55, (2004), 830-835.

Framinan J.M., Gupta J.N.D., Leisten R., "A Review and Classification of Heuristics for Permutation Flow-shop Scheduling with Makespan Objective", Journal of the Operational Research Society, Vol. 55, (2004), 1243-1255.

Garcia S., Molina D., Lozano M., Herrera F., "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behavior: a case study on the CEC'2005 special session on real parameter optimization", Journal of Heuristics, Vol. 15, (2009), 617-644.

Gao W.F., Li S.Y., "A modified artificial bee colony algorithm", Computers & Operations Research, Vol. 39, No. 3, (2012), 687-697.

Geyik F., Cedimoğlu I.H., "Atölye tipi çizelgeleme için uzman sistem tekniği ile basit öncelik kurallarının karşılaştırılması", Politeknik Dergisi, 4/1, (2001), 53-61.

Grabowski J., Wodecki M., "A Very Fast Tabu Search Algorithm for the Permutation Flow Shop Problem with Makespan Criterion", Computers & Operations Research, Vol. 31, (2004), 1891-1909.

Gupta J.N.D., Stafford E.F., "Flowshop Scheduling Research of the Five Decades", European Journal of Operational Research, Vol. 169, (2006), 699-711.

Ho J.C., Chang Y.L., "A new heuristic for the n-job, m-machine flow-shop problem", European Journal of Operational Research, Vol. 52, (1991), 194-202.

Jarboui B., Ibraim S., Siarry P., Rebai A., "A combinatorial particle swarm optimisation for solving permutation flowshop problems", Computers & Industrial Engineering, Vol. 54, No. 3, (2008), 526-538.

Irani R., Nasimi R., "Application of artificial bee colony-based neural network in bottom hole pressure prediction in underbalanced drilling", Journal of Petroleum Science and Engineering, Vol. 78, No. 1, (2011), 6-12.

Iyer S., Saxena B., "Improved genetic algorithm for the permutation flowshop scheduling problem", Computers & Operations Research, Vol. 31, No. 4, (2004), 593-606.

Kang F., Li J., Ma Z., "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions", Information Sciences, Vol. 181, No. 16, (2011), 3508-3531.

Karaboğa D., Yapay Zeka Optimizasyon Algoritmaları, Atlas Yayın Dağıtım, İstanbul, 2004.

Karaboğa D., "An idea based on honey bee swarm for numerical optimization", Technical Report TR06, Bilgisayar Mühendisliği Bölümü, Erciyes Üniversitesi, Türkiye, 2005.

Karaboğa D., Baştürk B., "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Journal of Global Optimization, Vol. 39, No. 3, (2007), 459-471, DOI: 10.1007/s10898-007-9149-x.

Karaboğa D., Baştürk B., "On The Performance Of Artificial Bee Colony (ABC) Algorithm", Applied Soft Computing, Vol. 8, No. 1, (2008), 687-697.

Karaboğa N., "A new design method based on artificial bee colony algorithm for digital IIR filters", Journal of the Franklin Institute, Vol. 346, No. 4, (2009), 328-348.

Karaboğa D., Akay B., "A Comparative Study of Artificial Bee Colony Algorithm", Applied Mathematics and Computation, Vol. 214, (2009), 108-132.

Karaboğa D., Öztürk C., "Neural Networks Training by Artificial Bee Colony Algorithm on Pattern Classification", Neural Network World, Vol. 19, No. 3, (2009), 279-292.

Karaboğa D., Akay B., "PID Controller Design by using Artificial Bee Colony, Harmony Search and The Bees Algorithms", Proceedings of the Institution of Mechanical Engineers, Part I, Journal of Systems and Control Engineering, Vol. 224, No. 7, (2010), 869-883.

Karaboğa D., Öztürk C., "Fuzzy clustering with artificial bee colony algorithm", Scientific Research and Essays, Vol. 5, No. 14, (2010), 1899-1902.

Karaboğa D., Öztürk C., "A novel clustering approach: Artificial Bee Colony (ABC) algorithm", Applied Soft Computing, Vol. 11, No. 1, (2011), 652-657.

Karaboğa D., Akay B., "A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems", Applied Soft Computing Volume, Vol. 11, No. 3, (2011), 3021-3031.

Kashan M.H., Nahavandi N., Kashan A.H., "DisABC: A new artificial bee colony algorithm for binary optimization", Applied Soft Computing, Vol. 12, (2012), 342-352.

Kellegöz T., "Toplam Geç Bitirme Zamanının En Küçüklenmesi Performans Ölçütlü Permütasyon Akış Tipi Çizelgeleme Problemlerinin Çözümünde Genetik Algoritma Yaklaşımı", Yüksek Lisans Tezi, Kırıkkale Üniversitesi, Fen Bilimleri Enstitüsü, Kırıkkale, 2006.

Keskintürk T., "Diferansiyel gelişim algoritması", İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, Vol. 5, No. 9, (2006), 85-99.

Laha D., "Heuristics and Metaheuristics for Solving Scheduling Problems", Handbook of Computational Intelligence in Manufacturing and Production Management, der. Laha D., Mundal P., 1-18, 2008.

Li G., Niu P., Xiao X., "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization", Applied Soft Computing, Vol. 12, (2012), 320-332.

Lian Z., Gu X., Jiano B., "A Similar Particle Swarm Optimization Algorithm for Permutation Flowshop Scheduling to Minimize Makespan", Applied Mathematics and Computation, Vol. 175, (2006), 773-785.

Lian Z., Gu X., Jiano B., "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan", Chaos, Solitons & Fractals, Vol. 35, No. 5, (2008), 851-861.

Liu B., Wang L., Jin Y.H., "An Effective PSO-Based Memetic Algorithm for Flow Shop Scheduling", IEEE Transactions on System, Man, and Cybernatics-Part B:Cybernetics, Vol. 37, No. 1, (2007), 18-27.

Liu H., Gao L., Pan Q., "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem", Expert Systems with Applications, Vol. 38, (2011), 4348-4360.

Liu Y.F., Liu S.Y., "A hybrid artificial bee colony algorithm for permutation flowshop scheduling problem", Applied Soft Computing, (2011), http://dx.doi.org/10.1016/ j.asoc.2011.10.024, basım aşamasında.

Luengo J., Garcia S., Herrera F., "A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric test", Expert Systems with Applications, Vol. 36, (2009), 7798-7808.

Marinakis Y., Marinaki M., Matsatsinis N., "A hybrid discrete artificial bee colony-GRASP algorithm for clustering, International Conference on Computers and Industrial Engineering", 548-553, Troyes, Fransa, 2009.

Moccellin, J.a.V., "A new heuristic method for the permutation flow shop scheduling problem", Journal of the Operational Research Society, Vol. 46, (1995), 883-886.

Morton T.E., Pentico D.W., Heuristic Scheduling Systems, Wiley-Interscience, USA, 1993.

Murata T., Ishibuchi H., Tanaka H., "Genetic algorithm for flowshop scheduling problems", Computers and Industrial Engineering, Vol. 30, No. 4, (1996), 1061-1071.

Nawaz M., Enscore Jr. E.E., Ham I, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem", OMEGA, The International Journal of Management Science, Vol. 11, No. 1, (1983), 91-95.

Nearchou A.C., "A Novel Metaheuristic Approach for the Flow Shop Scheduling Problem", Engineering Applications of Artificial Intelligence, Vol. 17, (2004), 289-300.

Nowicki E., Smutnicki C., "A fast tabu search algorithm for the permutation flow-shop problem", European Journal of Operational Research, Vol. 91, (1996), 160-175.

Nowicki E., Smutnicki C., "Some Aspects of Scatter Search in the Flow-shop Problem", European Journal of Operational Research, Vol. 169, (2006), 653-666.

Onwubolu G., Davendra D., "Scheduling Flow Shops Using Differential Evoluation Algorithm", European Journal of Operational Research, Vol. 171, (2006), 674-692.

Özkan C., Kisi Ö, Akay B., "Neural networks with artificial bee colony algorithm for modeling daily reference evapotranspiration", Irrigation Science, Vol. 29, No. 6, (2010), 431-441, DOI: 10.1007/s00271-010-0254-0.

Pan Q.K., Wang L., Zhao B.H., "An Improved Iterated Greedy Algorithm for the No-wait Flow Shop Scheduling Problem with Makespan Criterion", International Journal of Advanced Manufacturing Technologies, Vol. 38, (2008a), 778-786.

Pan Q.K., Tasgetiren M.F., Liang Y.C., "A discrete differential evolution algorithm for the permutation flowshop scheduling problem", Computers & Industrial Engineering, Vol. 55, No. 4, (2008b), 795-816.

Pan Q.K., Taşgetiren M.F., Suganthan P.N., Chua T.J., A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem, Information Sciences, Vol. 181, No. 12, (2011), 2455-2468.

Pinedo M., Scheduling: Theory, Algorithms and Systems, 3rd edition, Printice-Hall, New Jersey, 2008.

Ponnambalam S.G., Aravindan P., Chandrasekaran S., "Constructive and Improvement Flow Shop Scheduling Heuristics: An Extensive Evaluation", Production Planning & Control, Vol. 12, No. 4, (2001), 335-344.

Ponnambalam S.G., Jagannathan H., Kataria M., Gadicherla A., "A TSP_GA multi-objective algorithm for flow-shop scheduling", International Journal of Advanced Manufacturing Technologies, Vol. 23, (2004), 909-915.

Rao R.V., Patel V.K., "Optimization of mechanical draft counter flow wet-cooling tower using artificial bee colony algorithm", Energy Conversion and Management, Vol. 52, No. 7, (2011), 2611-2622.

Rajendran C., Ziegler H., "Ant-colony Algorithms for Permutation Flowshop Scheduling to Minimize Makespan/Total Flowtime of Jobs", European Journal of Operations Research, Vol. 155, (2004), 426-438.

Rameshkumar K., Suresh R.K., Mohanasundaram K.M., "Discrete Particle Swarm Optimization (DPSO) Algorithm for Permutation Flowshop Scheduling to Minimize Makespan", Lecture Notes in Computer Science, Vol. 3612, 2005.

Reeves C.R., "Improving the efficiency of tabu search for machine scheduling problem", Journal of Operational Research Society, Vol. 44, No. 4, (1993), 375-382.

Reeves C.R., "A genectic algorithm for flowshop sequencing", Computers and Operational Research, Vol. 22, No. 1, (1995), 5-13.

Reeves C., Yamada T., "Genetic algorithms, path relinking, and the flowshop sequencing problem", Evolutionary Computation, Vol. 6, No. 1, (1998), 45-60.

Ruiz R., Maroto C., "A Comprehensive Review and Evaluation of Permutation Flowshop Heuristics", European Journal of Operational Research, Vol. 165, (2005), 479-494.

Ruiz R., Maroto C., Alcaraz J., "Two New Robust Genetic Algorithms for the Flowshop Scheduling Problem", Omega, Vol. 34, (2006), 461-476.

Ruiz R., Stützle T., "A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem", European Journal of Operational Research, Vol. 177, (2007), 2033-2049.

Safarzadeh O., Zolfaghari A., Norouzi A., Minuchehr H., "Loading pattern optimization of PWR reactors using Artificial Bee Colony", Annals of Nuclear Energy, Vol. 38, No. 10, (2011), 2218-2226.

Sarin S., Lefoka M., "Scheduling heuristic for the n-job m-machine flow shop", OMEGA, The International Journal of Management Science, Vol. 21, No. 2, (1993), 229-234.

Sayadi M.K., Ramezanian R., Ghaffari-Nasab N., "A Discrete Firefly Meta-Heuristic with Local Search for Makespan Minimization in Permutation Flow Shop Scheduling Problems", International Journal of Industrial Engineering Computations, Vol. 1, (2010), 1-10.

Seçme G., "Akış tipi çizelgeleme problemlerinin yapay sinir ağları ile modellenmesi", Yüksek lisans tezi, Erciyes Üniversitesi SBE, 2006.

Shukran M.A.M., Chung Y.Y., Yeh W.C., Wahid N., Zaidi A.M.A., "Artificial Bee Colony based Data Mining Algorithms for Classification Tasks", Modern Applied Science, Vol. 5, No. 4, (2011), 217-231.

Singh A., "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem", Applied Soft Computing, Vol. 9, No. 2, (2009), 625-631.

Sönmez M., "Artificial Bee Colony algorithm for optimization of truss structures", Applied Soft Computing, Vol. 11, No. 2, (2011), 2406-2418.

Suliman S., "A two-phase heuristic approach to the permutation flow-shop scheduling problem", International Journal of Production Economics, Vol. 64, (2000), 143-152.

Sundar S., Singh A., "A Swarm Intelligence Approach to the Quadratic Minimum Spanning Tree Problem", Information Sciences, Vol. 180, No. 17, (2010), 3182-3191.

Sundar S., Singh A., Rossi A., "An artificial bee colony algorithm for the 0-1 multidimensional knapsack problem", Communications in Computer and Information Science, Vol. 94, (2010), 141-151.

Stützle T., "An Ant Approach to the Flow Shop Problem", 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT 98), 1560-1564, Aachen, Germany, (1998a).

Stützle T., "Applying Iterated Local Search to the Permutation Flow Shop Problem", Teknik Rapor, AIDA-98-04, FG Intellektik, TU Darmstadt, (1998b).

Szeto W.Y., Wu Y.H., Ho S.C., "An artificial bee colony algorithm for the capacitated vehicle routing problem", European Journal of Operational Research, Vol. 215, No. 1, (2011), 126-135.

Taillard E., "Some Efficient Heuristc Methods for the Flow Shop Sequencing Problem", European Journal of Operational Research, Vol. 47, (1990), 65-74.

Tang J., Zhang G., Lin B., Zhang B., "Hybrid genetic Algorithm for flow shop scheduling problem", International conferenece on intelligent computation technology and automation, 449-452, 2010.

Taşgetiren M.F., Liang Y.C., Sevki M., Gencyılmaz G., "Differential Evolution Algorithm for Permutation Flowshop Sequencing Problem with Makespan Criteria", Proceedings of the 4th International Symposium on Intelligent Manufacturing System (IMS 2004), Sakarya, Türkiye, 442-452, 2004.

Taşgetiren M.F., Liang Y.C., Sevkli M., Gencyılmaz G., "A Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in the Permutation Flowshop Sequencing Problem", European Journal of Operational Research, Vol. 177, (2007), 1930-1947.

Taşgetiren M.F., Pan Q.K., Suganthan P.N., Chen A.H., "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops", Information Sciences, Vol. 181, No. 16, (2011), 3459-3475.

Tseng L.Y., Lin Y.T., "A hybrid genetic local search algorithm for the permutation flowshop scheduling problem", European Journal of Operational Research, Vol. 198, No. 1, (2009), 84-92.

Wang L., Zheng D.Z., "An Effective Hybrid Heuristic for Flow Shop Scheduling", International Journal of Advanced Manufacturing Technologies, Vol. 21, (2003), 28-44.

Watase K., "A Factorial Representation of Permutations and Its Applications to Flow-shop Scheduling", Systems and Computers in Japan, Vol. 38, No. 1, (2007), 73-85.

Widmer M., Hertz A., "A new heuristic method for the flow shop sequencing problem", Eurpean Journal of Operational Research, Vol. 41, (1989), 186-193.

Vonderembse M.A., White G.P., Operations Management – Concepts, Methods and Strategies, West Pub. Company, St. Paul, MN, ABD, 1991.

Yağmahan B., Yenisey M.M., "Akış Tipi Çizelgeleme Problemi için KKE Parametre Eniyileme", İtü Dergisi/d, Vol. 5, No. 2, (2006), 133-141.

Yağmahan B., Yenisey M.M., "Ant Colony Optimization for the Multi-objective Flow Shop Scheduling Problem", Computers & Industrial Engineering, Vol. 54, (2008), 411-420.

Yağmahan B., Yenisey M.M., "A Multi-objective Ant Colony System Algorithm for Flow Shop Scheduling Problem", Expert Systems with Applications, Vol. 37, (2010), 1361-1368.

Ying K.C., Liao C.J., "An Ant Colony System for Permutaion Flow-shop Sequencing", Computers & Operations Research, Vol. 31, (2004), 791-801.

Zhang C.S., Ouyang D., Ning J., "An artificial bee colony approach for clustering", Expert Systems with Applications, Vol. 37, No. 7, (2010), 4761-4767.

Zhu G., Kwong S., "Gbest-guided artificial bee colony algorithm for numerical function optimization", Applied Mathematics and Computation, Vol. 217, No. 7, (2010), 3166-3173.

Zobolas G.I., Tarantilis C.D., Ioannon G., "Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems", Studies in Computational Intelligence, Vol. 128, (2008), 1-40.

Zobolas G.I., Tarantilis C.D., Ioannou G., "Minimizing Makespan in Permutation Flow Shop Scheduling Problems Using a Hybrid Metaheuristic Algorithm", Computers & Operations Research, Vol. 36, (2009), 1249-1267.

**EK – 1 ABC Algoritması İçin Hazırlanan Matlab Kodu**

```
clear all
close all
clc
tic
global n
global m
global local_best_ms
global local_best_tour
global data
global NP
NP=25;
iter=250;
load 'rec01.txt';
data=rec01;
[n m]=size (data);
tour_init=zeros(NP,n);
m_s=zeros(NP,1);
m_s_new=zeros(NP,1);
local_best_ms=zeros(NP,1);
local_best_tour=zeros(NP,n);
for i=1:NP-1
   tour_init(i,:) = randperm(n);
   m_s(i,:)=makespankang(tour_init(i,:));
end

NEH_1;

tour_init(NP,:)=best_scheduling;
```

```
m_s(NP,:)=make_spanNEH_best;
GlobalBestTour=tour_init;
GlobalMakeSpan=m_s;
local_best_ms=m_s;
local_best_tour=tour_init;

for o=1:iter
   for i=1:NP
      a=tour_init(i,:);
      rand_numb = round(random('uniform', 1, 5));
      switch rand_numb
         case 1
            tour_new=swap (1,a);
         case 2
            tour_new=insert(1,a);
         case 3
            tour_new=swap (2,a);
         case 4
            tour_new=insert(2,a);
         case 5
            tour_new=deconstruction(4,a);
      end
      m_s_new=makespankang(tour_new);
       if m_s_new <= m_s(i,:)
         local_best_ms(i,:)=m_s_new;
         local_best_tour(i,:)=tour_new;
      end
   end

   for i=1:(2*NP)
         [old, old_tour, old_tour_index]=tournamentSelection(3);
```

```
        m_s_old=makespankang(old);
        rand_numb = round(random('uniform', 1, 4));

      switch rand_numb
        case 1
          tour_new2=swap (1,old);
        case 2
          tour_new2=insert(1,old);
        case 3
          tour_new2=swap (2,old);
        case 4
          tour_new2=insert(2,old);
      end
  ms_new=makespankang(tour_new2);

    if ms_new < m_s_old
     local_best_ms(old_tour_index)=ms_new;
     local_best_tour(old_tour_index,:)=tour_new2;
     else
    end
  end
  for g=1:NP
      if local_best_ms(g) < GlobalMakeSpan(g)
        GlobalMakeSpan(g)=local_best_ms(g);
        GlobalBestTour(g,:)=local_best_tour(g,:);
      end
end
      BestInd=find(GlobalMakeSpan==min(GlobalMakeSpan));
      [u s]=size (BestInd);
      if u>1
        BestInd=BestInd(1);
```

```
        end
  a=GlobalBestTour(BestInd,:);
    for i=1:5
       [old1, old_tour1, old_tour_index1]=tournamentSelectionWorst(3);
       [tour_newIG tour_newIG_ms]=IG_1 (a);
       GlobalMakeSpan(old_tour_index1)=tour_newIG_ms;
       GlobalBestTour(old_tour_index1,:)=tour_newIG;
    end
        BestInd=find(GlobalMakeSpan==min(GlobalMakeSpan));
        [u s]=size (BestInd);
        if u>1
           BestInd=BestInd(1);
        end
        GlobalMinMS_iter(o,:)=GlobalMakeSpan(BestInd);
        GlobalMinTour_iter(o,:)=GlobalBestTour(BestInd,:);
end
toc
```

# Ö Z G E Ç M İ Ş

**Adı ve SOYADI** : Ömür TOSUN
**Doğum Tarihi ve Yeri** : 22/04/1980 – Kaş / ANTALYA
**Medeni Durumu** : Evli

**Eğitim Durumu**

**Mezun Olduğu Lise** : Antalya Anadolu Lisesi
**Lisans Diploması** : Dokuz Eylül Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü
**Yüksek Lisans Diploması** : Akdeniz Üniversitesi Sosyal Bilimler Enstitüsü, İşletme Ana Bilim Dalı
**Tez Konusu** : Tamir – Bakım Planlaması ve Bir Uygulama
**Doktora Diploması** : Akdeniz Üniversitesi Sosyal Bilimler Enstitüsü, İşletme Ana Bilim Dalı
**Tez Konusu** : Yapay Arı Kolonisi Algoritması ve Permütasyon Akış Tipi Çizelgeleme Problemine Uygulanması
**Yabancı Dil / Diller** : İngilizce, Almanca
**Çalıştığı Kurumlar** : Akdeniz Üniversitesi Sosyal Bilimler Enstitüsü İşletme Ana Bilim Dalı Araştırma Görevlisi (2005 - 2012)
**E-Posta** : omurtosun@akdeniz.edu.tr