T. C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KOMBİNATORYAL OPTİMİZASYON PROBLEMLERİNDE ARI SİSTEMİ YAKLAŞIMI

Tezi Hazırlayan
Pınar Zarif TAPKAN

Tezi Yöneten
Yrd. Doç. Dr. Lale ÖZBAKIR

Bilgisayar Mühendisliği Anabilim Dalı
Doktora Tezi

Haziran 2010
KAYSERİ

Yrd. Doç. Dr. Lale ÖZBAKIR danışmanlığında **Pınar Zarif TAPKAN** tarafından hazırlanan **"Kombinatoryal Optimizasyon Problemlerinde Arı Sistemi Yaklaşımı"** adlı bu çalışma, jürimiz tarafından Erciyes Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında **Doktora** tezi olarak kabul edilmiştir.

14.06.2010

JÜRİ:

Başkan: Prof. Dr. Hadi Gökçen

Üye : Prof. Dr. Adil Baykasoğlu

Üye : Prof. Dr. Derviş Karaboğa

Üye : Doç. Dr. Coşkun Özkan

Üye : Yrd. Doç. Dr. Lale Özbakır

ONAY:

Bu tezin kabulü, Enstitü Yönetim Kurulunun 22/06/2010 tarih ve 2010/20/02 sayılı kararı ile onaylanmıştır.

14 /06/2010

Prof. Dr. Nusret AYYILDIZ
Enstitü Müdürü

TEŞEKKÜR

Çalışmalarım boyunca, tez konumun belirlenmesi ve yürütülmesinde değerli görüş ve katkılarıyla beni yönlendiren, her konuda desteğini esirgemeyen, kıymetli tecrübelerinden faydalandığım hocalarım Sayın Yrd. Doç. Dr. Lale ÖZBAKIR'a ve Sayın Prof. Dr. Adil BAYKASOĞLU'na teşekkürü bir borç bilirim. Ayrıca EÜBAP-FBT-07-94 kodlu proje ile çalışmalarıma maddi destek sağlayan Erciyes Üniversitesi Rektörlüğü, Bilimsel Araştırma Projeleri Birimine teşekkür ederim.

Doktora çalışmamın her aşamasında beni teşvik eden, anlayış ve desteği ile her zaman yanımda olan, bugüne gelmemde büyük emeği geçen TAN ailesine ve eşim Cem TAPKAN'a; her an gülen yüzüyle tüm yorgunluğumu alan, yanımda olarak bana çalışma azmi veren hayatımın anlamı kızım Lâl TAPKAN'a; yanımda olamasa da desteğini her zaman hissettiren canım kardeşim Irmak Tuğba KAVAK'a ve çalışmalarım sırasında bana destek olan tüm arkadaşlarıma, özellikle de Sinem KULLUK'a teşekkür ederim.

KOMBİNATORYAL OPTİMİZASYON PROBLEMLERİNDE ARI SİSTEMİ YAKLAŞIMI

Pınar Zarif TAPKAN

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Doktora Tezi, Haziran, 2010
Tez Danışmanı: Yrd. Doç. Dr. Lale ÖZBAKIR

ÖZET

Birçok gerçek hayat probleminin kombinatoryal optimizasyon problemi olarak modellenebilmesi ve klâsik optimizasyon tekniklerinin bu tür problemleri çözmedeki çeşitli yetersizlikleri, kombinatoryal optimizasyon problemlerinin çözümünde hızlı ve etkin olarak kullanılacak araçların geliştirilmesi ihtiyacını doğurmuştur. Bu amaçla problemden ve modelden bağımsız bir yapıya sahip olan doğadan esinlenmiş sezgisel optimizasyon algoritmaları, son yıllarda artan bir hızla zor kombinatoryal optimizasyon problemlerinin çözümünde kullanılmaktadır. Bu tekniklerin bir dalı olan sürü zekâsı algoritmaları ise böceklerin problem çözme becerilerini taklit eden metasezgisel yöntemler geliştirebilmek için böcek davranışlarına odaklanmıştır.

Arıların yiyecek arama davranışları, öğrenme, hatırlama ve bilgi paylaşma özellikleri sürü zekâsının en ilgi çekici araştırma alanlarından birisidir. Birbiriyle etkileşen bireyler sistemi olarak ele alınan arı kolonisinde kolektif zekâ, sinerjik bilgi değişimine dayanmaktadır. Temel olarak, bulunan yiyecek kaynaklarının kalitesi hakkında bilgi paylaşımının gerçekleştirildiği arı kolonisinde amaç, farklı ve kaliteli yiyecek kaynaklarına ulaşabilmek için kolonideki diğer arıların da iyi bölgelere çekilmesine dayanmaktadır. Arılar arasındaki bu etkileşim, zor kombinatoryal optimizasyon problemlerine kaliteli ve uygun çözümlerin daha hızlı bulunmasını sağlamaktadır.

Bu tez çalışmasının amacı zor kombinatoryal optimizasyon problemlerine iyi çözümler üreten ve arı davranışlarını modelleyen yapay sistemler geliştirmektir. Bu doğrultuda zor kombinatoryal optimizasyon problemleri sınıfında yer alan Genelleştirilmiş Atama Problemi ve Çift Taraflı Montaj Hattı Dengeleme Problemi'ne etkin bir çözüm yaklaşımı geliştirmek amacıyla son yıllarda önerilen Arı Algoritması ve Yapay Arı Kolonisi Algoritması'ndan faydalanılmış ve oldukça başarılı sonuçlar elde edilmiştir.

Anahtar Kelimeler: Arı Sistemi, Arı Algoritması, Yapay Arı Kolonisi Algoritması, Genelleştirilmiş Atama Problemi, Çift Taraflı Montaj Hattı Dengeleme Problemi.

BEE SYSTEM APPROACH FOR COMBINATORIAL OPTIMIZATION PROBLEMS

Pınar Zarif TAPKAN

Erciyes University, Graduate School of Natural and Applied Sciences
Ph. D. Thesis, June, 2010
Thesis Supervisor: Assist. Prof. Lale ÖZBAKIR

ABSTRACT

Due to the fact that most of real life problems can be modelled as a combinatorial optimization problem and presence of various insufficiencies on solving these problems by classical optimization techniques, it has required developing rapid and effective tools to solve combinatorial optimization problems. For this purpose, problem and model independent nature inspired heuristic optimization algorithms have been utilized for solving hard combinatorial optimization problems with an increasing trend. Such a branch of nature inspired algorithms which are known as swarm intelligence focuses on insect behavior in order to develop some meta-heuristics which can mimic insect's problem solution abilities.

The foraging behaviour, learning, memorizing and information sharing characteristics of bees have recently been one of the most interesting research areas in swarm intelligence. The collective intelligence of interacting bee colony is based on synergic information exchange. Basically, the aim of the bee colony depends on attracting other bees to productive locations to collect different and qualified food sources by sharing information about quality of food sources. This interaction among bees provides finding qualified and feasible solutions to hard combinatorial optimization problems much more quickly.

This thesis is focused on developing artificial systems that generate good solutions to hard combinatorial optimization problems by utilizing bee behaviours. Accordingly, with the aim of developing an effective solution approach for Generalized Assignment Problem and Two-Sided Assembly Line Balancing Problem that can be classified as hard combinatorial optimization problems, recently proposed Bees Algorithm and Artificial Bee Colony Algorithm are utilized and considerably effective results are obtained.

Keywords: Bee System, Bees Algorithm, Artificial Bee Colony Algorithm, Generalized Assignment Problem, Two-Sided Assembly Line Balancing Problem.

İÇİNDEKİLER

KISALTMA VE SİMGELER

GAP	: Genelleştirilmiş Atama Problemi
ÇTMHDP	: Çift Taraflı Montaj Hattı Dengeleme Problemi
AA	: Arı Algoritması
YAK	: Yapay Arı Kolonisi
PSO	: Parçacık Sürü Optimizasyonu
KKO	: Karınca Koloni Optimizasyonu
AÇO	: Arıların Çiftleşme Optimizasyonu
SP	: Sağlanabilirlik Problemi
BAÇO	: Bal Arılarının Çiftleşme Optimizasyonu
ARUAP	: Açgözlü Rastgele Uyarlanır Arama Prosedürü
KAE	: Kraliçe Arı Evrim
AS	: Arı Sistemi
GSP	: Gezgin Satıcı Problemi
SARP	: Stokastik Araç Rotalama Problemi
AKO	: Arı Kolonisi Optimizasyonu
MHDP	: Montaj Hattı Dengeleme Problemi
TTMHDP	: Tek Taraflı Montaj Hattı Dengeleme Problemi
ÇTKMMHDP	: Çift Taraflı Karma Model Montaj Hattı Dengeleme Problemi
EKİS	: En Kısa İşlem Süresi
EUİS	: En Uzun İşlem Süresi
MiİSa	: Kendinden Sonraki İş Sayısı Toplamının Minimumu
MaİSa	: Kendinden Sonraki İş Sayısı Toplamının Maksimumu
MiİSü	: Kendinden Sonraki İşlerin Toplam İşlem Süresinin Minimumu
MaİSü	: Kendinden Sonraki İşlerin Toplam İşlem Süresinin Maksimumu
MaSKA	: Maksimum Sıralı Konumsal Ağırlık
MaOSKA	: Maksimum Ortalama Sıralı Konumsal Ağırlık
PBÖ	: Pozitif Bölgesel Kısıt Önceliği
KÖ	: Konumsal Kısıt Önceliği
SÖ	: Senkronizasyon Kısıtı Önceliği
BAP	: Bulanık Amaç Programlama

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

1. BÖLÜM
GİRİŞ

Matematiksel model, bir süreci ya da sistem davranışını matematiksel olarak temsil eden denklemler takımıdır. Matematiksel modeller çeşitli özelliklerine göre farklı sınıflara ayrılabilirler. Bu sınıflandırmalardan biri de karar değişkenlerinin pozitif reel değerler alması ya da tamsayı değerler alması şeklinde ikiye ayrılan sürekli optimizasyon problemi ve kesikli optimizasyon problemidir. Sürekli optimizasyon problemleri tüm fonksiyonların doğrusal olması durumunda doğrusal programlama, en az bir fonksiyonun doğrusal olmaması durumunda ise doğrusal olmayan programlama yöntemi ile çözülmektedir. Diğer taraftan kesikli optimizasyon problemleri ise karar değişkenlerinin tamsayı değerler alması durumunda tamsayılı programlama, karar değişkenlerinin kombinatoryal seçenekleri söz konusu olduğunda ise kombinatoryal optimizasyon yöntemleri ile çözülmektedir.

Tez çalışmasının temelini oluşturan kombinatoryal optimizasyon problemleri, dikkate alınan amaç fonksiyonunu en iyileyen kesikli karar değişkenlerinin değerlerinin bulunması ile ilgilenir. Diğer bir deyişle kombinatoryal optimizasyon, belirli kısıtlardan oluşmuş, sayılabilir sonlu bir uygun çözüm uzayında, en küçükleme ya da en büyükleme ölçütüyle, matematiksel olarak modellenmiş problemlerin incelenmesi ve en iyi çözümün bulunmasını içerir. Teorik ve pratik önemi olan çoğu optimizasyon problemi kombinatoryal yapıdadır; bu tür problemlere örnek olarak en kısa yol problemi, gezgin satıcı problemi, atama problemi, atölye çizelgeleme problemi ve araç rotalama problemi verilebilir. Kombinatoryal optimizasyon problemleri çözüm açısından hem kolay hem de zor problemleri bünyesinde barındırmakta ve bu bağlamda P ve NP-zor olmak üzere iki sınıfa ayrılmaktadır. Bir problem P sınıfında ise erişilebilir, kolay ve optimum çözümü elde edilebilir nitelikte; NP-zor sınıfında ise erişilemez, zor ve optimum çözümü makul zamanlarda elde edilemez niteliktedir. P sınıfındaki bir problem, çözüm zamanı problem boyutunun polinom fonksiyonu olarak artan bir

algoritma ile çözülebilir. Örneğin atama, minimum kapsayan ağaç ve şebeke akış problemleri P sınıfında yer alan kombinatoryal optimizasyon problemlerindendir. NP-zor sınıfındaki problemlerin çözümü için ise polinom zamanlı bir algoritma yoktur; çünkü optimum çözümü bulmak için gerekli süre, problem boyutuna bağlı olarak üstel artış göstermektedir. 0 ya da 1 değerini alan *n* değişkene sahip bir problem için tüm çözümlerin birerleme zamanı $O(2^n)$'dir. Küçük boyutlu problemler birerleme ile çözülebilmesine rağmen, büyük boyutlu problemler için bu yöntem ile çözüme ulaşmak mümkün değildir. NP-zor sınıfındaki problemler için dal-sınır ya da kesme düzlemi gibi etkin yöntemlerin başarısız olmasının nedeni, bu yöntemlerin de üstel sınırlara sahip olmasıdır. NP-zor yapıya sahip kombinatoryal optimizasyon problemlerine örnek olarak ise karesel atama problemi, gezgin satıcı problemi, montaj hattı dengeleme problemi, araç rotalama problemi, çizelgeleme problemi ve yer seçimi problemi verilebilir.

Kombinatoryal optimizasyon problemlerinin çözümünde kullanılan algoritmalar kesin algoritmalar ve sezgisel algoritmalar olmak üzere ikiye ayrılır. Doğrusal programlama, dinamik programlama, dal-sınır algoritması, kesme düzlem yöntemi gibi kesin algoritmalar optimum çözümü garanti eden algoritmalardır. Sezgisel algoritmalar ise optimum çözümü garanti etmeksizin daha az çözüm zamanı ile optimuma yakın iyi bir çözümü elde etmeyi hedefler. Optimum olmayan çözümler üretebilen bu tekniklerin geliştirilmesinin sebebi, problem boyutunun çok büyük olması ya da problemin küçük alt problemlere ayrılmasının zor olduğu durumlarda kesin yöntemlerle probleme çözüm bulmanın mümkün olmamasıdır.

Bahsedildiği gibi geleneksel matematiksel yöntemlerle sezgisel algoritmalar arasındaki en önemli ayrım çözümün optimalliği ve hesaplama süresidir. Geleneksel matematiksel yöntemler optimum çözümü verebilmekte ancak büyük boyutlu örnekler düşünüldüğünde hesaplama süresi bir dezavantaj haline gelmektedir. Ayrıca geleneksel yaklaşımlar orijinal problem üzerinde bazı varsayımlar ve basitleştirmeler gerektirmekte ve bulunan çözüm orijinal problemin temsili için yeterli olmayabilmektedir. Araştırmacılar çeşitli optimizasyon problemlerini klasik optimizasyon prosedürlerine uyarlamak için oldukça çaba göstermişlerdir. Ancak bir gerçek hayat problemini belli bir çözüm prosedürüne uyacak şekilde modellemek genellikle pek kolay değildir. Klasik optimizasyon algoritmaları farklı tipte değişkenler, amaç fonksiyonu ve kısıtlar içeren

problem formülasyonlarına uygulanabilecek genel bir çözüm stratejisi sunmamaktadır. Örneğin simpleks algoritması, doğrusal amaç fonksiyonu ve kısıtlara sahip modellerin çözümünde; geometrik programlama ise pozitif katsayılı polinom ya da işaretçe sınırlı olması gerekmeyen polinom yapıda amaç fonksiyonuna sahip doğrusal olmayan modellerin çözümünde kullanılabilmektedir. Ancak birçok optimizasyon problemi aynı formülasyon içinde farklı tipte değişkenler, amaç fonksiyonları ve kısıtlar içermektedir. Bu zorluğu aşabilmek için orijinal problemin parametreleri üzerinde bazı değişiklikler ya da varsayımlar yapmak gerekmekte (değişkenlerin yuvarlanması, kısıtların gevşetilmesi gibi) ancak bu da çözümün kalitesini etkilemektedir. Dolayısıyla klasik optimizasyon teknikleri bu tür problemlerin çözümü için yeterli olmamaktadır. İşte klasik optimizasyon tekniklerinin bu yetersizliklerini aşabilmek için problemden ve modelden bağımsız olan doğadan esinlenmiş sezgisel optimizasyon algoritmaları önerilmektedir. Bu teknikler hem etkin hem de daha esnek olup belirli problem gereksinimlerine göre uyarlanabilmektedir [1].

Birçok gerçek hayat probleminin kombinatoryal optimizasyon problemi olarak modellenebilmesi, bu tür problemlerin çözümünde hızlı ve etkin olarak kullanılacak araçların geliştirilmesi ihtiyacını doğurmuştur. Son yıllarda metasezgisel algoritmalar artan bir hızla zor kombinatoryal optimizasyon problemlerinin çözümünde kullanılmaktadır. Çünkü gerçek hayatta genellikle sadece iyi çözümlere ihtiyaç duyulmakta yani karar vericiler için alt optimum bir çözüm de yeterli olabilmektedir. Başlangıçta metasezgisel algoritmalar benzetimli tavlama, genetik algoritma ve tabu aramadan oluşurken, daha iyi çözümler üretebilmek için son yıllarda birçok farklı teknik geliştirilmiştir. Bu tekniklerin bir dalı olan sürü zekâsına dayalı algoritmalar ise böceklerin problem çözme becerilerini taklit eden metasezgisel yöntemler geliştirebilmek için böcek davranışlarına odaklanmıştır. Sosyal böcek kolonileri çevreden bilgi toplayan ve bu bilgiye göre davranışlarını uyarlayan dinamik bir sistem olarak düşünülebilir. Böcekler arasındaki etkileşimin bir sonucu olan kolektif zekânın en önemli parçalarından biri ise bireysel böcekler arasındaki bilgi paylaşımıdır.

Diğer taraftan arıların yiyecek arama davranışları, öğrenme, hatırlama ve bilgi paylaşma özellikleri sürü zekâsının en ilgi çekici araştırma alanlarından birisidir. Bir arı kolonisi birbiriyle etkileşen bireyler sistemi olarak düşünülebilir. Bu tür etkileşimli davranış örneklerinden biri de bal arılarının buldukları yiyecek kaynağı hakkındaki bilgiyi

paylaştıkları salınım dansıdır. Bu dans aracılığıyla kaliteli bir yiyecek kaynağı bulan arılar, yiyecek kaynağı hakkındaki yön, uzaklık ve nektar miktarı bilgilerini diğer arılarla paylaşırlar. Böylece farklı yiyecek kaynakları bulabilmek için, kolonideki diğer arılar da iyi bölgelere çekilmektedir. Arıların dans davranışlarıyla ilgili yapılmış çalışmalar, arının dans esnasındaki yönünün yiyecek kaynağı ile güneş arasındaki ilişkiyi, dansın yoğunluğunun yiyecek kaynağının uzaklığını ve dansın süresinin ise nektar miktarını gösterdiğini ortaya çıkarmıştır. Yani arıların kolektif zekâsı sinerjik bilgi değişimine dayanmaktadır [2, 3].

Bal arılarının davranışlarına yönelik yapılan araştırmalar yeni optimizasyon algoritmalarının geliştirilmesini sağlamıştır. Bal arılarında yiyecek toplama ölçütü yeni yiyecek kaynaklarının ne kadar hızlı bulunup tüketildiğiyle; yapay yiyecek toplama ise benzer olarak uygun ya da iyi kalitedeki çözümlerin bulunma hızıyla ilişkilidir. Yeni yiyecek kaynaklarının bulunma maliyeti ise arılar arasındaki etkileşime bağlı olarak azalmaktadır. Yani yapay arılar arasındaki etkileşim uygun çözümlerin daha hızlı bulunmasını sağlamakta ve bulunan yiyecek kaynaklarının kalitesi de artmaktadır. Arılar arasındaki bu etkileşim zor kombinatoryal optimizasyon problemlerine iyi çözümler bulmaya yardımcı olmaktadır.

Bu tez çalışmasının amacı da zor kombinatoryal optimizasyon problemlerine iyi çözümler üreten ve arı davranışlarını modelleyen yapay sistemler geliştirmektir. Bu doğrultuda sırasıyla Fisher ve ark. [4] ve Bartholdi [5] tarafından NP-zor yapıya sahip olduğu ispatlanan Genelleştirilmiş Atama Problemi (GAP) ve Çift Taraflı Montaj Hattı Dengeleme Problemi'ne (ÇTMHDP), Arı Algoritması (AA) [6] ve Yapay Arı Kolonisi (YAK) algoritması [7] temel alınarak çözüm aranmış ve ilgili algoritmaların bu zor kombinatoryal optimizasyon problemleri üzerindeki performansı incelenmiştir. Her iki problem için de bilimsel yazındaki kesin ve sezgisel çözüm yöntemleriyle yapılan karşılaştırmalar, önerilen algoritmaların etkin bir performansa sahip olduğunu göstermiştir.

Tez çalışmasının ikinci bölümü, sürü zekâsı optimizasyon tekniklerinden biri olan arı sisteminin kombinatoryal optimizasyon problemlerinin çözümünde kullanımına temel oluşturması için sürü zekâsı ve arı sistemine ayrılmıştır. İlk olarak sürü zekâsından ve bu kavrama dayalı optimizasyon algoritmalarından bahsedildikten sonra doğadaki

gerçek arı davranışları detaylandırılarak bilimsel yazında arı sistemi ile bağlantılı mühendislik çalışmaları incelenmiştir. Ardından özellikle kombinatoryal optimizasyon problemlerinin çözümünde kullanılan arı sistemi uygulamalarına değinilmiştir. Ayrıca bu bölümde arıların yiyecek arama davranışları hakkında detaylar verilerek tez çalışmasında temel alınan AA ve YAK algoritması açıklanmıştır.

Üçüncü bölümde genellikle sürekli optimizasyon problemlerine uygulanmış olan AA ve YAK algoritmasının karmaşık tamsayılı optimizasyon problemlerindeki performansını inceleyebilmek için NP-zor bir problem olan GAP'nin çözümünde kullanılmasına yer verilmiştir. Ayrıca GAP'nin AA ile çözümünde kullanılan komşuluk yapılarının algoritma üzerindeki etkisi incelenmiştir.

Dördüncü bölümde yine zor bir kombinatoryal optimizasyon problemi olan ÇTMHDP'nin genel yapısı ve bilimsel yazındaki çözüm yaklaşımları incelendikten sonra probleme ait farklı kısıtlar altında ilgili algoritmalarla elde edilen geniş deneysel çalışma sonuçları sunulmuştur. Diğer taraftan AA, bulanık çok amaçlı ÇTMHDP'nin çözümü için de kullanılmış ve bulanık amaçlar farklı teknikler altında incelenerek bu tekniklerin algoritma performansı üzerindeki etkisi incelenmiştir.

Son olarak sonuç ve öneriler bölümünde, tez çalışmasından elde edilen sonuçlar ve öneriler tartışılmıştır.

2. BÖLÜM
SÜRÜ ZEKÂSINA DAYALI OPTİMİZASYON ALGORİTMALARI

2.1. Giriş

Bu bölüm, sürü zekâsı optimizasyon tekniklerinden biri olan arı sisteminin kombinatoryal optimizasyon problemlerinin çözümünde kullanımına temel oluşturması için sürü zekâsı ve arı sistemine ayrılmıştır. İlk olarak sürü zekâsından ve bu kavrama dayalı optimizasyon algoritmalarından bahsedildikten sonra doğadaki gerçek arı davranışları detaylandırılarak bilimsel yazında arı sistemi ile bağlantılı mühendislik çalışmaları incelenecektir. Ardından özellikle kombinatoryal optimizasyon problemlerinin çözümünde kullanılan arı sistemi uygulamalarına değinilecektir. Ayrıca bu bölümde arıların yiyecek arama davranışları hakkında detaylar verilerek tez çalışmasında temel alınan AA ve YAK algoritması açıklanacaktır.

2.2. Sürü Zekâsına Dayalı Optimizasyon Algoritmaları

Bilimsel yazında kombinatoryal ve sayısal optimizasyon problemlerinin çözümü için geliştirilen birçok modern sezgisel algoritma bulunmaktadır. Bu algoritmalar ele alınan kritere bağlı olarak popülasyon tabanlı, tekrarlı, stokastik ve deterministik gibi farklı gruplara ayrılabilir. Popülasyon tabanlı algoritmalar, tek bir çözüm yerine bir çözümler kümesini temel almakta; tekrarlamalı algoritmalar ise çözüme çoklu iterasyonlar kullanarak yaklaşmaktadır. Eğer kullanılan algoritma bir çözümü geliştirmek için olasılık tabanlı bir kural kullanıyorsa stokastik, kesin bir kural kullanıyorsa deterministik olarak adlandırılmaktadır. Diğer taraftan popülasyon tabanlı algoritmalar evrimsel algoritmalar ve sürü zekâsına dayalı algoritmalar olmak üzere iki önemli sınıfa ayrılmaktadır [8]. Evrimsel algoritmalar, organizma popülasyonlarının genetik kalıtım ve en iyinin yaşaması gibi biyolojik işlemlerden esinlendiği stokastik optimizasyon algoritmalarını içerir. Genetik algoritma, genetik programlama, evrimsel strateji,

evrimsel programlama ve diferansiyel gelişim algoritmaları popüler evrimsel algoritmalardandır.

Tez çalışmasında temel alının sürü zekâsı ise son yıllarda birçok araştırmacı bilim adamı için bir ilgi alanı haline gelmiştir. Sürü zekâsı, yetenekleri kısıtlı canlıların bir takım haline geldiklerinde üstün yetenek ve yönetim gerektiren davranışlar sergilemesi olarak açıklanabilir. Bonabeau ve ark. [9] ise sürü zekâsını "sosyal böcek kolonilerinin ya da diğer hayvan topluluklarının kolektif davranışlarından esinlenen herhangi bir algoritma ya da dağıtımlı problem çözme mekanizması tasarlama girişimi" olarak tanımlamaktadır. Bonabeau ve ark. [9] bakış açılarını sadece termit, arı, yaban arısı ve diğer karınca türleri gibi sosyal böceklere yoğunlaştırmışlardır. Ancak sürü kelimesi genel olarak birbiriyle ilişkili ajan ya da bireyler topluluğu şeklinde kullanılmaktadır. Ajanlar arasındaki bu ilişkiler doğal ortamda yiyecek bulma, yuva inşa etme ya da genişletme, ajanlar arasındaki etkili işbölümü, yavru besleme ve dış etkenlere cevap verebilme şeklinde ortaya çıkmaktadır [9]. Sürü yapısına örnek olarak, kovanları etrafında hareket halinde olan arılar, bireysel ajanları karıncalar olan karınca kolonisi, kuş sürüleri, hücre ve molekül sürüsü olarak düşünülebilen bağışıklık sistemi verilebilir. Doğadan esinlenen sezgisel yöntemlerin temel özellikleri, doğada bulunan bir olguyu modellemeleri, stokastik bir yapıya sahip olmaları, çok ajanlı sistemlerde genellikle paralel bir yapı bulunması ve geri besleme bilgisinin kullanılmasıdır [10]. Kendi kendine örgütlenen ve içinde bulunduğu çevreye uyum sağlayabilen dağıtımlı problem çözme sistemleri gibi zeki sürü davranışları edinebilmek için kendi kendine örgütlenme ve iş bölümü olmak üzere iki temel özellik gerekli ve yeterlidir [9].

- Kendi kendine örgütlenme, sistemin düşük seviye bileşenleri arasındaki etkileşim sayesinde global seviyede yapılar olarak karşımıza çıkan dinamik mekanizmalar kümesi olarak tanımlanabilir. Bu mekanizmalar sistemin bileşenleri arasındaki etkileşim için temel kuralları oluşturur. Bu kurallar ise etkileşimin global örüntüyle hiçbir ilişkisi olmadan tamamen yerel bilgiye dayanarak gerçekleştirilmesini sağlar. Diğer bir deyişle, sürüyü oluşturan ajanlar değişen şartlara göre kendilerini örgütleyebilirler, yaptıkları işin cinsini değiştirme ve duruma göre kendilerine uygun işi tayin edebilme yeteneklerine sahiptirler. Bonabeau ve ark. [9] kendi kendine örgütlenmenin 4 temel özelliğini

pozitif geri besleme, negatif geri besleme, sürekli değişim ve çoklu etkileşim şeklinde belirlemişlerdir.

- Pozitif geri besleme, uygun yapıların yaratılmasını sağlayan basit bir ampirik davranıştır. Bazı karınca türlerinde iz bırakma ve iz takip etme ya da arılarda dans etme gibi iyileştirme ve güçlendirme davranışları pozitif geri beslemeye örnek olarak verilebilir.
- Negatif geri besleme ise kolektif örüntüyü dengede tutmaya yardım etmektedir. Yiyecek kaynağının tükenmesi, kalabalıklaşması ya da yiyecek kaynağında rekabet şeklinde ortaya çıkabilen doygunluğu engelleyebilmek için negatif geri besleme mekanizmasına ihtiyaç duyulmaktadır.
- Rastgele yürüyüşler, hatalar, sürü bireyleri arasında rastgele iş değişimi gibi sürekli değişimler, yaratıcılık ve yenilik için çok önemlidir. Gelişen yapılarda rastgelelik, yeni çözümlerin keşfini sağladığı için büyük öneme sahiptir.
- Kendi kendine örgütlenme genel olarak hem kendi hareketlerinin hem de diğer bireylerin hareketlerinin sonuçlarını düşük seviyede kullanabilen karşılıklı etkileşime sahip bireyler gerektirir.

- Zeki sürü davranışlarının diğer bir özelliği olan işbölümü, belirli özelliklere sahip bireyler tarafından aynı anda gerçekleştirilen farklı işlerin varlığı şeklinde tanımlanmaktadır. İşbirliği içindeki bu uzmanlaşmış bireylerin aynı andaki iş performansının, uzmanlaşmamış bireylerin sıralı iş performansından daha etkin olduğuna inanılmaktadır. İş bölümü aynı zamanda sürünün araştırma uzayındaki değişen şartlara cevap verebilmesini de sağlamaktadır.

Sürü zekâsı temel olarak basit ajanlar arasındaki dolaylı ve dolaysız etkileşimi, değişen çevre şartlarına uyumu gösteren esnekliği ve bazı bireylerin görevlerini gerçekleştirememeleri durumunda kolonideki işlerin devam edebilmesi kabiliyetini temsil eden sağlamlığı vurgulamaktadır. Koloniyi oluşturan ajanlar çok basit kuralları takip etmekte ve bireysel ajanların nasıl hareket edeceklerini belirleyen merkezi bir kontrol yapısı olmamasına rağmen ajanlar arasındaki yerel ve belli bir seviyedeki rastgele etkileşim, zeki global davranışın oluşumuna liderlik etmektedir. Sürü zekâsı algoritmalarının kombinatoryal optimizasyon, iletişim ağları ve robotikteki başarılı

uygulama sayısı üssel olarak artmaktadır [11]. Sürü zekâsına dayalı algoritmalardan en yaygın olarak kullanılan Parçacık Sürü Optimizasyonu (PSO) ve Karınca Koloni Optimizasyonu (KKO) algoritması izleyen bölümlerde daha detaylı olarak verilmiştir.

2.2.1. Parçacık Sürü Optimizasyonu Algoritması

PSO algoritması kuş sürülerinin davranışlarından esinlenerek Kennedy ve Eberhart [12] tarafından geliştirilmiş popülasyon tabanlı stokastik bir optimizasyon tekniğidir. PSO algoritmasının klasik optimizasyon tekniklerinden en önemli farkı türev bilgisine ihtiyaç duymamasıdır. Bu özellik birçok problemin çözümü için gerekli olan karmaşık işlem yükünün hafifletilmesini sağlamaktadır. Algoritma özellikle çok boyutlu uzayda doğrusal olmayan fonksiyonların optimizasyonunda üstün performans göstermekte ve fonksiyon optimizasyonu, bulanık sistem kontrolü, yapay sinir ağı eğitimi gibi birçok alanda başarıyla uygulanabilmektedir.

PSO algoritması temelde kuş sürülerinin yiyecek aramaları ve yiyecek bulduktan sonra birbirlerinden faydalanarak sürü halinde yiyeceğe doğru yönelmelerini modellemektedir. Bu modeldeki en önemli nokta yiyeceği ilk bulan kuşun diğerlerine rehberlik etmesi ve bireyler arasında sosyal bilgi paylaşımının var olmasıdır. Kuşların yerini bilmedikleri bir yiyeceği aramaları, bir probleme çözüm aramaya; parçacık olarak adlandırılan her bireyin pozisyonu, bir çözüme; bireyin hızı, değişime ve popülasyon ise kuş sürüsüne benzetilmektedir.

PSO algoritması rastgele çözümler içeren bir popülasyonla başlatılmakta ve bu çözümler güncellenerek optimum çözüm araştırılmaktadır. Parçacık olarak adlandırılan aday çözümler, kuşların yiyecek ararken yiyeceğe en yakın kuşu takip etmeleri gibi, o andaki optimum parçacığı izleyerek problem uzayında dolaşmaktadırlar. Parçacık hareket ettiğinde, kendi koordinatlarını bir fonksiyona göndermekte, böylece parçacığın uygunluk değeri yani yiyeceğe ne kadar uzaklıkta olduğu ölçülmektedir. Bir parçacığa ait hız ve yön bilgisinin her seferinde nasıl değişeceği, kendi koordinatları ile komşu parçacıkların en iyi koordinatlarının birleşimi ile belirlenmektedir. Ajanların kendi tecrübelerine ve komşu ajanların tecrübelerine göre hareket etmeleriyle ajanlar arasında bilgi paylaşımı sağlanmaktadır [11].

2.2.2. Karınca Koloni Optimizasyonu Algoritması

KKO algoritması, yapay karınca adı verilen her bir ajan davranışının gerçek karınca davranışlarına benzetildiği çok ajanlı bir sistemdir. Sürü zekâsına dayalı algoritmaların en başarılı örneklerinden biri olan KKO algoritması klasik Gezgin Satıcı Problemi, Karesel Atama Problemi, Genelleştirilmiş Atama Problemi, Araç Rotalama Problemi, Çoklu Sırt Çantası Problemi gibi çok çeşitli problemlerde uygulama alanı bulmuştur.

KKO algoritması, karıncaların yiyecek arama davranışından esinlenerek Dorigo [13] tarafından önerilmiş, özellikle kesikli optimizasyon problemlerinde başarılı uygulamaları olan bir yöntemdir. Birçok karınca kolonisinde karıncalar yiyecek ararken, öncelikle yuvalarının etrafında rastgele dolaşarak keşfe başlarlar. Yiyecek kaynaklarını bulduklarında, yiyeceğin kalitesi ve miktarını değerlendirdikten sonra bir kısmını yuvaya taşırlar. Bu dönüş sırasında diğer karıncaların da aynı kaynağı bulabilmeleri için yiyeceğin kalitesine ve miktarına bağlı olarak kimyasal feremon maddesini geçtikleri yolun üzerine bırakırlar. Bırakılan bu izler, diğer karıncalara rehberlik ederek belirli bir olasılıkla o yolu takip etmelerine ve kaynağı bulmalarına yardım eder. Feremon vasıtasıyla yapılan bu dolaylı iletişim, karıncaların yiyecek kaynağı ile yuva arasında en kısa yolu bulmalarını sağlar. İşte karıncaların bu davranışları KKO algoritmasının geliştirilmesinde ilham kaynağı olmuştur.

Diğer taraftan KKO algoritmasının başarılı uygulamaları arıların kombinatoryal optimizasyon problemlerinin çözümünde kullanılmasına bir basamak oluşturmuş ve son yıllarda arıların doğal davranış özellikleri çeşitli metasezgisel algoritmaların geliştirilmesine olanak sağlamıştır.

2.3. Doğadaki Gerçek Arı Davranışları

Böcekler dünyasındaki en çarpıcı mühendislik ve mimarlık bilgisine sahip olan arılar, sosyal hayatları ve aralarındaki iletişim ile diğer pek çok canlıdan ayrılmaktadır. Bir arı kolonisi, bir kraliçe, birkaç yüz erkek ve 10.000-80.000 işçi arıdan oluşur. Kraliçe arının temel görevi yumurtlamak olup üreme sadece kraliçe arı vasıtasıyla gerçekleşir. Kraliçe arı yumurtlamadan başka, koloninin bütünlüğünü ve kovandaki sistemin işleyişini sağlayan önemli maddeler de salgılar. Erkek arıların tek fonksiyonu ise kraliçeyi döllemektir; yiyecek toplama, temizlik, yavruların bakımı gibi işlevlerle

ilgilenmezler. Kovanda petek örme, yiyecek toplama, arı sütü üretme, kovan ısısını düzenleme, temizlik, savunma gibi diğer tüm işleri ise işçi arılar yapar.

Mühendislik çalışmalarında ele alınan arı davranışlarının en önemlileri yiyecek arama ve çiftleşmedir. Arı sistemi içerisinde bu tür davranışların nasıl gerçekleştiği aşağıda kısaca özetlenmiştir.

Arıların Yiyecek Arama Davranışı: Arılar çoğu zaman yiyecek bulmak için kovandan ayrılarak geniş alanları taramak zorunda kalırlar. Yeni bir yiyecek kaynağı bulan arılar, koloninin diğer üyelerine haber vermek üzere kovana geri dönerler. Arılar sağır olmalarına rağmen yiyecek kaynağının yerini, koloninin diğer üyelerine dans ederek tarif edebilirler. Yiyecek kaynağının bulunabilmesi için kaynağın kovana uzaklığı, doğrultusu, zenginliği gibi gerekli olabilecek her türlü bilgi bu dansta gizlidir. Yiyecek kaynağını keşfeden arı kovana döner ve diğer arıların dikkatini çekecek şekilde sürekli olarak belirli hareketleri tekrarlamaya başlar. Arının genel davranışlarından yiyecek kaynağı ile ilgili tüm bilgiler elde edilebilir. Örneğin nektar toplamış bir arı kovana döndüğünde sadece nektarı boşaltıp geri uçarsa bu, arının faydalandığı kaynak bilinen bir kaynaktır veya verimsizdir anlamına gelmektedir. Yiyecek kaynağı bulan arılardan elde edilen bu bilgiler doğrultusunda diğer arılar kolaylıkla yiyecek kaynağının yerini bulurlar. Yiyecek kaynağına çok fazla arı toplanması, kovanda dans eden arıların sayısı ile doğrudan bağlantılıdır. Tek bir arının dansı ile tüm kovan harekete geçmez. Öncelikle koloniden bir grup arı öncü olarak gider. Bu öncü grup uçuştan döndüğünde onlar da dans ediyorsa daha fazla arı hedefe doğru yönelir. Buldukları kaynak ne kadar iyi ise, o kadar uzun süre dans ederler ve daha fazla arı toplarlar. Böylece arıların dikkati daima en verimli yiyecek kaynağına yönelmiş olur. Diğer taraftan yiyecek kaynağını bulan arı kovana geri dönmeden önce yiyecek kaynağına özel bir koku bulaştırır. Arıların çiçekleri işaretlemeleri sayesinde, diğer arılar bu çiçeğin nektarının daha önce başka arılarca tüketildiğini anlar ve o çiçeği terk ederler. Böylece hem vakit hem de enerji kaybından kurtulurlar.

Bilimsel yazında arıların yiyecek arama davranışına dayalı biyolojik çalışmalara bakıldığında, Yonezawa ve Kikuchi [14] arıların yiyecek arama davranışını inceleyerek grup zekâsının önemini gösteren bir algoritma geliştirmişlerdir. Geliştirilen algoritmanın kolonide bir ve üç arı olduğu varsayımıyla simülasyonu yapılmıştır. Karar

verme sürecinde üç arıya sahip sistemin bir arıya sahip sisteme göre daha hızlı çalıştığı gösterilmiştir. Diğer taraftan arıların değişen çevre şartlarına kolaylıkla adapte olabilen bir yiyecek arama davranışına sahip olduğu belirtilmiştir. Schmickl ve ark. [15] ise çok ajanlı bir simülasyon platformu üzerinde arıların yiyecek arama davranışının sağlamlığını incelemişlerdir. Çevresel değişikliklerin yiyecek arama stratejisini ve etkinliğini nasıl etkilediğini araştırarak, arı kolonisinin bu konuda sağlam ve iyi bir adaptasyona sahip olduğunu belirtmişlerdir. Yiyecek arama davranışı konusunda, karınca kolonisi davranışlarına dayalı feremon tabanlı algoritmalar ile arı kolonisi davranışlarına dayalı feremon tabanlı olmayan algoritmalar arasında bir karşılaştırma çalışması gerçekleştiren Lemmens [16], elde ettiği deney sonuçlarına göre (i) feremon tabanlı olmayan algoritmaların genel anlamda daha az süre gerektirdiğini ve daha hızlı olduğunu, (ii) feremon tabanlı algoritmaların küçük boyutlu problemler için iterasyon başına daha az süre gerektirdiğini, (iii) boyut arttıkça feremon tabanlı olmayan algoritmaların feremon tabanlı algoritmalara göre daha iyi performansa sahip olduğunu belirtmiştir.

Arıların Çiftleşme Davranışı: Kraliçe arı çiftleşmek için kovandan bir grup arıyla birlikte yola çıkar. Bir süre sonra beraberindeki arılardan ayrılır ve erkek arıların toplandığı alanlara doğru tek başına uçar. Bu alana belirli bir oranda yaklaştığında erkek arıların kendisini bulmalarını sağlayan bir tür feremon salgılamaya başlar. Çiftleşme uçuşu adı verilen bu uçuş sırasında erkek arıların kraliçeyi fark etmeleri ile çiftleşme gerçekleşir. Döllenmeden sonra kraliçe arı kovana geri dönerken erkek arılar genellikle hayatlarını kaybederler. Tek bir erkek arının spermleri kraliçe arının üreme kesesini doldurmaya yetmediğinden kraliçe birden fazla erkek arıdan sperm alır. Döllenmeden sonra erkek arılardan gelen bütün spermler üreme kesesinde biriktirilir. Kraliçe, 4-5 senelik ömrü boyunca çiftleşme uçuşu sırasında edindiği bu spermleri kullanacaktır. Diğer pek çok canlıdaki üreme hücrelerinin aksine erkek arıların spermleri kraliçenin vücudunda bozulmadan senelerce muhafaza edilebilecek bir yapıya sahiptir. Kraliçe arı bu keseden kendi isteğine göre sperm bırakarak döllenmeyi düzenler. Adams ve ark. [17], Page ve ark. [18], Dietz [19], Laidlaw ve Page [20], Rinderer ve Collins [21] arıların çiftleşme davranışını biyolojik olarak inceleyen çeşitli çalışmalar sunmuşlardır.

2.4. Arı Sistemi ile Bağlantılı Mühendislik Çalışmaları

Arı davranışlarıyla ilgili biyolojik araştırmalar uzun yıllardır devam etse de bilgisayar bilimleri ile ilgili çalışmalar pek yaygın değildir. Günümüze kadar yapılmış çalışmalar; yiyecek arama davranışına dayalı, çiftleşme davranışına dayalı ve kombinatoryal optimizasyon problemleri uygulamaları olmak üzere 3 grup altında incelenmiştir.

2.4.1. Yiyecek Arama Davranışına Dayalı Çalışmalar

Sato ve Hagiwara [22] genetik algoritma üzerinde arıların yiyecek arama davranışına dayalı değişiklikler yaparak Arı Sistemi algoritmasını oluşturmuşlardır. Bir arı kolonisinde her arı bireysel olarak yiyecek aramakta, yiyecek kaynağını bulduktan sonra salınım dansıyla bilgi paylaşımında bulunmakta ve yine bireysel olarak yeni bir yiyecek kaynağı aramaya devam etmektedir. Benzer olarak önerilen algoritmada da önce global arama yapılmaktadır. Bu amaçla basit genetik algoritma ile daha yüksek uygunluk değerine sahip süper kromozomlar elde edilmektedir. Daha sonra kromozomların çoğu yoğunlaştırılmış çaprazlama ile süper kromozomların bilgilerini elde etmekte ve çoklu popülasyon kullanılarak daha yoğun arama yapılabilmektedir. Geleneksel çaprazlamada her çift rastgele seçilmekte iken, yoğunlaştırılmış çaprazlamada bütün kromozomlar süper kromozomla bir çift oluşturmaktadır. Ayrıca Arı Sistemi algoritmasının yerel arama kabiliyetini artırmak için sözde-simpleks yöntemi de algoritmaya dâhil edilmiştir. Eğer bir döngü boyunca elde edilen çözüm tatmin edici değilse global arama tekrarlanmaktadır. Bilindiği gibi genetik algoritma iyi bir global arama kabiliyetine sahip olsa da yerel arama kabiliyeti yeterli değildir. Ancak Arı Sistemi ile yerel minimuma düşme olasılığı azalmaktadır. Çünkü algoritmanın amacı genetik algoritmanın global arama kabiliyetini azaltmadan yerel arama kabiliyetini geliştirmektir. Yapılan testlerde Arı Sistemi algoritması geleneksel genetik algoritma ile karşılaştırılmış ve özellikle yüksek derecede karmaşık, çok değişkenli fonksiyonlarda önerilen algoritmanın daha iyi performans gösterdiği görülmüştür.

Sürekli fonksiyon uygulamaları açısından arıların yiyecek arama davranışını temel alan çalışmalara bakıldığında, Yang [23] özellikle fonksiyon optimizasyonu problemlerinde etkin olan Sanal Arı algoritmasını geliştirmiştir. Önerilen algoritma genetik algoritmaya benzese de bağımsız çok sayıda arının paralel çalışmasından dolayı daha etkindir. Deneysel çalışma sonuçları, önerilen algoritmanın genetik algoritmadan daha etkin

olduğunu göstermiştir. Diğer taraftan tez çalışmasında temel alınan YAK algoritması ve AA sırasıyla Karaboğa [7] ve Pham ve ark. [6] tarafından önerilmiş olup arıların yiyecek arama davranışını modelleyen sürü zekâsına dayalı metasezgisel algoritmalardır. YAK algoritması ve AA özellikle sayısal optimizasyon problemlerinin çözümü için geliştirilse de sonraki yıllarda hem fonksiyonel hem de kombinatoryal optimizasyon problemlerinin çözümünde kullanılmıştır. YAK algoritması ve AA'nın ve bilimsel yazındaki bu algoritmalara dayalı çalışmaların detayları sırasıyla 2.6 ve 2.7 bölümlerinde verilecektir.

2.4.2. Çiftleşme Davranışına Dayalı Çalışmalar

Abbass [24] arıların çiftleşme davranışından esinlenerek Arıların Çiftleşme Optimizasyonu (AÇO) adı verilen yeni bir arama algoritması geliştirmiştir. Gerçek hayatta bir çiftleşme uçuşu kraliçenin dansıyla başlamakta ve erkek arılar çiftleşmek için kraliçe arıyı takip etmektedir. Her çiftleşmede spermler koloninin genetik havuzunu oluşturmak için kraliçe arının üreme kesesine alınarak birleştirilmektedir. Kraliçe arı ise bu spermlerin rastgele bir karışımını seçerek yumurtaları döllemektedir. Benzer olarak AÇO algoritmasındaki çiftleşme uçuşu, durum uzayındaki bir geçiş kümesi olarak düşünülebilir ve kraliçe arı farklı durumlar arasında hareket ederek her bir durumda olasılıklı olarak erkek arıyla çiftleşir. Kraliçe çiftleşme uçuşunun başındaysa yani hızı yüksekse ya da erkek arının uygunluk değeri kraliçeninki kadar iyiyse çiftleşme olasılığı daha yüksek olacaktır.

AÇO algoritması kraliçeye ait çözümün rastgele oluşturulmasıyla başlar, bu çözüm işçi arılar tarafından sezgisel bir yöntemle geliştirilir ve çiftleşme uçuşları gerçekleştirilir. Her bir çiftleşme uçuşunda kraliçenin enerjisi ve hızı başlangıç durumuna getirilmektedir. Daha sonra kraliçe arı kendi hızına bağlı olarak farklı çözümler arasında hareket ederek erkek arılarla çiftleşmektedir. Eğer bir erkek arı kraliçe arıyla çiftleşirse (erkek arı olasılıklı karar kuralını geçerse), erkek arının spermi kraliçe arının üreme kesesine alınır (kısmî çözümler listesi) ve kraliçe arının hızı ve enerjisi azaltılır. Kraliçe arı çiftleşme uçuşunu tamamladıktan sonra kovana döner ve spermlerden birini rastgele seçerek çaprazlama ve mutasyon işlemlerini gerçekleştirir. İşçi arı ise yavru arının geliştirilmesinden sorumludur ve işçi arı sayısı algoritmada kullanılan sezgisel yöntem sayısını göstermektedir. Eğer uygunluk değeri en iyi olan yavru arının çözümü kraliçe

arınınkinden daha iyiyse, kraliçe arı çözümü bu çözümle değiştirilir. Kalan yavru arılar ise öldürülür ve yeni çiftleşme uçuşu başlar.

Sonraki birçok çalışmaya temel oluşturan AÇO algoritmasının bu hâlinde, koloninin bir kraliçe ve bir işçi arıdan oluştuğu varsayılmaktadır. Önerilen algoritma, tek bir çiftleşme uçuşunda gerçekleştirilebilecek maksimum çiftleşme sayısını gösteren kraliçe arının üreme kesesi büyüklüğü, kraliçe tarafından üretilecek yavru arı sayısı ve yerel aramanın derinliğini belirleyen yavru arının geliştirilmesi için harcanacak süre olmak üzere üç parametreye sahiptir.

Önerilen algoritmanın geniş uygulama alanı bulduğu Genel Kısıt Sağlanabilirlik Problemi, değişkenlerin alanları dâhilinde bir kısıtlar kümesini sağlayacak değişkenler kümesinin belirlenmesidir. Sağlanabilirlik Problemi (SP) ise Genel Kısıt Sağlanabilirlik Problemi'nin özel bir türü olup, bu problemde her bir değişkenin alanı doğru ya da yanlış olarak nitelendirilmektedir. 3-SP ise Sağlanabilirlik Problemi'nin özel bir türü olup bu problem çeşidinde her kısıt üç değişken içermektedir. AÇO algoritmasının performansı 3-SP üzerinde test edilmiş ve sonuçlar algoritmanın oldukça başarılı olduğunu göstermiştir. Abbass [25] arı kolonisinin bir kraliçe arı ve birden fazla işçi arıya, Abbass [26] ise birden fazla kraliçe arı ve bir grup işçi arıya sahip olduğu varsayımıyla AÇO algoritmasını güncellemiştir. Geliştirilen algoritmalar 3-SP üzerinde test edilmiş ve en iyi sonuçların düşük koloni boyutu ve ortalama üreme kesesi boyutu ile elde edildiği görülmüştür.

Teo ve Abbass [27] çözüm uzayını daha iyi bölgelere taşıyacak geleneksel tavlama yaklaşımını kullanarak AÇO algoritmasını değiştirmişlerdir. Orijinal AÇO algoritmasındaki gibi bütün yörüngelerin kabulünün aksine, ancak daha iyi bir çözüm uzayına hareket gerçekleştirilebiliyorsa kabul kararı alınmakta, daha kötü bir çözüm uzayına ancak kraliçe arının uygunluk değerine bağlı bir fonksiyonla olasılıklı olarak geçilmektedir. 3-SP üzerinde yapılan deneysel çalışmalar sonucunda yeni tavlama fonksiyonuyla daha fazla iyileşme sağlandığı görülmüştür.

AÇO algoritmasındaki karar verme sürecine biyolojik açıdan bakıldığında erkek arılar genellikle başka kolonilerden geldikleri için kraliçe arıdan bağımsız oldukları görülmektedir. Bu sebeple çözümler arasındaki geçişlerin erkek arının uygunluk değerine bağlı olması şeklinde tekrar düzenlenen AÇO algoritması Teo ve Abbass [28]

tarafından önerilmiş olup 3-SP üzerinde önceki AÇO algoritmalarıyla karşılaştırılmıştır. Yeni AÇO algoritması ile daha iyi sonuçlar elde edildiği hatta algoritmanın önceki hâlleriyle çözüm bulunamayan problemlere çözümler bulunabildiği tespit edilmiştir.

Chang [29] AÇO algoritmasının kombinatoryal optimizasyon problemlerinin çözümündeki etkinliğini teorik açıdan ortaya koyarak algoritmanın global optimuma yakınsadığını ispatlamıştır. Ayrıca AÇO algoritmasının benzetimli tavlama ile genetik algoritmanın melez bir birleşimi olarak düşünülebileceğini belirtmiştir. Benzetimli tavlama kraliçe arının çiftleşme uçuşundaki erkek arı spermlerinin üreme kesesine alınmasına karşılık gelirken, genetik algoritma bazı farklarla yavru arı üretimi ve gelişimi adımlarına karşılık gelmektedir. Ayrıca bu çalışmada AÇO algoritması, Stokastik Dinamik Programlama Problemleri'nin çözümü için de kullanılmıştır.

Bozorg Haddad ve ark. [30] AÇO algoritmasına dayanarak yüksek derecede doğrusal olmayan, kısıtlı ve kısıtsız matematiksel modellerin çözümü için Bal Arılarının Çiftleşme Optimizasyonu (BAÇO) algoritmasını geliştirmişlerdir. BAÇO algoritmasının performansı birçok kısıtlı ve kısıtsız matematiksel optimizasyon fonksiyonunda test edilerek elde edilen sonuçlar genetik algoritma ile karşılaştırılmıştır. Sonuçlar BAÇO algoritmasının az bir farkla daha üstün performans sağladığını göstermiştir. Ayrıca geliştirilen algoritma, tek su deposunda optimum operasyon politikasını geliştirmek için de kullanılmış ve yine oldukça iyi sonuçlar elde edilmiştir. Afshar ve ark. [31] ise BAÇO algoritmasını sürekli optimizasyon problemleri için güncelleyerek doğrusal olmayan, sürekli ve kısıtlı Tek Su Deposu Problemi için bir uygulama gerçekleştirmişlerdir. LINGO 8.0 NLP ile elde edilen global optimum değerlerle yapılan karşılaştırmada algoritmanın optimum değere oldukça hızlı yakınsadığı görülmüştür. Arefi ve ark. [32] dağıtım ağlarındaki harmonik durum değişkenlerini tahmin edebilmek için BAÇO algoritmasına dayalı bir algoritma önermişlerdir. Simülasyon sonuçları önerilen algoritmanın hız ve doğruluk açısından ağırlıklı en küçük kareler yöntemi, genetik algoritma ve tabu arama algoritmalarına göre çok daha etkin olduğunu göstermiştir. Marinakis ve Marinaki [33] Olasılıklı Gezgin Satıcı Problemi için Açgözlü Rastgele Uyarlanır Arama Prosedürü (ARUAP), genişletilmiş komşuluk arama stratejisi ve BAÇO algoritmasına dayalı melez bir yapı önermişlerdir. Parçacık sürü optimizasyonu, klasik ARUAP algoritması, tabu arama ve karınca koloni

optimizasyonu algoritmaları ile yapılan karşılaştırmalar, önerilen algoritmanın iyi bir performansa sahip olduğunu göstermiştir.

AÇO ve BAÇO algoritmalarının uygulama çalışmalarına bakıldığında ise Bozorg Haddad ve Afshar [34] AÇO algoritmasını Su Kaynakları Yönetimi Problemi'ne; Fathian ve ark. [35] BAÇO algoritmasını bir veri madenciliği tekniği olan kümelemeye ve Horng [36] BAÇO algoritmasını dijital görüntü sıkıştırma uygulamalarında güçlü bir teknik olan vektör nicelemeye uygulamışlardır.

Diğer taraftan bilimsel yazında, arıların çiftleşme davranışını genetik algoritmanın performansını iyileştirme amaçlı olarak kullanan çalışmalar da mevcuttur. Jung [37] genetik algoritmanın performansını iyileştirebilmek için Kraliçe Arı Evrim (KAE) algoritmasını geliştirmiştir. KAE algoritmasında kraliçe arı, rulet tekerleği seçme yöntemi dışında bir seçme algoritmasıyla belirlenen arılar tarafından melezlenmektedir. Ancak bu yaklaşımla erken yakınsama olasılığı artmaktadır. Bu olasılığı azaltmak için bütün bireylerin düşük bir oranla mutasyona uğraması yerine bazı bireyler yüksek derecelerde mutasyona uğratılmıştır. Önerilen algoritma, bir kombinatoryal ve iki fonksiyon optimizasyon problemi üzerinde test edilmiş ve yapılan deneyler önerilen algoritmanın genetik algoritmayı daha hızlı global optimuma götürdüğünü göstermiştir. Azeem ve Saad [38] KAE algoritması üzerinde bazı değişiklikler yapmışlardır. Geliştirilen algoritmaya göre eğer kraliçe arının uygunluk değerine çok yakın ya da daha iyi uygunluk değerine sahip bir çözüm elde edilmişse, bu çözüm başka bir havuza alınmakta ve o havuzun kraliçe arısı olmaktadır. Diğer bir fark ise çaprazlama operatörü olup orijinal algoritma her bir genin belli bir olasılık dâhilinde çaprazlandığı bir-biçimli çaprazlama kullanırken geliştirilen algoritmada, ağırlıkların her bir genin popülasyondaki diğer bireylere olan benzerliğine göre belirlendiği ağırlıklandırılmış bir-biçimli çaprazlama kullanılmaktadır. Bu tip çaprazlama ile genetik algoritma daha fazla yeni durum uzayını araştırabilmektedir. Önerilen algoritma Bulanık Bilgi Kontrolü'nde ölçeklendirme faktörünün ayarlanması için doğrusal olmayan iki örnek üzerinde test edilmiştir. Sonuçlar geliştirilen algoritmanın geleneksel kontrol algoritmalarından çok daha iyi sonuçlar verdiğini göstermiştir. Qin ve ark. [39] ise KAE algoritmasını Ekonomik Güç Gönderme Problemi'ne uygulamışlar ve KAE algoritmasının geleneksel genetik algoritmadan daha hızlı ve sağlam olduğunu belirtmişlerdir.

Karcı [40] ise genetik algoritmanın performansını artırmak için ‘Arı Çaprazlaması’ adı verilen 3 yeni çaprazlama yaklaşımı önermiştir. Kraliçe arı diğer arılarla bir birleşme gerçekleştirdiğinden çaprazlama yapılacak ilk kromozom kraliçe arıya aittir. Birinci çaprazlama yaklaşımında en iyi uygunluk değerine sahip birey sabitlenmekte ve kalan kromozomlar her nesilde en az bir kere bu kromozomla çaprazlanmaktadır. İkinci yaklaşımda en kötü uygunluk değerine sahip birey sabitlenmekte ve yine kalan kromozomlar her nesilde en az bir kere bu kromozomla çaprazlanmaktadır. Üçüncü yaklaşımda ise popülasyondaki bireyler uygunluk değerine göre sıralanmakta ve ilk nesilde sabitlenen birey, bu sıradaki ilk kromozom, ikinci nesilde ikinci kromozom vb. olmaktadır. Önerilen kromozom tipleri bir-biçimli çaprazlama ile karşılaştırılmıştır. Yapılan testlerin çoğunda arı çaprazlamasının daha az iterasyon sayısına sahip olduğu görülmüş ve en kötü çözümlerin bir-biçimli çaprazlama ile elde edildiği belirtilmiştir. Diğer taraftan bir-biçimli çaprazlama ile popülasyondaki farklılık kısa bir süre içinde kaybedilmiş, arı çaprazlamasıyla ise farklılık daha uzun süre korunabilmiştir.

2.4.3. Kombinatoryal Optimizasyon Problemlerinde Arı Sistemi Uygulamaları

Tez çalışmasının temel konusu olan kombinatoryal optimizasyon problemlerinde arı sistemi uygulamaları bu bölümde incelenmiştir. Günümüze kadar yapılan uygulama çalışmaları ulaştırma problemleri, telekomünikasyon uygulamaları, çizelgeleme problemi, ekonomik güç gönderme problemi, navigasyon problemi, sağlanabilirlik problemi, veri madenciliği, kaynak paylaştırma problemi ve dalga boyu atama ve rotalama problemi şeklinde karşımıza çıkmaktadır. Belirtilen problemlerin çözümü için önerilen arı sistemine dayalı çözüm yaklaşımları detaylı bir şekilde izleyen bölümlerde verilmiştir.

2.4.3.1. Ulaştırma Problemleri

Lucic [41] ulaştırma problemlerinin çözümü için Arı Sistemi (AS) algoritmasını önererek Gezgin Satıcı Problemi (GSP) ve Stokastik Araç Rotalama Problemi (SARP) üzerinde çalışmıştır.

GSP her noktadan sadece bir kere geçecek minimum uzunluğa sahip turu bulmayı amaçlayan NP-zor bir problemdir. Önerilen algoritma, arıların nektar topladığı grafik üzerindeki noktalardan birine, kovanın yerleştirilmesiyle başlar. Yapay arılar belli bir

süre boyunca nektar toplarlar ve kovanın pozisyonu rastgele değiştirilir. Arılar yeni pozisyondan tekrar nektar toplamaya başlarlar ve kovanın yeri tekrar değiştirilir. Araştırma prosesindeki her bir iterasyon, kovanın değişen bir pozisyonunu göstermekte ve bir ya da daha fazla uygun çözüm bulunduğunda iterasyon sona ermektedir.

Yapay arılar kesikli zamana sahip bir çevrede yaşamakta, dolayısıyla her bir iterasyon belli sayıda aşamalardan oluşmaktadır. Arılar herhangi bir aşamada ziyaret edecekleri noktaları rastgele seçmektedirler. İki nokta arasındaki uzaklık arttıkça bu bağlantının bir arı tarafından seçilme olasılığı da azalmaktadır. Araştırma prosesinin başında uzaklığın etkisi az iken iterasyon sayısı arttıkça uzaklığın etkisi de artırılmaktadır. Diğer taraftan belli bir bağlantıyı ziyaret etmiş arıların toplam sayısı ne kadar büyükse o bağlantının gelecekte de seçilme olasılığı o kadar yüksek olmaktadır. Bu da kolonideki bireysel arılar arasındaki etkileşimi göstermektedir.

Bir aşama süresince arılar belli sayıda noktayı ziyaret ederek kısmî bir gezgin satıcı turu oluşturmakta ve kovana dönmektedir. Daha sonra kovanda bir karar verme prosesine katılarak, yiyecek kaynağına dönmeden önce dans ederek kovandaki diğer arıları yönlendirme, diğer arıları yönlendirmeden yiyecek kaynağına geri dönme ya da yiyecek kaynağını terk etme kararlarından birini vermektedirler. Diğer arıları yönlendirmeden yiyecek kaynağına geri dönme alternatifi, arıların sosyal böcekler olup aralarında bir etkileşim olması sebebiyle çok düşük bir olasılığa sahiptir. Bir arının aynı kısmî turu kullanması ya da onu terk etmesi ise kısmî turun uzunluğuna bağlıdır. Arının bulduğu tur ne kadar uzunsa, sonraki aşamada aynı turu kullanma olasılığı o kadar azdır. Belli bir bağlantı üzerindeki nektar miktarı, bağlantının uzunluğuyla ters orantılıdır. Herhangi bir aşamanın başında eğer bir arı aynı kısmî turu kullanmamaya karar verirse dans alanına gider ve bir olasılık fonksiyonuna göre başka bir arıyı takip eder. Bu fonksiyon kısmî turun toplam uzunluğuna ve o turu kaç arının önerdiğine bağlıdır.

Diğer taraftan gerçek hayatta bütün arılar eş zamanlı olarak yiyecek aramaya çıkmamaktadırlar. Algoritmada da her iterasyonun başında bütün arıların kovanda olduğu varsayılarak her aşamada yiyecek aramaya çıkan arı sayısı artırılmıştır. Bunlara ek olarak kovanı tekrar yerleştirmeden önce mevcut iterasyonda elde edilen çözümü geliştirmek amacıyla 2-opt ve 3-opt sezgisel algoritmaları uygulanmıştır.

AS algoritmasının performansı çeşitli GSP'leri üzerinde test edilmiş, sonuçlar 100 noktadan daha az boyuta sahip test problemlerinde AS algoritmasının optimum çözümü bulabildiğini ve hesaplama süresinin yeterince düşük olduğunu göstermiştir.

Lucic ve Teodorovic [42], Lucic ve Teodorovic [43], Lucic ve Teodorovic [44] çalışmalarında, test problemleri açısından Lucic [41] çalışmasının çeşitli uzantılarını sunmuşlardır.

Diğer taraftan SARP ise depoların ve hizmet edilecek noktaların yerleri, araç kapasiteleri kesin olarak ve hizmet edilecek noktaların talepleri yaklaşık olarak biliniyorken ulaştırma maliyetini minimize edecek rotanın bulunması olarak tanımlanmaktadır. Noktalardaki taleplerin belirsiz olması sebebiyle araçlar belli bir noktaya gidip yetersiz kapasiteden dolayı hizmet veremeyebilmektedir. Bu durumda aracın depoya geri döndüğü, yükünü boşalttığı, başarısız olduğu noktaya geri döndüğü ve planlanan tura devam ettiği varsayılmaktadır. Dolayısıyla noktalardaki talepler rastgele bir değişken olarak düşünülmekte ve gerçek talep değerleri ancak noktalara gidildiğinde bilinebilmektedir. Lucic [41] SARP'ne iyi çözümler üretebilmek için AS algoritması ile bulanık mantık yaklaşımını birleştirmiştir. Önerilen yaklaşımın temel iki adımı vardır. İlk adımda Araç Rotalama Problemi, AS algoritması kullanılarak ilk noktanın depo olarak düşünüldüğü bir GSP gibi çözülür ve genellikle orijinal problem için uygun olmayan bir çözüm elde edilir. İkinci adımda ise bir aracın rotasının ne zaman bitirileceğine ve önceki adımda bulunan çözümü ve bulanık kuralları kullanarak sıradaki aracın rotasının ne zaman başlatılacağına karar verilir. Geliştirilen model çeşitli GSP örnekleri üzerinde test edilmiş, elde edilen sonuçlar AS algoritmasıyla elde edilen en iyi çözümle karşılaştırılmış ve en iyi çözüme çok yakın sonuçlar bulunduğu görülmüştür. Lucic ve Teodorovic [45] çalışması ise Lucic [41] çalışmasında SARP için önerilen çözüm yaklaşımını içermektedir.

Teodorovic ve Dell'Orco [46] deterministik kombinatoryal problemlerin yanında belirsizlik altındaki kombinatoryal problemlerin de çözümü için AS algoritmasının daha genel bir hâli olan Arı Kolonisi Optimizasyonu (AKO) metasezgiselini önermişlerdir. Arılar arasındaki iletişimde, yaklaşık sebeplendirme ve bulanık mantık kurallarının kullanıldığı algoritmanın performansı, Tur Eşleştirme Problemi üzerinde analiz edilmiştir. Tur Eşleştirme Problemi, araç ve yolcuların toplam hareket uzunluğunun ya

da toplam gecikmenin minimize edildiği ya da araç kullanımının dengelendiği, araç ve yolcuların rota ve çizelgelerinin belirlenmesi problemidir. Önerilen yaklaşımı destekleyecek teorik sonuçlar olmasa da başlangıç sonuçları umut vericidir. Son olarak Wong ve ark. [47] ise GSP'nin çözümü için AKO algoritması ile 2-opt yerel arama sezgiselini birleştirmişlerdir.

2.4.3.2. Telekomünikasyon Uygulamaları

Nakrani ve Tovey [48] internet hizmetlerine dinamik olarak Yer Paylaştırma Problemi için Bal Arısı algoritmasını önermiştir. Önerilen algoritmada bir internet hizmet sağlayıcısındaki hizmet sağlayıcı ve HTTP istek kuyrukları, sırasıyla yiyecek arayan arılar ve yiyecek kaynakları olarak modellenmiştir. Önerilen algoritmanın sonuçları optimum yer ayırma politikasını hesaplayan bir algoritma, yer ayırma politikasını hesaplamada önceki bilgileri kullanan açgözlü bir algoritma ve mümkün bütün statik yer ayırma politikaları arasında en iyiyi bulan optimum-statik algoritmayla karşılaştırılmıştır. Sonuçlar önerilen algoritmanın statik ya da açgözlü algoritmalardan daha iyi olduğunu göstermiştir. Wedde ve ark. [49] ise arıların dans dili ve yiyecek arama davranışlarından etkilenerek Arı Kovanı adı verilen, telekomünikasyon ağında rotalama için hata toleranslı, uyarlanır ve sağlam bir rotalama protokolü sunmuşlardır.

2.4.3.3. Çizelgeleme Problemi

Chong ve ark. [50], Nakrani ve Tovey [48] çalışmasından etkilenerek NP-zor bir problem olan Atölye Tipi Çizelgeleme Problemi'nin çözümü için arıların yiyecek arama davranışını kullanan yeni bir yaklaşım önermişlerdir. Algoritmanın performansı çeşitli test problemleri üzerinde karınca koloni optimizasyonu ve tabu arama algoritmalarıyla karşılaştırılmıştır. Test sonuçları, çözüm kalitesi ve hesaplama süresi açısından tabu aramanın diğer iki sezgiselden daha iyi performans gösterdiğini ortaya koymuştur. Diğer taraftan arı algoritması, karınca koloni optimizasyonu algoritmasından daha başarılı olup her iki yöntemin hesaplama süreleri yaklaşık olarak eşittir. Koudil ve ark. [51] ise AÇO algoritmasını Bölüntüleme ve Çizelgeleme Problemleri'nin entegrasyonuna uyarlamışlardır. Önerilen yaklaşımın test sonuçları genetik algoritmayla karşılaştırılmış ve AÇO algoritmasının çözüm kalitesi açısından iyi çözümler ürettiği ve hesaplama süresi açısından genetik algoritmadan daha iyi olduğu gösterilmiştir.

2.4.3.4. Ekonomik Güç Gönderme Problemi

Ekonomik Güç Gönderme Problemi doğrusal olmayan, kısıtlı ve karmaşık bir optimizasyon problemi olup aynı anda hem maliyeti minimize etmeyi hem de güç sistemindeki talepleri karşılamayı amaçlamış bir problemdir.

Kumar ve ark. [52] Ekonomik Güç Gönderme Problemi'nin çözümü için Arı Optimizasyon algoritmasını önermişlerdir. Önerilen yaklaşım ile umut verici sonuçlar elde edilmiş olup algoritmanın sağlamlığı ve etkinliği ispatlanmıştır. Chokpanyasuwan ve ark. [53] ise jeneratör kısıtlarına sahip Ekonomik Güç Gönderme Problemi'nin çözümü için AKO algoritmasını kullanmışlardır. Elde edilen simülasyon sonuçları benzetimli tavlama, genetik algoritma, tabu arama, parçacık sürü optimizasyonu ile karşılaştırılmış ve sonuçlar önerilen yaklaşımın daha yüksek kalitede sonuçları daha hızlı bir şekilde elde ettiğini göstermiştir.

2.4.3.5. Diğer Uygulamalar

Bilimsel yazındaki diğer kombinatoryal optimizasyon problemi uygulamaları incelendiğinde, Bianco [54] arıların navigasyon davranışından hareketle büyük boyutlu navigasyon için bir eşlemleme modeli önermektedir. Arılar yiyecek ararken kilometrelerce yol kat etmekte ve mükemmel bir hassasiyette arama yapabilmektedirler. Arıların bu davranışından etkilenerek geliştirilen modelle elde edilen test sonuçları hassas ve doğru arama yapılabildiğini göstermiştir. Drias ve ark. [55] arıların en kolay ulaşılabilir ve en zengin kaynağı bulabildikleri yiyecek arama davranışlarından esinlenerek Arı Sürüsü Optimizasyonu algoritmasını geliştirmişler ve NP-tam bir yapıya sahip Maksimum Ağırlıklandırılmış Sağlanabilirlik Problemi'ne uygulamışlardır. Benatchba ve ark. [56] ise AÇO algoritmasını bir Veri Madenciliği Problemi'ni Maksimum Sağlanabilirlik Problemi olarak ifade ederek çözmede kullanmış ve %96 sağlanabilirlik elde etmişlerdir.

Quijano ve Passino [57] Kaynak Paylaştırma Problemi için arıların yiyecek arama davranışları üzerine kurulu Arılarda Yiyecek Arama algoritmasını geliştirerek çok bölgeli sıcaklık kontrolü için dinamik kaynak paylaştırma üzerine bir mühendislik uygulaması gerçekleştirmişlerdir. Markovic ve ark. [58] ise optik ağlarda Maksimum Rotalama ve Dalga Boyu Atama Problemi'ni çözmek için Teodorovic ve Dell'Orco [46]

çalışmasında önerilen AKO algoritmasını kullanmışlardır. Maksimum Rotalama ve Dalga Boyu Atama Problemi, verilen bir optik ağda talep matrisi ve dalga boyu sayısı biliniyorken kurulacak ışık yolu sayısını maksimize etmekle ilgilenmektedir. Önerilen algoritma Avrupa Optik Ağı üzerinde test edilmiş ve sonuçlar doğrusal programlama gevşetme ve tabu arama algoritması ile karşılaştırılmıştır. Deneysel sonuçlar önerilen algoritmanın makul bir sürede karşılaştırma yapılan algoritmalardan daha iyi sonuçlar elde etmiştir.

YAK ve AA algoritmalarına ait bilimsel yazın taraması, algoritmaların detaylarının verildiği 2.6 ve 2.7 bölümlerinde yer almakta olup, bu bölümde yapılan bilimsel yazın taraması sonucu oluşturulan özet, araştırmacılar, geliştirilen algoritma ve uygulama alanları açısından Tablo 2.1'de verilmiştir.

Tablo 2.1. Arı sistemi ve uygulama alanları konusundaki çalışmaların sınıflandırılması.

		Araştırmacılar	**Algoritma**	**Uygulama Alanı**
Yiyecek arama davranışı	Genetik algoritmaya dayalı	Sato ve Hagiwara [22]	Arı Sistemi algoritması	Fonksiyon optimizasyon problemleri
	Sürekli fonksiyon uygulamaları	Yang [23]	Sanal Arı algoritması	Fonksiyon optimizasyon problemleri
		Karaboğa [7]	YAK algoritması	Fonksiyon optimizasyon problemleri
		Pham ve ark. [6]	AA	Fonksiyon optimizasyon problemleri
Çiftleşme davranışı	Çiftleşme davranışı	Abbass [24]	AÇO	3-SP
		Abbass [25]	AÇO algoritmasında değişiklik	3-SP
		Abbass [26]	AÇO algoritmasında değişiklik	3-SP
		Teo ve Abbass [27]	AÇO algoritmasında değişiklik	3-SP
		Teo ve Abbass [28]	AÇO algoritmasında değişiklik	3-SP
		Chang [29]	AÇO algoritması uygulaması	Stokastik dinamik programlama problemi
		Bozorg Haddad ve ark. [30]	BAÇO	Su kaynakları yönetimi problemi
		Afshar ve ark. [31]	BAÇO algoritmasında değişiklik	Su kaynakları yönetimi problemi
		Arefi ve ark. [32]	BAÇO algoritmasında değişiklik	Dağıtım ağlarındaki harmonik durum değişkenlerinin tahmini
		Marinakis ve Marinaki [33]	BAÇO algoritmasında değişiklik	Olasılıklı gezgin satıcı problemi
		Bozorg Haddad ve Afshar [34]	AÇO algoritması uygulaması	Su kaynakları yönetimi problemi
		Fathian ve ark. [35]	BAÇO algoritması uygulaması	Veri madenciliği
		Horng [36]	BAÇO algoritması uygulaması	Dijital görüntü sıkıştırmada vektör niceleme
	Genetik algoritmaya dayalı	Jung [37]	KAE	Kombinatoryal ve fonksiyon optimizasyon problemleri
		Azeem ve Saad [38]	KAE algoritmasında değişiklik	Bulanık bilgi kontrolünde ölçeklendirme faktörünün ayarlanması
		Qin ve ark. [39]	KAE algoritması uygulaması	Ekonomik güç gönderme problemi
		Karcı [40]	Arı Çaprazlaması	

Kombinatoryal optimizasyon uygulamaları	Ulaştırma problemi	Lucic [41]	AS algoritması	GSP ve SARP
		Lucic ve Teodorovic [42]	AS algoritması	GSP
		Lucic ve Teodorovic [43]	AS algoritması	GSP
		Lucic ve Teodorovic [44]	AS algoritması	GSP
		Lucic ve Teodorovic [45]	AS algoritması+Bulanık Mantık	SARP
		Teodorovic ve Dell'Orco [46]	AKO algoritması	Tur eşleştirme problemi
		Wong ve ark. [47]	AKO algoritması uygulaması	GSP
	Telekomünikasyon	Nakrani ve Tovey [48]	BA algoritması	Dinamik yer paylaştırma problemi
		Wedde ve ark. [49]	Arı Kovanı algoritması	Telekomünikasyon ağında rotalama
	Çizelgeleme	Chong ve ark. [50]	BA algoritmasında değişiklik	Atölye tipi çizelgeleme problemi
		Koudil ve ark. [51]	AÇO algoritması uygulaması	Bölüntüleme ve çizelgeleme problemi
	Ekonomik güç gönderme problemi	Kumar ve ark. [52]	Arı Optimizasyon algoritması	EGG problemi
		Chokpanyasuwan ve ark. [53]	AKO algoritması uygulaması	EGG problemi
	Diğer uygulamalar	Bianco [54]	Eşlemleme modeli	Navigasyon problemi
		Drias ve ark. [55]	Arı Sürüsü Optimizasyonu algoritması	Maksimum ağırlıklandırılmış SP
		Benatchba ve ark. [56]	AÇO algoritması uygulaması	Veri madenciliği problemi
		Quijano ve Passino [57]	Arılarda Yiyecek Arama algoritması	Kaynak paylaştırma problemi
		Markovic ve ark. [58]	AKO algoritması uygulaması	Dalga boyu atama ve rotalama problemi

2.5. Arıların Yiyecek Arama Davranışları

Arıların yiyecek arama davranışları, öğrenme, hatırlama ve bilgi paylaşma özellikleri sürü zekâsında en çok ilgi çeken alanlardan biridir. Arıların yiyecek arama davranışına dayalı olarak Tereshko [59] tarafından geliştirilen modele göre arı sürülerinde kolektif zekâ, 3 temel bileşen (yiyecek kaynakları, görevli arılar ve görevli olmayan arılar) ve 2 davranış türü (nektar kaynaklarından yiyecek toplama ve yiyecek kaynağını terk etme) içerir.

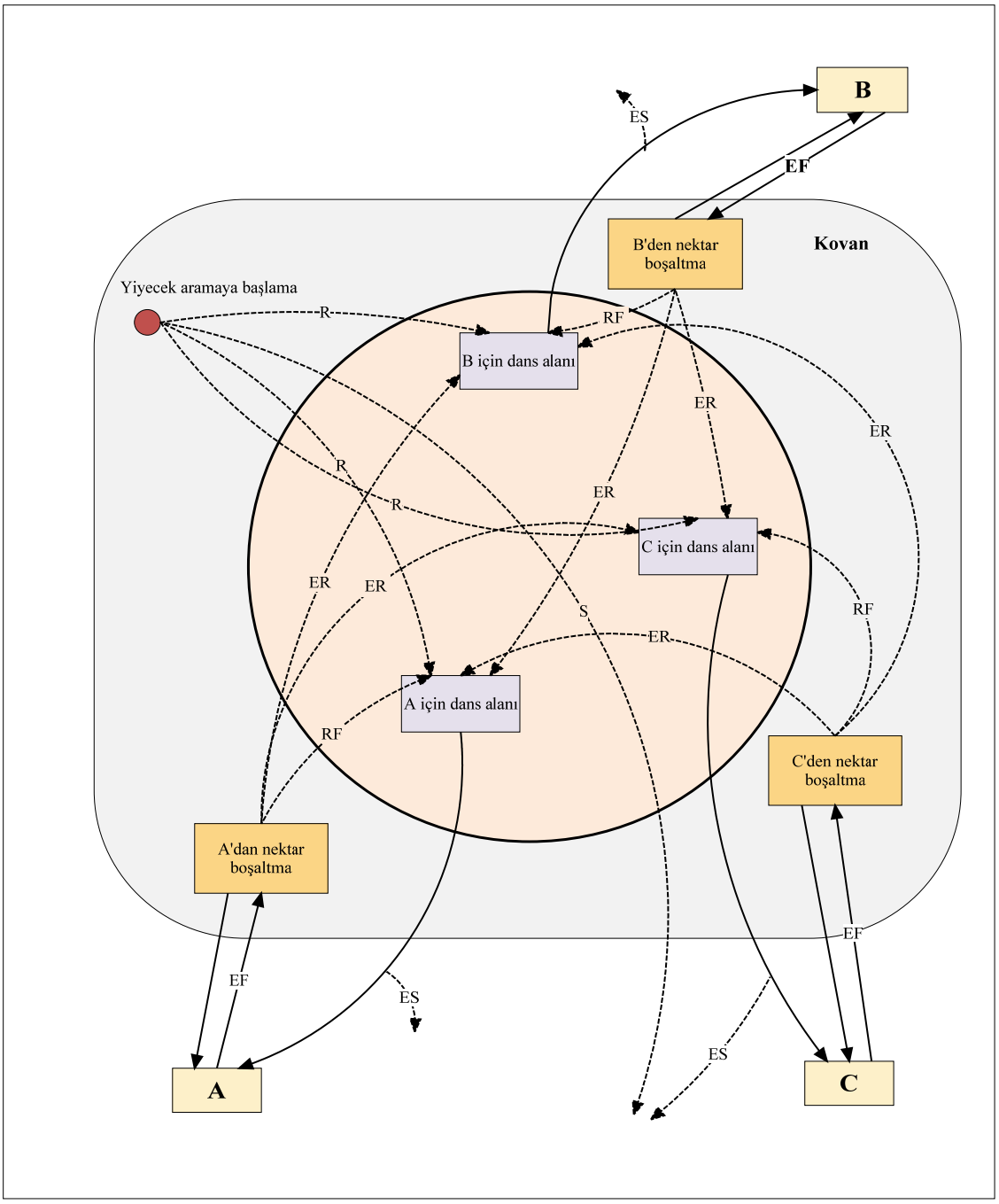
1. Yiyecek kaynakları: Yiyecek kaynağının kalitesi; kovana olan uzaklık, nektar zenginliği, enerji yoğunluğu ve bu enerjinin çıkarım kolaylığı gibi birçok etkene bağlıdır. Ancak basitlik açısından bir yiyecek kaynağının kalitesi tek bir nicelikle ifade edilmektedir.
2. Görevli arılar: Tüketmekte oldukları ya da görevlendirildikleri yiyecek kaynağıyla ilişkilendirilmişlerdir. İlgili yiyecek kaynağının kalitesi, kovana olan uzaklık ve yön bilgilerini kendileriyle birlikte taşıyarak bu bilgileri belirli bir olasılık dâhilinde diğer arılarla paylaşırlar.
3. Görevli olmayan arılar: Sürekli olarak tüketmek için yiyecek kaynağı arayan görevli olmayan arılar, kâşif ve izci arılar olmak üzere iki türdür. Kâşif arılar arı sistemi içerisinde tamamen bağımsız davranarak herhangi bir önbilgi kullanmadan yiyecek kaynağı ararken, izci arılar kovanda bekleyerek görevli arılarca paylaşılan bilgiler doğrultusunda yeni bir yiyecek kaynağı bulan arılardır. Yapılan biyolojik çalışmalara göre kovandaki kâşif arı sayısı %5-10 arasında değişmektedir [60].

Arılar arasındaki bilgi değişimi kolektif bilginin en önemli göstergesidir. Kovanın tamamı ele alındığında bütün kovanlarda genel olarak bulunan belirli bölümler mevcuttur. Bunlardan en önemlisi bilgi değişiminin gerçekleştirildiği dans alanıdır. Arılar arasında yiyecek kaynağının kalitesi hakkındaki iletişim, dans alanında gerçekleştirilmekte ve bu dansa salınım dansı adı verilmektedir. Salınım dansının uzunluğu, yiyecek kaynağının kârlılığını; güneşe olan açısı, yiyecek kaynağının yerini; dans esnasında yapılan zig-zag hareketlerinin sayısı, yiyecek kaynağına olan uzaklığı göstermektedir. Mevcut tüm zengin yiyecek kaynakları hakkındaki bu bilgi, dans alanında izci arılara sunulmakta ve izci arılar en kârlı kaynağı seçmektedirler. Daha

kârlı kaynaklar hakkında daha fazla bilgi paylaşımı olacağından izci arıların kârlı yiyecek kaynaklarını seçme olasılığı da daha yüksek olmaktadır. Arı sisteminde kendi kendine örgütlenme aşağıdaki gibi gerçekleşmektedir [7].

- Pozitif geri besleme: Yiyecek kaynaklarındaki nektar miktarı arttıkça o yiyecek kaynaklarını ziyaret eden izci arı sayısı da artar.
- Negatif geri besleme: Terk edilen yiyecek kaynaklarındaki araştırma prosesi sonlandırılır.
- Sürekli değişim: Kâşif arılar yeni yiyecek kaynakları keşfetmek için rastgele arama yaparlar.
- Çoklu etkileşim: Arılar yiyecek kaynağının pozisyonu hakkında edindikleri bilgileri dans alanında kovandaki diğer arılarla paylaşırlar.

Yiyecek toplayan arıların temel davranış özellikleri görsel olarak Şekil 2.1'de sunulmaktadır.



Şekil 2.1. Arıların yiyecek arama davranışı [41].

Şekil 2.1'de gösterildiği gibi bir arı başlangıçta kovan etrafındaki yiyecek kaynakları hakkında herhangi bir bilgisi olmadan yiyecek aramaya başlar ve görevli olmayan arı olarak adlandırılır. Bu tür bir arı için iki alternatif vardır:

- Kâşif arılar: İçsel güdüler sebebiyle, herhangi bir önbilgi olmaksızın, kovan etrafında kendiliğinden yiyecek aramaya başlayan arılardır (S).

- İzci arılar: Başka arılarca yapılan salınım dansını izleyerek yiyecek kaynakları hakkında bilgi edinen ve daha sonra bu bilgiye göre hareket ederek yiyecek kaynaklarına ulaşan arılardır (R).

Görevli olmayan arı yiyecek kaynağını bulduktan sonra yiyecek kaynağının yerini hafızasında tutarak o kaynağı tüketmeye başlar. Dolayısıyla görevli olmayan arı, bir görevli arı haline gelir. Yiyecek kaynağından bir miktar nektar alarak kovana dönüp yiyecek depolama alanına nektarı boşaltan bir arı için 4 alternatif vardır ve bu alternatiflerin gerçekleşme olasılığı yüksek oranda yiyecek kaynağının kalitesine bağlıdır.

- Nektar miktarı düşük seviyelere inmişse ya da tükenmişse, arı bu yiyecek kaynağını terk eder ve görevli olmayan bir arıya dönüşür (ES).
- Aynı yiyecek kaynağına dönmeden önce dans ederek kovandaki diğer arıları bilgilendirir (RF).
- Eğer yiyecek kaynağında yeterli miktarda nektar varsa, bilgi paylaşımında bulunmadan yiyecek kaynağını tüketmeye devam edebilir (EF).
- Tüketmekte olduğu yiyecek kaynağının kalitesinden tatmin olmayan arı, dans alanında önerilen yeni bir yiyecek kaynağını aramaya başlar (ER).

2.6. Yapay Arı Kolonisi Algoritması

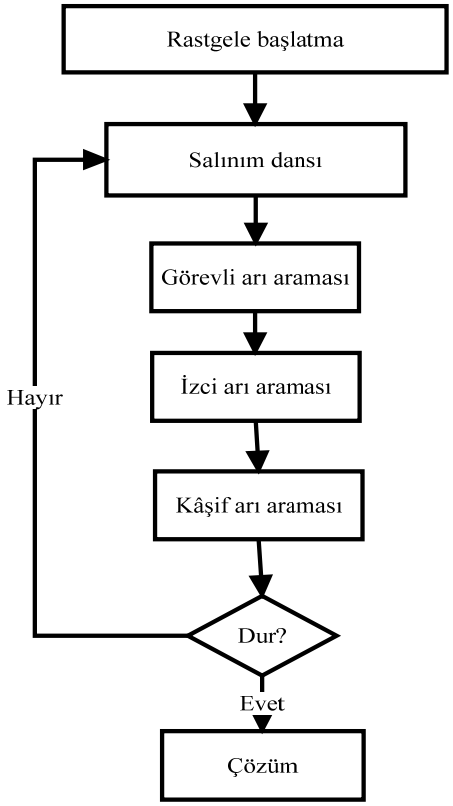
YAK algoritması arıların yiyecek arama davranışından esinlenen, çok boyutlu ve çok doruklu optimizasyon problemlerinin çözümü için geliştirilmiş, popülasyon tabanlı bir algoritmadır. Algoritmada bir yiyecek kaynağı pozisyonu, optimizasyon probleminin olası bir çözümünü temsil ederken yiyecek kaynağındaki nektar miktarı ise ilgili çözümün kalitesi yani uygunluk değeri ile ilişkilidir. Bir yapay arı kolonisi görevli, izci ve kâşif arılar olmak üzere 3 grup arı içermektedir. Kolonin ilk yarısı görevli arılardan, ikinci yarısı ise izci arılardan oluşmaktadır. Diğer taraftan görevli arı sayısı, kovan etrafındaki yiyecek kaynağı sayısına yani popülasyondaki çözüm sayısına eşittir. Kâşif arılar ise bir yiyecek kaynağındaki nektar miktarı tükendiğinde ortaya çıkmakta ve yiyecek ararken herhangi bir önbilgiye sahip olmadıklarından buldukları yiyecek kaynağı, düşük arama maliyetine ve ortalama bir kaliteye sahip olmaktadır. Ancak kâşif

arılar bazen hiç bilinmeyen zengin bir yiyecek kaynağı da keşfedebilirler. Algoritmanın temel adımları Tablo 2.2'de verilmektedir.

Tablo 2.2. YAK algoritmasının temel adımları.

1. Görevli arı sayısı kadar yiyecek kaynağını rastgele oluştur
2. *Repeat*
3. Görevli arıları yiyecek kaynaklarına yerleştir
4. Nektar miktarlarına bağlı olarak izci arıları yiyecek kaynaklarına yerleştir
5. Arılarca terk edilen yiyecek kaynaklarının tüketim prosesini sonlandır
6. Yeni yiyecek kaynaklarının keşfi için kâşif arıları yiyecek kaynaklarına gönder
7. O ana kadar bulunan en iyi yiyecek kaynağını hafızada tut
8. *Until* (istenen şartlar sağlanana kadar)

YAK algoritmasında her araştırma döngüsü, görevli arıları yiyecek kaynaklarına göndererek nektar miktarlarının değerlendirilmesi, yiyecek kaynakları hakkındaki nektar bilgisi paylaşımından sonra izci arılarca yiyecek kaynağı bölgelerinin seçimi ve nektar miktarlarının değerlendirilmesi, kâşif arıların yeni yiyecek kaynaklarına rastgele gönderilmesi olmak üzere 3 temel adımdan oluşur. YAK algoritmasına ait akış diyagramı Şekil 2.2'de sunulmaktadır.



Şekil 2.2. YAK algoritması akış diyagramı [61].

YAK algoritmasına ilişkin detaylı adımlar Tablo 2.3'te yer almakta olup kullanılan notasyonlar aşağıdaki gibi tanımlanmıştır.

P: Görevli arı sayısı (p=1,…,P)

σ^p: Görevli arı çözümü

p_p: İzci arı ataması için p. görevli arıya ait olasılık değeri

MaksIter: Maksimum iterasyon sayısı (durdurma kriteri)

$fit(\sigma^p)$: Görevli arı çözümünün uygunluk fonksiyonu değeri

limit: Gelişme olmaksızın geçirilebilecek maksimum iterasyon sayısı

Tablo 2.3. YAK algoritmasının detaylı adımları.

1. Başlangıç popülasyonunu oluştur (σ^p, p=1,…,P)
2. Popülasyonu değerlendir
3. k=1
4. *Repeat*
5. Görevli arılar için yeni çözümler üret ve değerlendir
6. Görevli arılar için açgözlü seçim prosesini uygula
7. σ^p çözümleri için olasılık değerlerini (p_p) hesapla
8. p_p değerlerine bağlı olarak seçilen σ^p çözümlerini kullanarak, izci arılar için yeni çözümler üret ve bu çözümleri değerlendir
9. İzci arılar için açgözlü seçim prosesini uygula
10. Kâşif arılar için, varsa, terk edilen çözümleri belirle ve rastgele yeni üretilen çözümle değiştir
11. O ana kadar elde edilen en iyi çözümü hafızada tut
12. k=k+1
13. *Until* (k=*MaksIter*)

Başlangıç aşamasında arılar tarafından bir yiyecek kaynağı kümesi rastgele seçilir ve nektar miktarları belirlenir. Bir yiyecek kaynağı bulmuş olan görevli arılar kovana dönerek dans alanında bekleyen arılarla, ilişkili oldukları yiyecek kaynağı hakkındaki nektar bilgisini paylaşırlar. İzci arılarla bilgi paylaşımından sonra bütün görevli arılar, ilişkili oldukları yiyecek kaynağına geri dönerek hafızalarındaki bu kaynağın komşuluğunda yeni bir yiyecek kaynağını görsel bilgi yardımıyla seçerler ve nektar miktarını değerlendirirler. Eğer komşu çözümün nektar miktarı hafızadaki çözümden daha iyiyse, hafızadaki çözüm güncellenir; aksi takdirde hafızadaki çözüm değiştirilmez. İzci arılar ise dans alanında görevli arılarca verilen nektar miktarı bilgisine bağlı bir olasılıkla (p_p) bir yiyecek kaynağı alanı seçerler. Bir yiyecek kaynağının seçilme olasılığı aşağıdaki gibi hesaplanmaktadır.

$$p_p = \frac{fit(\sigma^p)}{\sum_{n=1}^{P} fit(\sigma^n)} \tag{2.1.}$$

Yukarıdaki formülasyonda $fit(\sigma^p)$, p. pozisyondaki yiyecek kaynağının nektar miktarı ile orantılı olup p. çözümün uygunluk değerini göstermektedir. Dolayısıyla p. kaynağın

seçilme olasılığı, *p*. kaynağın nektar miktarının bütün kaynakların nektar miktarları toplamına oranlanması ile bulunur. Görüldüğü gibi bir yiyecek kaynağındaki nektar miktarı arttıkça o yiyecek kaynağının izci arılar tarafından tercih edilme olasılığı da artmaktadır. İzci arılar seçilen alana geldiklerinde, görevli arılarda olduğu gibi görsel bilgi ile hafızalarındaki kaynağın komşuluğunda yeni bir yiyecek kaynağı seçer ve nektar miktarını değerlendirirler. Diğer taraftan bir yiyecek kaynağındaki nektar miktarı tüketildiğinde, kâşif bir arı tarafından rastgele yeni bir yiyecek kaynağı belirlenir ve önceki çözümle değiştirilir. Başka bir deyişle, YAK algoritmasında yiyecek kaynağını temsil eden bir çözüm önceden belirlenmiş iterasyon sayısınca (*limit*) geliştirilememişse o yiyecek kaynağının tükendiği varsayılır ve ilgili yiyecek kaynağı görevli arı tarafından terk edilerek, görevli arı bir kâşif arıya dönüştürülür. Belirtmek gerekir ki, YAK algoritması her iterasyonda en fazla bir kâşif arı çözümüne izin vermektedir. Bahsedilen bu adımlar önceden belirlenmiş sayıda iterasyon boyunca (*MaksIter*) tekrarlanır. Sonuç olarak YAK algoritması *P*, *limit* ve *MaksIter* olmak üzere 3 kontrol parametresine sahiptir.

Algoritmanın detaylı adımlarından da anlaşıldığı gibi YAK algoritması 4 farklı seçim prosesi kullanır.

- Dans alanında verilen bilgilere göre umut verici bölgeleri keşfetmek için izci arıların kullandığı global seçim prosesi
- Hafızadaki yiyecek kaynağının komşuluğunda yeni bir yiyecek kaynağı belirlemek için görevli arılar ve izci arıların görsel bilgiye dayanarak gerçekleştirdiği yerel seçim prosesi
- Bütün arılar tarafından gerçekleştirilen, aday yiyecek kaynağının nektar miktarı mevcut kaynağın nektar miktarından daha iyiyse, arının mevcut kaynak yerine aday kaynağı hafızasına almasına dayanan açgözlü seçim prosesi
- Kâşif arılarca gerçekleştirilen rastgele seçim prosesi

Bir yapay arı kolonisinde var olan görevli, izci ve kâşif arıların algoritmadaki görevleri şu şekilde özetlenebilir.

- Görevli arılar: Problemin farklı çözümlerini oluşturan ve bu çözümlerin kalitesine ilişkin bilgi sahibi olan yapılar

- İzci arılar: Görevli arıları takip ederek komşu çözümleri oluşturup yerel aramayı gerçekleştiren yapılar

- Kâşif arılar: Yerel optimuma yakalanmayı engellemek amacıyla, görevli ve izci arılardan bağımsız yeni çözüm alanları arayan yapılar

YAK algoritmasının çalışma prensibi ve temel yapısı ilk kez Karaboğa [7] tarafından önerilmiş olup tek doruklu (tek yerel optimuma sahip) ve çok doruklu (iki ya da daha fazla yerel optimuma sahip) sayısal optimizasyon problemlerinin çözümü için kullanılmıştır. Karaboğa [7] çalışmasının bir uzantısı olarak, elde edilen deneysel çalışma sonuçları Baştürk ve Karaboğa [62] tarafından genetik algoritma ile; Karaboğa ve Baştürk [8] tarafından genetik algoritma, parçacık sürü optimizasyonu ve parçacık sürüden esinlenen evrimsel algoritma ile; Karaboğa ve Baştürk [63] tarafından diferansiyel gelişim, parçacık sürü optimizasyonu ve evrimsel algoritma ile; Karaboğa ve Akay [64] tarafından Harmoni Arama algoritması, AA ve Geliştirilmiş AA ile karşılaştırılmıştır. Bu çalışmalar neticesinde YAK algoritmasının çok boyutlu ve çok doruklu fonksiyonlarda oldukça etkin ve karşılaştırma yapılan algoritmalardan daha iyi performansa sahip olduğu görülmüştür. Diğer taraftan Karaboğa ve Baştürk [63] çalışmalarında, farklı kontrol parametresi değerleri altında YAK algoritmasının performansını incelenmişlerdir. Popülasyon büyüklüğü arttıkça algoritmanın daha iyi sonuçlar verdiği ancak belirli bir değerden sonra popülasyon büyüklüğü artırılsa da algoritmanın performansının geliştirilemediği görülmüştür. *Limit* değerinin ise tek doruklu fonksiyonlarda algoritmaya bir etkisi yokken çok doruklu fonksiyonlarda *limit* değeri yükseldikçe algoritma performansının düştüğü gözlenmiştir. Tüm bu çalışmalar sonucunda Karaboğa ve Akay [65] YAK algoritmasının performansını değerlendirebilmek için çok boyutlu, tek doruklu-çok doruklu, düzenli-düzensiz, ayrışır-ayrışamaz özelliklere sahip test problemleri üzerinde geniş bir deneysel çalışma sunmuşlardır. Elde edilen sonuçları genetik algoritma, parçacık sürü optimizasyonu algoritması, diferansiyel gelişim algoritması ve evrimsel stratejiler ile karşılaştırmışlar ve YAK algoritmasının diğer algoritmalara göre daha iyi ya da benzer sonuçlara ulaştığını belirtmişlerdir. Diğer taraftan bir optimizasyon algoritmasıyla bir problemi çözerken algoritma parametrelerinin ayarlanmasının algoritma performansı üzerinde çok önemli bir etkisi olduğu gerçeğiyle Akay ve Karaboğa [66] kontrol parametrelerinin YAK algoritmasının performansı üzerindeki etkisini araştırmışlardır. Ayrıca YAK

algoritması ile diferansiyel gelişim ve parçacık sürü optimizasyonu algoritmaları da karşılaştırılarak hangi algoritmanın parametre değerlerine karşı daha duyarlı olduğu incelenmiştir.

YAK algoritmasının kısıtsız optimizasyon problemlerindeki başarısından sonra Karaboğa ve Baştürk [67] kısıtlı optimizasyon problemlerinin çözümü için YAK algoritmasında çeşitli değişiklikler yapmışlardır. Bu değişiklikler temelde seçim mekanizmasına bağlı olup orijinal YAK algoritmasında kullanılan açgözlü seçim mekanizması yerine kısıt elde tutma mekanizması adı da verilen Deb seçim mekanizması [68] kullanılmıştır. Deb seçim mekanizmasına göre iki çözüm şu üç kritere göre karşılaştırılmaktadır: 1) Uygun bir çözüm, uygun olmayan bir çözüme göre tercih sebebidir, 2) İki uygun çözüm arasından amaç fonksiyonu değeri daha iyi olan tercih edilir, 3) İki uygun olmayan çözüm arasından kısıt ihlâli en az olan seçilir. Popülasyonu uygun çözümlerle başlatmak zaman kaybettirici bir proses olup uygun çözümler üretmek her zaman mümkün olmayabildiğinden, önerilen YAK algoritmasında başlangıç popülasyonunda uygunluk şartı yer almamaktadır. Ancak Deb kuralı sayesinde çözümler uygun alanlara yönlendirilmekte, diğer taraftan da kâşif arılar ile yeni ve uygun olmayabilen çözümler elde edilerek çeşitlilik sağlanmaktadır. Orijinal YAK algoritması ile kısıtlandırılmış problemler için geliştirilen YAK algoritmasındaki diğer bir farklılık ise kâşif arıların terk edilen yiyecek kaynaklarının yerine hemen yeni bir çözüm üretmesi yerine kâşif arıların yeni yiyecek kaynağı aramaya önceden belirlenmiş periyotlarda gönderilmesidir. Bu periyot YAK algoritmasının diğer bir kontrol parametresi olup kâşif üretme periyotu olarak adlandırılmıştır. Her bir kâşif üretme periyotunda terk edilmiş bir yiyecek kaynağı olup olmadığı kontrol edilmekte, eğer varsa kâşif arı yeni yiyecek kaynakları bulmak için gönderilmektedir. Kısıtlandırılmış optimizasyon problemleri için tekrar düzenlenen YAK algoritması doğrusal, doğrusal olmayan ve karesel amaçlara sahip çeşitli kısıtlandırılmış optimizasyon problemleri üzerinde test edilerek parçacık sürü optimizasyonu ve diferansiyel gelişim algoritması ile karşılaştırılmış ve YAK algoritmasının ortalama değerler açısından karşılaştırma yapılan algoritmalara göre daha etkin olduğu belirtilmiştir. Diğer taraftan Akay ve Karaboğa [69] tamsayılı programlama problemlerine YAK algoritmasını uygulayarak algoritmanın performansını çeşitli parçacık sürü optimizasyonu algoritmaları ve dal-sınır algoritması ile

karşılaştırmışlardır. Algoritmanın tamsayılı programlama problemleri üzerindeki performansını görebilmek için elde edilen çözümler en yakın tamsayı değerine yuvarlanmıştır. Deneysel çalışma sonuçları YAK algoritmasının tamsayılı programlama problemlerinde sağlam bir yapıya ve diğer algoritmalara göre karşılaştırılabilir ya da daha iyi performansa sahip olduğu belirtilmiştir.

YAK algoritmasının farklı uyarlamaları incelenecek olursa Quan ve Shi [70] YAK algoritmasına anlaşmalı eşlemedeki sabit nokta teoremine dayalı yeni bir arama operatörü eklemiş ve geliştirilen YAK algoritmasını çeşitli doğrusal, doğrusal olmayan ve karesel çok değişkenli test fonksiyonlarının çözümünde kullanmışlardır. Tsai ve ark. [71] ise izci arıların rulet tekerleği yöntemine göre seçildiği orijinal YAK algoritmasındaki izci arıların seçim aşamasını Newton'un evrensel çekim yasasına göre tekrar düzenlemişlerdir. Kang ve ark. [72] Ters Analiz Problemi'nin çözümü için, kısıtlandırılmamış problemlere özel olarak tasarlanmış ve gradyan bilgisi kullanmayan Nelder-Mead simpleks yöntemi [73] ile YAK algoritmasını birleştiren melez bir algoritma önermişlerdir. Deneysel sonuçlar önerilen algoritmanın ters analiz için etkin bir araç olduğunu göstermiştir. Bao ve Zeng [74] orijinal YAK algoritmasındaki izci arıların erken yakınsamaya sebep olabilen orantısal seçim stratejisi yerine popülasyonun çeşitliliğini artırabilmek ve erken yakınsamayı engelleyebilmek için sıraya bağlı seçim stratejisi, bölücü seçim stratejisi [75] ve turnuva seçim stratejilerini [76] algoritmaya uyarlamışlardır. Elde edilen simülasyon sonuçları geliştirilen YAK algoritmasının orijinal YAK algoritmasına göre daha iyi sonuçlar verdiğini göstermiştir. Mezura-Montes ve Cetina-Dominguez [77] kısıtlandırılmış sayısal optimizasyon problemlerinin çözümü için temel YAK algoritması üzerinde çeşitli değişiklikler yapmışlardır. Bunlardan ilki kâşif arı davranışlarıyla ilgili olup rastgele çözümler üretmek yerine mevcut en iyi çözümün komşuluğundaki çözümlerden faydalanmaya dayanmaktadır. Çünkü kısıtlandırılmış alanlarda çalışılırken, temel YAK algoritmasındaki kâşif arı davranışları yerel optimum çözümlerden kaçınmaya yardımcı olamamakta ve genellikle uygun olmayan çözümler üreterek araştırmayı tetikleyici özelliğe sahip olamamaktadır. YAK algoritmasına ikinci bir katkı ise Hamida ve Schoenauer [78] tarafından önerilen ve tolerans değerine bağlı dinamik bir mekanizmayı eşitlik kısıtlarına uygulamaktır. Böylece araştırmanın başlarında eşitlik kısıtları kolaylıkla sağlanabilecek, araştırmanın ileriki iterasyonlarında ise tolerans değeri azaltılarak algoritmanın uygun çözümler

bulması sağlanacaktır. Geliştirilen yöntem ile orijinal YAK algoritmasının performansının iyileştiği belirtilmiştir. Duan ve ark. [79] doğada yaşayan canlı organizmaların evrim prensibinden esinlenen ve kombinatoryal optimizasyon problemlerinin yaklaşık çözümleri için kullanılan, kuantum hesaplama prensiplerine dayalı Kuantum Evrimsel algoritması ile YAK algoritmasını birleştirilerek melez bir algoritma önermişlerdir. Deneysel çalışma sonuçları önerilen algoritmanın karmaşık optimizasyon problemlerinin çözümü için uygun ve etkin olduğunu göstermiştir. Tsai ve ark. [80] orijinal YAK algoritmasındaki izci hareketlerini evrensel yerçekimi yasasına dayandırarak interaktif YAK algoritmasını önermişlerdir. Geliştirilen algoritma, orijinal YAK ve parçacık sürü optimizasyonu algoritmaları ile karşılaştırılmış ve interaktif YAK algoritmasının diğerlerine göre daha üstün performansa sahip olduğu belirtilmiştir.

YAK algoritması, geliştiricileri tarafından yapay sinir ağlarının eğitiminde de kullanılmıştır. Karaboğa ve Öztürk [81] ileri bildirimli sinir ağlarının eğitiminde, makine öğrenmede sıkça kullanılan farklı veri kümelerinin sınıflandırılması için YAK algoritmasından faydalanmışlardır. Algoritmanın performansı gradyan tabanlı algoritmalar olan geri yayılım ve Levenberg-Marquardt algoritmaları ile popülasyon tabanlı algoritmalar olan parçacık sürü optimizasyonu, diferansiyel gelişim ve genetik algoritma ile karşılaştırılmıştır. Deneysel çalışma sonuçları YAK algoritmasının sınıflandırma problemlerinde ileri bildirimli sinir ağlarının eğitimine başarıyla uygulanabildiğini göstermiştir. YAK algoritmasının diğer uygulama alanları, radyal dağıtım sistemlerinde güç kaybının en küçüklenmesini amaçlayan ağ yapılandırması [82], dolaysız doğrusal transformasyon yöntemi [83], dijital IIR filtre tasarımı [84], yaprak-kısıtlı minimum kapsayan ağaç problemi [85], maksimum mekanik işleme hızına erişme amacına sahip optimum proses parametreleri kombinasyonunu bulma problemi [86], radyal tabanlı fonksiyon ağlarının eğitimi [87], gömülü riske sahip tedarik zinciri maliyetinin en küçüklenmesi [88], özdeş olmayan iş büyüklüklerine sahip tek toplu işleme makinesinde çizelgeleme [89], en düşük serbest enerjiye sahip protein yapısını bulma problemi [90], NP-Tam bir problem olan sudoku bulmacalarının çözümü [91], çok katmanlı algılayıcı yapay sinir ağının eğitimi [92] olarak karşımıza çıkmaktadır. Son olarak ise Karaboğa ve Akay [93] arı sürüsü davranışları ve bu

davranışlara dayalı geliştirilen algoritmaları içeren bir sınıflandırma ve inceleme çalışması sunmuşlardır.

2.7. Arı Algoritması

AA popülasyon tabanlı bir arama algoritması olup sürü zekâsına dayalı meta sezgisel yöntemlerden birisidir. Algoritma gerçek arıların yiyecek arama davranışlarını modellemeye dayanmakta olup literatürde kombinatoryal ve genellikle sürekli optimizasyon problemlerinin çözümünde kullanılmıştır.

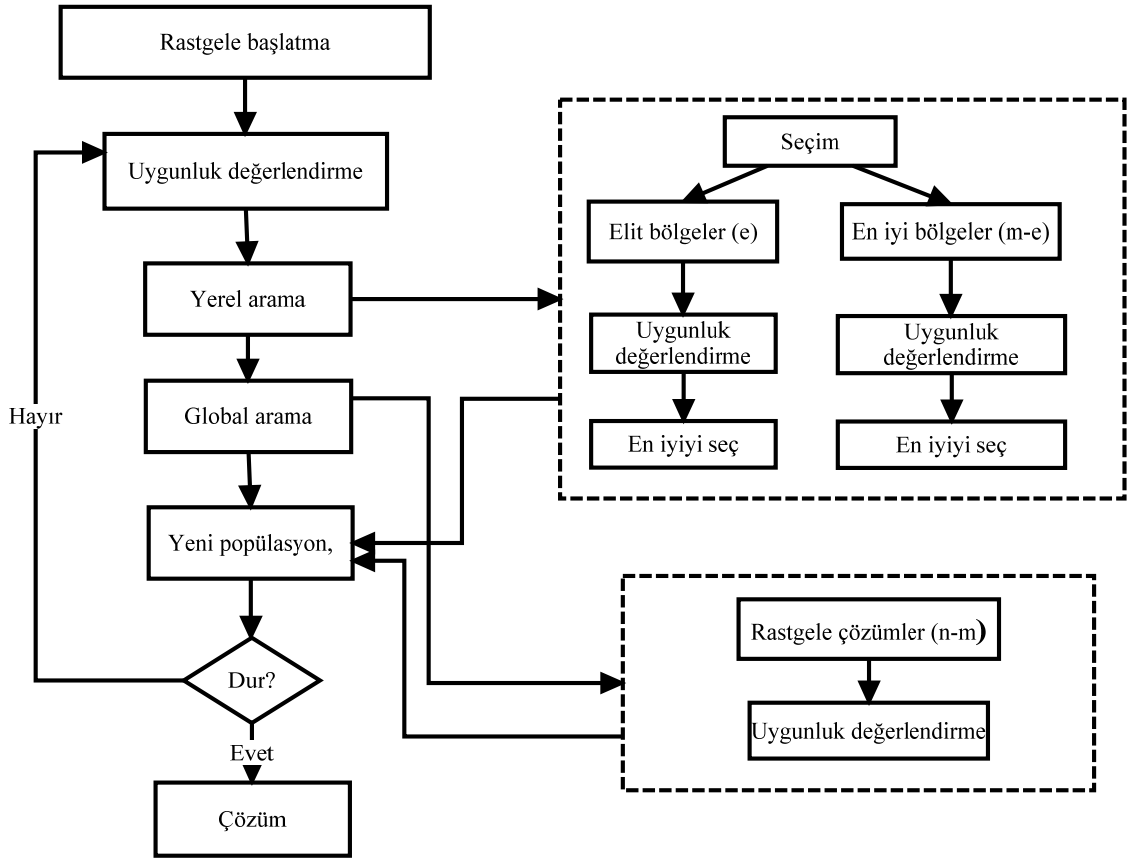
AA'nın genel yapısı Tablo 2.4'te detaylı olarak verilmiştir.

Tablo 2.4. AA'nın temel adımları.

1. Rastgele çözümlerle başlangıç arı popülasyonunu oluştur
2. Popülasyonun uygunluğunu değerlendir
3. *While* (durdurma kriteri sağlanana kadar)
   // Yeni arı popülasyonunun oluşturulması
4. Komşuluk araması için bölgeleri seç
5. Arıları seçilen bölgelere gönder ve uygunluklarını değerlendir
6. Her bir bölgedeki en iyi uygunluk değerine sahip arıyı seç
7. Kalan arıları rastgele arama için ata ve uygunluklarını değerlendir
8. *End While*

Doğadaki arılarda yiyecek arama işlemini önce kâşif arılar başlatır ve rastgele yiyecek kaynakları seçerler. Ayrıca yiyecek toplama süreci boyunca popülasyonun belli bir yüzdesi kâşif arılar olarak kalır. Benzer olarak AA da n adet kâşif arının araştırma uzayına rastgele yerleştirilmesi ile başlar. Kâşif arılarca ziyaret edilen noktaların uygunlukları 2. adımda değerlendirilir. Doğadaki arılara bakıldığında, bir arı kolonisi çok sayıda yiyecek kaynağını tüketmek için birçok yönde ve büyük uzaklıklar boyunca arama yapabilmektedir. Prensipte nektar miktarı fazla olan ve düşük bir çabayla tüketilebilen yiyecek kaynakları daha fazla arı tarafından, az miktarda nektara sahip yiyecek kaynakları ise daha az arı tarafından ziyaret edilecektir. Bu gerçekten hareketle 4. adımda en iyi uygunluk değerine sahip arılar (m) komşuluk araması için seçilir. 5. adımda seçilen arıların komşuluğunda araştırma başlar ve daha umut verici çözümleri temsil eden en iyi e bölgeye, seçilen diğer bölgelere (m-e) göre daha fazla arı

gönderilerek daha detaylı arama yapılır. Yeni popülasyonun oluşturulması için her bölgedeki en iyi uygunluk değerine sahip arı 6. adımda seçilir. 7. adımda popülasyondaki diğer arılar (*n-m*) yeni potansiyel çözümler elde etmek için rastgele olarak araştırma uzayına atanırlar. Böylece her iterasyonun sonunda yeni popülasyon, seçilen her bölgenin temsilcileri ve rastgele arama yapan kâşif arılar olmak üzere iki parçadan oluşacaktır. Diğer taraftan AA kâşif arı sayısı ile temsil edilen popülasyon boyutu (*n*), ziyaret edilen *n* bölge içinden seçilen bölge sayısı (*m*), seçilen *m* bölge içindeki en iyi bölge sayısı (*e*), en iyi *e* bölgeye gönderilen izci arı sayısı (*nep*), seçilen diğer bölgelere (*m-e*) gönderilen izci arı sayısı (*nsp*), komşuluk arama boyutu (*ngh*) ve durdurma kriteri olmak üzere birçok parametre içermektedir. AA'na ait akış diyagramı Şekil 2.3'te sunulmaktadır.



Şekil 2.3. AA akış diyagramı [61].

AA'nın çalışma prensibi ve temel yapısı ilk kez Pham ve ark. [6] tarafından önerilmiş olup çok boyutlu ve çok doruklu sayısal test fonksiyonları üzerinde algoritmanın etkinliği ve sağlamlığı gösterilmiştir. Elde edilen sonuçlar deterministik simpleks yöntemi, stokastik benzetimli tavlama, genetik algoritma ve karınca koloni

optimizasyonu algoritmaları ile karşılaştırılmıştır. Önerilen algoritma, karşılaştırma yapılan optimizasyon algoritmalarına göre gradyan bilgisini çok az kullandığından yerel optimumdan kolayca kurtulabilmektedir. Dolayısıyla AA genel olarak optimizasyon hızı ve sonuçların doğruluğu açısından diğer tekniklerden daha üstün performans göstermiştir.

AA geliştiricileri tarafından birçok alana uygulanmış olup bu çalışmalardan bazıları izleyen şekilde özetlenmiştir. İlk olarak Pham ve ark. [94] ahşap kaplama levhalarındaki kusurların belirlenmesinde AA ile yapay sinir ağlarını optimize etmişlerdir. Bu çalışmada arılar nöronlar arasındaki bağlantılara yerleştirilmiş olup ağırlıkların optimum değerini aramaktadır. Yapay sinir ağındaki ağırlıkların optimizasyonu için geri yayılım algoritması yerine kullanılan AA'nın amacı en küçük hata fonksiyonu değerine sahip arıyı bulmaktır. Elde edilen deneysel çalışma sonuçları geri yayılım ve minimum uzaklık sınıflandırıcı [95] algoritmaları ile karşılaştırılmış ve AA'nın doğru sonuçlara daha hızlı ulaştığı belirtilmiştir. Benzer şekilde Pham ve ark. [96] çok katmanlı algılayıcı ağlarının eğitiminde, Pham ve ark. [97] istatistiksel proses kontrolünde kullanılan kontrol grafiklerinde örüntü tanıma için radyal temel fonksiyon ağının eğitiminde, Pham ve ark. [98] ise yine kontrol grafiklerinde örüntü tanıma için öğrenen vektör nicelendirme ağının eğitiminde AA algoritmasını kullanmış ve geri yayılım algoritmasına göre daha iyi sonuçlar elde edilmiştir.

AA'nın diğer uygulama alanları ise hücresel imalat sistemlerinde parça aileleri ve makine hücrelerinin eş zamanlı belirlenmesine dayanan hücre oluşturma problemi [99], tek makinede çizelgeleme problemi [100], bulanık mantık kontrolörlerinin ayarlanması [101], veri sınıflandırma [102], çok boyutlu optimizasyon problemlerinden standart bir mekanik tasarım problemi olan kaynaklı kiriş yapısının tasarımı [103], ahşap kusurlarının sınıflandırılmasında özellik belirleme ve parametre optimizasyonu [104], bir veri kümesini farklı gruplara ayırabilmek için belirlenmesi gereken niteliklerin sayısını azaltmaya yönelik nitelik seçimi problemi [105], baskı devre kartı montaj planlama problemi [106], bulanık kümeleme [107], Coca Cola için şişe şekli tasarımı [108], bir robot kolunun ters kinematiğinin modellenmesi için çok katmanlı algılayıcı yapay sinir ağının eğitimi [109], mekanik tasarım optimizasyonu [110, 111] ve çok amaçlı ekonomik güç gönderme problemi [112, 113] olarak karşımıza çıkmaktadır.

AA'nın farklı uyarlamalarına bakacak olursak Pham ve Haj Darwish [114] orijinal AA'ndaki kontrol parametresi sayısını indirgemeye yönelik olarak bulanık açgözlü seçim mekanizmasını kullanmışlardır. Geliştirilen algoritma ile *m*, *e*, *nep* ve *nsp* parametrelerinin kullanıcı tarafından belirlenmesi yerine önerilen mekanizma ile bu parametre değerleri otomatik olarak seçilmektedir. Pham ve ark. [115] ise kimya mühendisliği alanında dinamik optimizasyon problemlerinin çözümü için orijinal AA'ndaki rastgele aramayı tamamlayacak şekilde mutasyon, sünme, çaprazlama, ara değerleme ve dış değerleme operatörlerini algoritmaya dâhil etmişlerdir. Önerilen algoritma karınca koloni optimizasyonu algoritması ile karşılaştırılmış ve önerilen algoritmanın daha üstün performans gösterdiği belirtilmiştir. Pham ve Sholedolu [116] parçacık sürü optimizasyonu algoritmasının erken yakınsama problemini çözebilmek için ilgili algoritmaya AA'nı da dâhil ederek melez bir yapı geliştirmişler ve önerilen algoritmayı çok katmanlı yapay sinir ağlarının eğitiminde kullanmışlardır. Deterministik problemlerin AA ile çözümde tek değerlendirme ve sabit uygunluk değeri yeterli iken, stokastik optimizasyon problemlerinde aynı noktada çok sayıda deneme ve ortalama bir uygunluk değeri hesaplaması gerekmektedir. Bu bilgiden hareketle Pham ve ark. [117] stokastik optimizasyon problemlerinin çözümü için AA'na yeni bir uygunluk değerlendirme mekanizması eklemişlerdir. Bu mekanizma ile her arı için tek bir denemedeki uygunluk değeri yerine birçok deneme sonucunda elde edilen uygunluk değerlerinin ortalaması kullanılmıştır. Pham ve Castellani [61] AA'na araştırma hızını ve doğruluğu artıracak iki prosedür eklemişlerdir. Bunlardan ilki komşuluk arama boyutunun uyarlanır hâle getirilmesidir. Bu mekanizma, yerel arama ile daha iyi uygunluk değerleri elde edildiğinde komşuluk boyutunun sabit tutulmasına, uygunluk değerlerinde bir gelişme olmadığında ise komşuluk boyutunun azaltılmasına dayanmaktadır. AA'nda ikinci iyileştirme ise YAK algoritmasında olduğu gibi bir çözüm belirli bir iterasyon boyunca geliştirilemediğinde rastgele oluşturulan yeni bir çözümle değiştirilmesini içermektedir. Önerilen algoritmanın etkinliği farklı karmaşıklık derecelerine sahip fonksiyon optimizasyon problemleri üzerinde evrimsel algoritma, parçacık sürü optimizasyonu ve YAK algoritması ile karşılaştırılmıştır. AA neredeyse bütün test problemlerinde optimum ya da optimuma yakın sonuçlar elde etmiş ve deneysel sonuçlar AA'nın doğruluk, öğrenme hızı ve sağlamlıktaki gücünü ispatlamıştır.

3. BÖLÜM
GENELLEŞTİRİLMİŞ ATAMA PROBLEMİ ÇÖZÜM YAKLAŞIMLARI

3.1. Giriş

Çalışmanın bu bölümünde genellikle sürekli optimizasyon problemlerine uygulanmış olan AA ve YAK algoritmasının karmaşık tamsayılı optimizasyon problemlerindeki performansını inceleyebilmek için NP-zor bir problem olan GAP'nin çözümünde kullanılmasına yer verilmiştir. Ayrıca GAP'nin AA ile çözümünde kullanılan komşuluk yapılarının algoritma üzerindeki etkisi incelenmiştir.

3.2. Genelleştirilmiş Atama Problemi

GAP'nin amacı minimum toplam maliyetle belirli bir iş kümesini belirli ajanlar kümesine atamayı içermektedir. Her ajan sınırlı kapasiteye sahip tek bir kaynağı temsil etmekte olup her iş mutlaka tek bir ajana atanmalıdır. Bu atama esnasında ajan kapasitesinden belirli bir miktar kullanılmaktadır. GAP bilgisayar ve iletişim ağları, yerleştirme problemleri, araç rotalama, grup teknolojisi ve çizelgeleme gibi birçok uygulama alanına sahiptir. GAP'ne ait geniş inceleme çalışmaları ve uygulama alanları Martello ve Toth [118, 119], Cattrysse [120], Cattrysse ve ark. [121] ve Öncan [122] çalışmalarında sunulmaktadır. Bilimsel yazında GAP'nin çözümü için Ross ve Soland [123], Fisher ve ark. [4], Martello ve Toth [119], Savelsbergh [124], Nauss [125] tarafından çeşitli kesin çözüm algoritmaları önerilmiştir. Diğer taraftan GAP'nin çözümü için birçok sezgisel yöntem de geliştirilmiştir. Martello ve Toth [118, 119] yerel arama ve açgözlü yöntemin birleşiminden oluşan bir algoritma sunarken, Osman [126] GAP'nin çözümü için benzetimli tavlama ve tabu arama algoritmalarını geliştirmiştir. Chu ve Beasley [127] eniyilik ve uygunluğu aynı anda iyileştirmeye çalışan bir genetik algoritma yaklaşımı önermişlerdir. Farklı değişken

derinlik arama algoritmaları [128-130], çıkarım zinciri tabanlı tabu arama algoritmaları [131-133], açgözlü rastgele uyarlanır sezgiseline dayalı maks-min karınca sistemi [134], yol birleştirme yaklaşımları [135-140], karınca koloni optimizasyonu [141], kısıt-oran sezgiseli ile birleştirilmiş genetik algoritma [142], diferansiyel gelişim algoritması [143] son yıllarda GAP için önerilmiş diğer meta-sezgisel yaklaşımlar olarak verilebilir.

AA ve YAK algoritmasının uygulama alanı olarak GAP'nin seçilmesi ise Fisher ve ark. [4] tarafından ispatlandığı gibi problemin NP-zor yapısından kaynaklanmaktadır. Diğer taraftan Martello ve Toth [119] GAP'nin NP-tam bir yapıya sahip olduğunu da ispatlamıştır. Bu çalışmada geliştirilen arı algoritmalarının GAP üzerindeki performansı, algoritmaların karmaşık kısıtlandırılmış kombinatoryal optimizasyon problemlerindeki becerisi için iyi bir gösterge olacaktır.

GAP tamsayılı programlama modeli, kullanılan notasyonlarla birlikte aşağıdaki şekilde formüle edilmektedir.

I: İşler kümesi (i=1,…,n)

J: Ajanlar kümesi (j=1,…,m)

b_j: j ajanının kaynak kapasitesi ($b_j \geq 0$)

a_{ij}: i işi j ajanına atandığında ihtiyaç duyulan kaynak miktarı ($a_{ij} \geq 0$)

c_{ij}: i işini j ajanına atama maliyeti ($c_{ij} \geq 0$)

x_{ij}: Karar değişkeni (x_{ij}=1, i işi j ajanına atanırsa; 0, diğer durumlarda)

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Kısıtlar

$$\sum_{i=1}^{n} a_{ij} x_{ij} \leq b_j \qquad 1 \leq j \leq m \quad \forall j \qquad (3.1.)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad 1 \leq i \leq n \quad \forall i \qquad (3.2.)$$

$x_{ij} \in \{0,1\}$ $\quad 1 \leq i \leq n \quad \forall i, \quad 1 \leq j \leq m \quad \forall j$

Yukarıdaki formülasyonda amaç fonksiyonu toplam atama maliyetini temsil ederken birinci kısıt kümesi ajanlara ait kaynak kapasitesiyle ilişkili olup ikinci kısıtlar kümesi her bir işin tek bir ajana atanmasını sağlamaktadır.

3.3. Genelleştirilmiş Atama Problemi için Geliştirilmiş Arı Algoritması

GAP'nin çözümü için çıkarım zinciri komşuluk mekanizmasına sahip AA geliştirilmiş, geniş bir deneysel çalışma neticesinde önerilen algoritma sonuçları bilimsel yazında sunulan meta-sezgisel algoritma sonuçları ile karşılaştırılmış ve önerilen algoritmanın iyi sonuçlar verdiği tespit edilmiştir. Geliştirilmiş AA'nın detaylı adımları Tablo 3.1'de yer almakta olup kullanılan notasyonlar aşağıdaki gibi tanımlanmıştır.

S:	Kâşif arı sayısı (s=1,…,S)
σ^s:	Kâşif arı çözümü
$fit(\sigma^s)$:	Kâşif arı çözümünün uygunluk fonksiyonu değeri
α_j:	j. ajanın kapasitesinden 1 birim fazla kullanma maliyeti
e:	En iyi görevli arı sayısı
nep:	e adet görevli arının her birine gönderilecek izci arı sayısı
nsp:	P-e adet görevli arının her birine gönderilecek izci arı sayısı ($nsp<nep$)
σ^{ls}:	Yerel arama ile bulunan komşu çözüm, ls={*kaydırma, çiftkaydırma, çıkarımzinciri, eniyiizci*}
$fit(\sigma^{ls})$:	Yerel arama ile bulunan komşu çözümün uygunluk fonksiyonu değeri, ls={*kaydırma, çiftkaydırma, çıkarımzinciri, eniyiizci*}
$LimitSayacı(\sigma^p)$:	σ^p çözümü için gelişme olmaksızın geçirilen iterasyon sayısı
σ^{eniyi}:	En iyi çözüm
$fit(\sigma^{eniyi})$:	En iyi çözümün uygunluk fonksiyonu değeri

Tablo 3.1. GAP için geliştirilmiş AA adımları.

1. Parametreleri başlangıç durumuna getir
2. ARUAP algoritması ile *S* adet kâşif arı çözümü oluştur
3. Kâşif arı çözümlerinin uygunluk fonksiyonlarını değerlendir

$$fit(\sigma^s) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} + \left(\sum_{j=1}^{m} \alpha_j maks\left\{0, \sum_{i=1}^{n} a_{ij}x_{ij} - b_j\right\}\right)$$

4. I=0
5. Do

   Artan şekilde sırala$_{s=1....S}$ $fit(\sigma^s)$ ve en iyi *P* adet çözümü görevli arı olarak belirle
   En iyi *e* adet görevli arıyı seç
   En iyi *e* adet görevli arının her birine *nep* adet izci arı ata
   Kalan *P-e* adet görevli arının her birine *nsp* adet izci arı ata
   k=0
   Do
      Kaydırma
         Eğer $fit(\sigma^{kaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{kaydırma}$
      Çift Kaydırma
         Eğer $fit(\sigma^{çiftkaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{çiftkaydırma}$
      Görevli arılara atanmış her bir izci arı için
      {
         Çıkarım Zinciri
            Eğer $fit(\sigma^{çıkarımzinciri}) < fit(\sigma^p)$ ise $\sigma^{eniyiizci} = \sigma^{çıkarımzinciri}$
      }
      Eğer $fit(\sigma^{eniyiizci}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{eniyiizci}$ ve $LimitSayacı(\sigma^p)$=0,
      Aksi takdirde $LimitSayacı(\sigma^p)$=$LimitSayacı(\sigma^p)$+1
      En iyi çözümü güncelle
         Eğer (σ^p uygun ve $fit(\sigma^p) < fit(\sigma^{eniyi})$ ise $\sigma^{eniyi} = \sigma^p$
      Eğer ($LimitSayacı(\sigma^p) > limit$) ise ARUAP algoritması ile yeni bir kâşif arı çözümü oluştur
      α_j değerlerini güncelle, $fit(\sigma^p)$ değerlendir
      k=k+1
   While (k<*P*)
   ARUAP algoritması ile *S-P* adet yeni kâşif arı çözümleri oluştur.
   I=I+1
While (*I*=*MaksIter*)

Geliştirilmiş AA, parametrelerin başlangıç değerlerine atanması ile başlar (*S, P, e, nep, nsp, ÇZ-Uzunluğu, MaksIter, limit,* α_j) ve genellikle uygun çözümler üreten ve Bölüm 3.3.1'de detayları verilecek olan ARUAP algoritması ile *S* adet başlangıç kâşif arı çözümü oluşturulması ile devam eder. Oluşturulan çözümler kümesinden *P* adet iyi çözüm görevli arı çözümleri olarak belirlenir. *P* adet iyi çözüm arasından seçilen *e* adet çözüm, en iyi çözümler olarak belirlenir. Daha detaylı bir komşuluk araması için bu en iyi çözümlere *nep* adet izci arı gönderilir. Daha az sayıda izci arı ise kalan *P-e* adet

çözüme gönderilir. Yerel arama için her görevli arıya sırasıyla kaydırma (Bölüm 3.3.2.1) ve çift kaydırma (Bölüm 3.3.2.2) komşuluk mekanizmaları uygulanır. Daha iyi bir çözüm bulunmuşsa görevli arı çözümü güncellenir. Her bir görevli arıya atanan izci arılar, çıkarım zinciri komşuluk yapısı (Bölüm 3.3.2.3) ile komşu çözümler oluşturur. En iyi izci arı orijinal görevli arı ile karşılaştırılır, eğer daha iyiyse görevli arı çözümü güncellenir. Diğer taraftan görevli arı çözümleri, o ana kadar bulunan en iyi çözümle karşılaştırılır ve gerekli şartlar sağlanmışsa (uygun ve bir önceki en iyi çözümden daha iyi bir çözüm elde edilmişse) en iyi çözüm güncellenir. Görevli arı çözümü *limit* adet iterasyon boyunca geliştirilememişse ARUAP algoritması kullanılarak yeni bir kâşif arı çözümü oluşturulur. Bütün bu işlemler sonucunda uygun bir çözüm bulunamamışsa Tablo 3.5'te verilen algoritma kullanılarak α_j değerleri artırılır; en az bir uygun çözüm bulunmuşsa α_j değerleri aynı algoritma kullanılarak azaltılır. Böylece uygun bir çözüm bulunamamışsa araştırma bölgesi farklılaştırılmaya, en az bir uygun çözüm bulunduğunda ise bu çözüm etrafında daha detaylı bir arama yapılabilmesi sağlanmaya çalışılmıştır. Komşuluk araması tamamlandıktan sonra *S-P* adet yeni çözüm oluşturularak başlangıç popülasyon sayısının tamamlanması sağlanır. Böylece bir sonraki iterasyona ait popülasyon, güncellenen *P* adet görevli arı çözümü ile *S-P* adet yeni kâşif arı çözümünden oluşmaktadır.

Bu çalışmada orijinal AA'nın bazı adımları GAP'nin çözümü için farklılaştırılmıştır. Bu farklılıklardan ilki sürekli optimizasyon problemlerinin çözümü için önerilen orijinal AA'na ait komşuluk boyutu parametresi yerine GAP için tasarlanmış çıkarım zinciri komşuluk mekanizmasına ait *ÇZ-Uzunluğu* parametresinin kullanılmasıdır. Çıkarım zinciri komşuluğunda gerçekleştirilecek kaydırma hareketi sayısını gösteren *ÇZ-Uzunluğu* komşuluk boyutunu belirlemektedir. Diğer bir farklılık ise kâşif arılarla ilgili olup orijinal AA'nda kâşif arılar algoritma sonlandırılana kadar çözüm uzayında arama yapmaya devam ederken Geliştirilmiş AA'nda buna ek olarak bir araştırma bölgesindeki nektar miktarı yeterince azaldığında yeni kâşif arıların algoritmaya dâhil edilmesine karar verilmiştir. Diğer bir deyişle belirli bir iterasyon ya da süre sonunda geliştirilemeyen bir çözüm varsa o çözüm tekrar oluşturulmaktadır. Geliştirilmiş AA'nın önemli adımlarının detayları sonraki bölümlerde verilmiştir.

3.3.1. Arı Kolonisinin Oluşturulması

Başlangıç arı kolonisi, rastgele çözümler yerine genellikle uygun çözümler üreten Açgözlü Rastgele Uyarlanır Arama Prosedürü (ARUAP, [134]) kullanılarak oluşturulmuştur. ARUAP algoritmasında gerçekleştirilen seçim işlemi bir olasılık fonksiyonuna bağlı olarak yapılmakta ve bu fonksiyon her iterasyon sonunda iyi çözümlere göre güncellenmektedir. İlgili algoritmanın adımları Tablo 3.2'de yer almakta olup algoritmanın kısaca işleyişi ve kullanılan notasyonlar aşağıdaki gibidir.

- Her bir adımda atanacak bir sonraki iş seçilir.
- Seçilen işin atanacağı ajan belirlenir.
- Bütün işler bir ajana atanana kadar bu iki adım tekrarlanır.

S_j: j ajanına atanan işler kümesi

L_i: i işinin atanabileceği ajanlar kümesi

p_{ij}: i işinin j ajanına atanma olasılığı

$\sigma(i)$: i işinin atanmış olduğu ajan

Tablo 3.2. Açgözlü rastgele uyarlanır arama prosedürü adımları.

1. $S_j = \emptyset$, $\forall j=1,\ldots,m$ olsun
2. Her iş için bütün ajanları içeren bir L_i ajanlar listesi oluştur, başlangıçta $L_i = \{1,\ldots,m\}\ \forall i$ olsun
3. İşlerin herhangi bir sırasını ele al, $i=1$
4. *While* (bütün işler atanana kadar) *Repeat*
   L_i listesindeki bütün ajanlar için aşağıdaki olasılık fonksiyonunu hesapla (minimum maliyete sahip ajanın seçilme olasılığı daha yüksek olacaktır)
   $$p_{ij} = \frac{b_j / a_{ij}}{\sum_{l \in L_i} b_l / a_{il}}, \qquad j \in L_i$$
   i işinin atanacağı ajanı bu olasılık değerine göre rastgele seç (seçilen ajan j^* olsun)
   i işini j^* ajanına ata, $S_{j^*} = S_{j^*} \cup \{i\}$
   Eğer $\sum_{i \in S_{j^*}} a_{ij^*} > b_{j^*}$ ise j^* ajanını bütün listelerden sil. 4 nolu adımı tekrarla (kapasite kısıtı sağlanmayabilir)
   $i=i+1$
5. $i \in S_j$ ise $\sigma(i)=j$

3.3.2. Komşuluk Yapıları

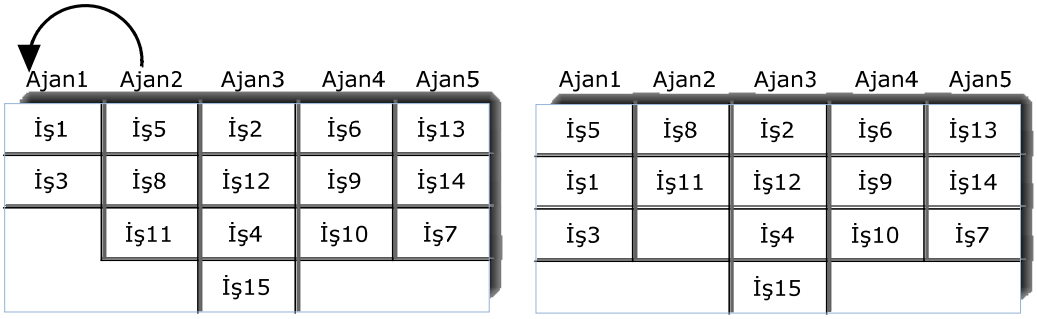
GAP'nin AA ile çözümünde kullanılan kaydırma, çift kaydırma ve çıkarım zinciri komşuluk yapıları aşağıdaki bölümlerde detaylı olarak açıklanmıştır.

3.3.2.1. Kaydırma Komşuluk Yapısı

Kaydırma komşuluk yapısında komşu çözümler, orijinal çözümdeki bir işin ajan atamasının değiştirilmesi sonucu elde edilir (*i* işi *j* ajanından çıkarılarak *w* ajanına atanır $w \neq j$). İlgili komşuluğun işleyişi Tablo 3.3'te özetlenmiş ve Şekil 3.1'de örnek bir uygulama verilmiştir.

Tablo 3.3. Kaydırma komşuluk yapısının işleyişi.

1. $S=\{i | i \in \{1,\ldots,n\}\}$, $i=1$, $\sigma^{kaydırma}=\sigma^{p}$ olsun
2. Eğer $S=\emptyset$ ise dur; aksi takdirde $\sigma'=\sigma^{p}$, i işi σ' den çıkarılır, $S=S-\{i\}$
3. j^{*} ajanı $j \in J/\{\sigma(i)\}$ kümesi içinden $c_{ij}+\alpha_j maks\{0,(\sum_{i\in I,\sigma(i)=j} a_{ij})+a_{ij}-b_j\}$ fonksiyonunu minimize eden ajan olsun
   i işini j^{*} ajanına ata, σ' çözümünü al ve $fit(\sigma')$ hesapla
4. Eğer $fit(\sigma')<fit(\sigma^{\text{kaydırma}})$ ise $\sigma^{kaydırma}=\sigma'$
5. $i=i+1$, 2 nolu adıma dön
6. $\sigma^{kaydırma}$ çözümünü al



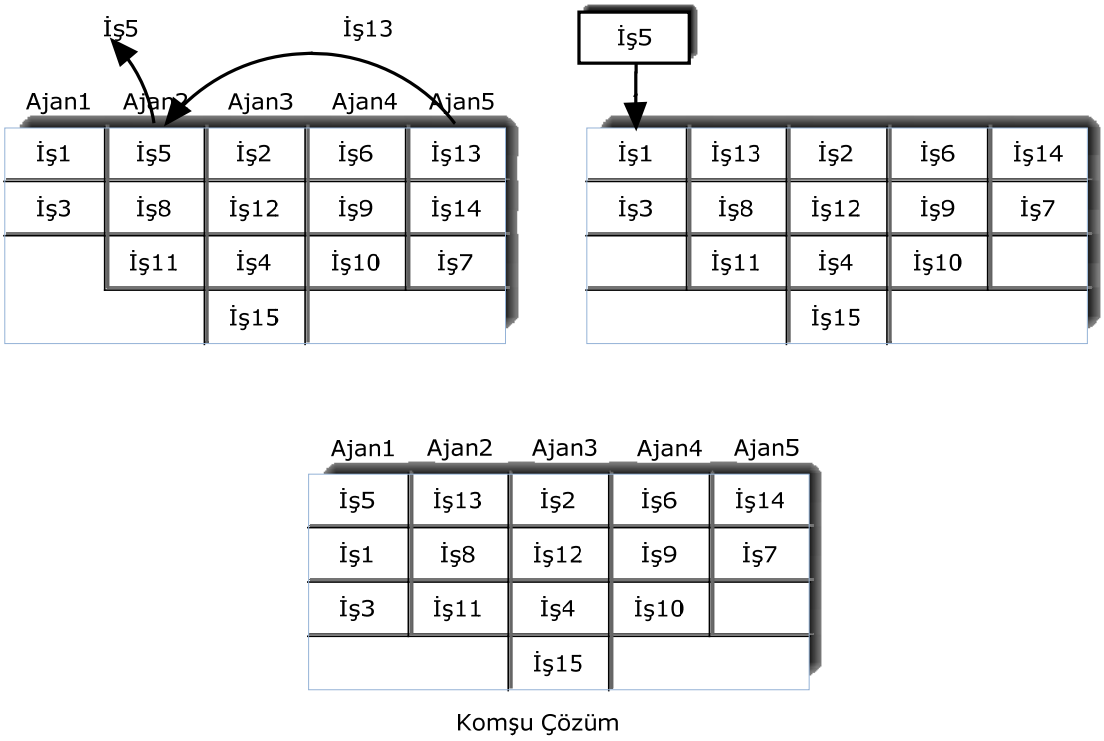
Şekil 3.1. Kaydırma komşuluk yapısı örneği.

Yukarıdaki örnekte 2 nolu ajana atanmış olan 5. iş, kaydırma hareketinden sonra 1 nolu ajana atanmıştır.

3.3.2.2. Çift Kaydırma Komşuluk Yapısı

Bu komşuluk yapısı çıkarım zinciri komşuluğunun özel bir biçimi olup iki kaydırma hareketi gerçekleştirildiğinden çıkarım zinciri uzunluğunun 2 olduğu durumdur (*i* işi *j* ajanından çıkarılarak *w* ajanına atanır $w \neq j$, *w* ajanından *k* işi çıkarılır ve *q* ajanına atanır $q \neq w$). Çift kaydırma komşuluğu kendi içinde, iki farklı işin ajan atamalarının kendi arasında yer değiştirilmesine dayanan, değiştirme komşuluk yapısını da içerir (orijinal çözümde *i* işi *j* ajanına, *k* işi *w* ajanına atanmış iken, *i* işinin ataması *w* ajanına, *k* işinin ataması da *j* ajanına olacak şekilde değiştirilir). Çift kaydırma ve çıkarım zinciri komşuluk yapıları arasındaki temel fark, çıkarım zinciri komşuluğunda her kaydırma

hareketi için işler *B* listesinden seçilirken çift kaydırma komşuluğunda yeni bir kaydırma hareketinin bütün işler listesi kullanılarak belirlenmesidir. Bu komşuluk yapısı Yagiura ve ark. [133] çalışmasında önerilen çift kaydırma mekanizmasının basitleştirilmiş hâlidir. Şekil 3.2'de çift kaydırma komşuluk yapısına ait bir örnek verilmiş olup 5 nolu ajana atanmış olan 13. iş 2 nolu ajana, 2 nolu ajana atanmış olan 5. iş 1 nolu ajana kaydırılarak iki kaydırma hareketi gerçekleştirilmiştir.



Şekil 3.2. Çift kaydırma komşuluk yapısı örneği.

3.3.2.3. Çıkarım Zinciri Komşuluk Yapısı

Diğer komşuluk yapılarına göre daha güçlü ancak daha karmaşık olan çıkarım zinciri komşuluk yapısında bir komşu çözüm, sayısı zincir uzunluğu ile belirlenen çoklu kaydırma hareketlerinin gerçekleştirilmesi ile elde edilir. i_0 işinin atanmış olduğu $\sigma(i_0)$ ajanından çıkarılıp serbest bir iş haline getirildiğini varsayalım. Bu çıkarım hareketi sonucunda $\sigma(i_0)$ ajanının kullanılabilir kaynak miktarı artmıştır. $Mevcut(i_0)$, i_0 işinin çıktığı ajanda kalan kaynak miktarı olup aşağıdaki gibi tanımlanmıştır.

$$Mevcut(i_0) = \begin{cases} a_{i_0,\sigma(i_0)} - p_{\sigma(i_0)}(\sigma) & eğer\ a_{i_0,\sigma(i_0)} > p_{\sigma(i_0)}(\sigma) \\ a_{i_0,\sigma(i_0)} & diğer\ durumlarda \end{cases} \qquad (3.3.)$$

$$p_j(\sigma) = maks\left\{0, \left(\sum_{i\in I,\sigma(i)=j} a_{ij}\right) - b_j\right\} \quad (3.4.)$$

$a_{k,\sigma(i_0)} \leq Mevcut(i_0)$ eşitsizliğini sağlayan işler içinden $\sigma(i_0)$ ajanına kaydırılması en kârlı olan işin i_1 olduğu varsayımıyla i_1 işi $\sigma(i_0)$ ajanına kaydırılır ve bu çözüme *referans yapısı* adı verilir. Bu çıkarım hareketinden sonra serbest iş i_0 Tablo 3.4'te (Adım-3) verilen uygunluk fonksiyonuna olan etkisine göre başka bir ajana atanmaya çalışılacaktır. Bu aşamaya da *deneme hareketi* adı verilir. Bir sonraki çıkarım hareketi deneme hareketleri sonucu elde edilen çözümlere değil, bir önceki referans yapısına uygulanmaktadır. Çıkarım zinciri komşuluğunun genel işleyişi Tablo 3.4'te, örnek bir uygulama ise Şekil 3.3'te verilmiştir.

Tablo 3.4. Çıkarım zinciri komşuluk yapısının işleyişi.

1. $S=\emptyset$ olsun
2. Eğer $S=I'$ ya da l=*ÇZ-Uzunluğu* ise dur; aksi takdirde rastgele bir $i_0 \in I' \backslash S$ seç, $S=SU\{i_0\}$ ve $\sigma'=\sigma$ olsun (i_0 işi $\sigma(i_0)$ ajanından çıkarılır)
3. j^* ajanı $j \in J\backslash\{\sigma(i_0)\}$ kümesi içinden
$$c_{i_0 j} + \alpha_j maks\left\{0, \left(\sum_{i\in I,\sigma(i)=j} a_{ij}\right) + a_{i_0 j} - b_j\right\}$$
fonksiyonunu minimize eden ajan ve l=0 olsun
4. Eğer $B(i_l)\backslash\{i_k | k \leq l\} = \emptyset$ ise 2 nolu adıma dön, aksi takdirde $l=l$+1 olsun ve 5 nolu adıma geç.
5. $i_l \in B(i_{l-1})\backslash\{i_k | k \leq l-1\}$ listesinden rastgele bir iş seç ve $\sigma'(i_l)=\sigma(i_{l-1})$ (i_l işinin çıkarım hareketi)
   $\sigma'(i_0)=\sigma(i_l)$ (i_0, $\sigma(i_l)$ ajanına eklenir - deneme hareketi)
   $\sigma'(i_0)=j^*$ (i_0, j^* ajanına eklenir - deneme hareketi)
6. 4 nolu adıma dön.

Tablo 3.4'te kullanılan çeşitli fonksiyon ve kümeler aşağıdaki gibi tanımlanmıştır.

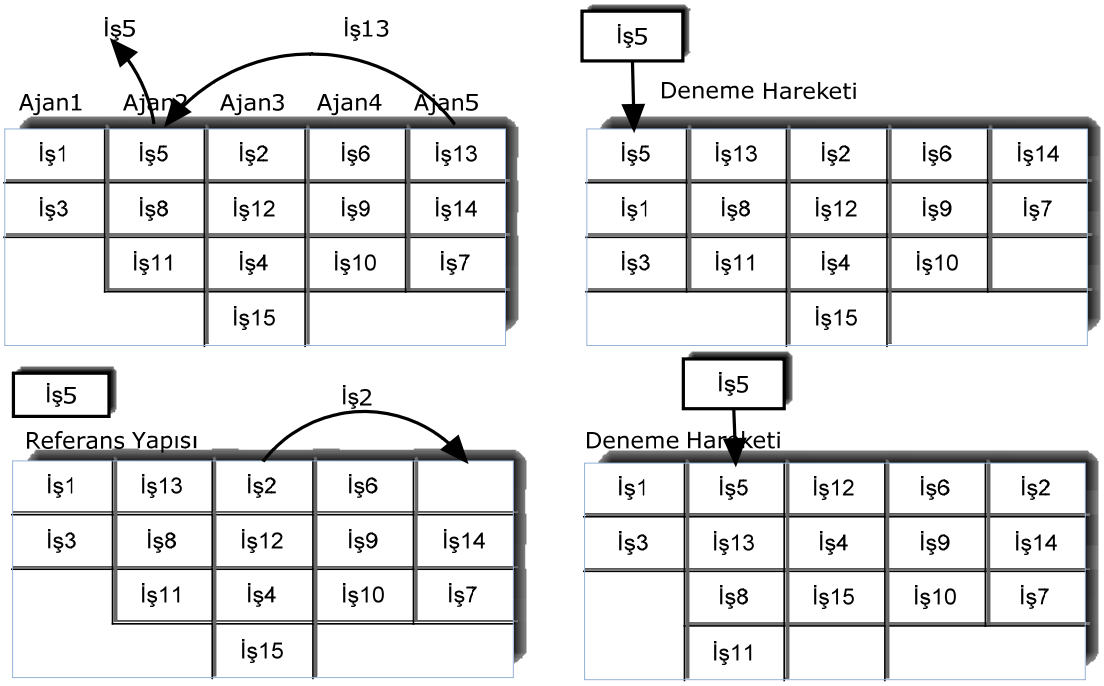
$$I' = \left\{k \in I \middle| \exists h \in I \; s.t. \; a_{h,\sigma(k)} \leq Mevcut(k) \; ve \; \sigma(h) \neq \sigma(k)\right\} \quad (3.5.)$$

$$skor(i,j) = c_{ij} \quad (3.6.)$$

$$eniyiskor(i) = min\left\{skor(k,\sigma(i)) \middle| k \in I', \sigma(k) \neq \sigma(i) ve a_{k,\sigma(i)} \leq Mevcut(i)\right\} \quad (3.7.)$$

$$B(i) = \left\{k \in I' \middle| skor\left(k,\sigma(i)\right) = eniyiskor(i), \sigma(k) \neq \sigma(i) ve a_{k,\sigma(i)} \leq Mevcut(i)\right\} \quad (3.8.)$$

$\boldsymbol{I}'$, başka bir ajandan $\sigma(i)$ ajanına kaydırılması mümkün işler kümesini; *skor(i,j)*, *i* işini *j* ajanına kaydırma maliyetini; $eniyiskor(i)$, $\boldsymbol{I}'$kümesindeki kaynak gereksinimi *Mevcut(i)*'den az olan işler içinden en düşük skor değerine sahip işin skor değerini; *B(i)*, ise en iyi skor değerine sahip işler kümesini temsil etmektedir.



Şekil 3.3. Çıkarım zinciri komşuluk yapısı örneği.

Yukarıdaki şekilde 5. iş 2 nolu ajandan çıkarılarak serbest iş haline getirilmiştir. Kaynak kullanım miktarı *Mevcut(5)* değerinden az olan işler arasında en iyi skor değerine sahip olan 13. iş kaydırma hareketi için seçilir. 13. iş 5 nolu ajandan çıkarılarak 2 nolu ajana atanır ve bu çözüme referans yapısı adı verilir. Sonraki adımda uygunluk fonksiyonuna olan etkisine göre, serbest olan 5. iş 1 nolu ajana atanır ve deneme hareketi gerçekleştirilmiş olur. Elde edilen bu çözüm *ÇZ-Uzunluğu*=2 için oluşturulmuştur ve çift kaydırma hareketi olarak adlandırılır.

ÇZ-Uzunluğu>2 ise aynı adımlar önceki referans yapısı üzerinde tekrarlanır. Örneğe dönecek olursak, *Mevcut(13)* değeri güncellenerek bir sonraki kaydırma hareketi için 2. iş seçilir. 2. iş 5 nolu ajana atandıktan sonra yine uygunluk fonksiyonuna olan etkisine göre serbest olan 5. iş 2 nolu ajana atanır. Elde edilen bu çözüm ise *ÇZ-Uzunluğu*=3 için oluşturulmuş durumdur ve önceden belirlenen zincir uzunluğunca aynı adımlar tekrarlanarak tam bir komşu çözüme ulaşılır.

Yagiura ve ark. [133] tabu arama algoritması ile birlikte kaydırma, çift kaydırma ve çıkarım zinciri komşuluk yapılarını kullanmıştır. Bu tez çalışmasında ise yerel arama stratejileri daha farklı bir şekilde uygulanmıştır.

- Mevcut çalışmada skor fonksiyonu olarak maliyetler (c_{ij}) tercih edilmiştir. Yagiura ve ark. [133] çalışmasında verilen diğer skor fonksiyonlarının kullanımıyla bazı problemlerde daha iyi çözümler ya da daha düşük CPU süreleri elde edilebilmektedir. Ancak bütünlük açısından tüm problemler için minimum maliyet değeri kullanılmıştır.

- Kaydırma komşuluk yapısı her iş için tekrarlanmakta ve en iyi gelişme, yeni çözüm olarak kabul edilmektedir.

- Çift kaydırma mekanizması, çıkarım zinciri uzunluğunun 2 olduğu durum şeklinde basitleştirilmiş ve aynı zamanda değiştirme deneme hareketine de izin verilmiştir. En iyi gelişmeyi veren çift kaydırma hareketini belirleyebilmek için I' kümesindeki her iş için ilgili mekanizma çalıştırılmıştır.

- Çıkarım zinciri komşuluk yapısının karmaşıklığı ve uzun işlem süresi sebebiyle ilk gelişme, yeni çözüm olarak kabul edilmiştir. Aynı çözüm üzerinde çıkarım zinciri komşuluğuyla farklı çözümler elde eden izci arıların her birinin farklı bir işten başlayarak kaydırma hareketlerini gerçekleştirmesi sağlanmıştır.

- Tablo 3.5'te detayları verilen ceza katsayısının (α_j) uyarlanır kontrolü mekanizması, önerilen algoritmada da kullanılmıştır. Ancak α_j başlangıç değerleri, algoritmanın yakınsama yeteneğini görebilmek için 1 olarak belirlenmiştir.

- AA popülasyon tabanlı bir arama stratejisi kullanmakta ve arama stratejisinin bu özelliğinin problem karmaşıklığı arttıkça daha faydalı olması beklenmektedir.

3.3.3. Uygunluk Fonksiyonu

GAP'nde optimum çözümlerin uygun olmayan çözümlere çok yakın olduğu tespit edildiği için [133] araştırma alanında uygun olmayan çözümlerin de yer almasına izin verilmiştir. Böylece uygun çözümlere sahip bölgelerin dar olduğu ya da ayrı ayrı birkaç

dar bölgeden oluştuğu durumlarda fayda sağlanmıştır. Bu doğrultuda başlangıç ve komşu çözümlerin oluşturulması aşamasında, uygun olmayan çözümlerin üretilmesine izin verilmiştir. Dolayısıyla uygunluk fonksiyonu olarak, GAP'ne ait amaç fonksiyonuna uygun olmayan çözümlerin cezalandırılmasını amaçlayan bir ceza terimi eklenmiştir. Böylece GAP üzerine yapılan çalışmaların çoğunda olduğu gibi uygun olmayan alanlarda da çalışmaya izin verilmiş, ancak uygun olmayan çözümler uygun olmama derecesine göre cezalandırılmıştır.

$$fit(\sigma^s) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} + \left(\sum_{j=1}^{m} \alpha_j maks\left\{0, \sum_{i=1}^{n} a_{ij}x_{ij} - b_j\right\}\right) \quad (3.9.)$$

Uygunluk fonksiyonundaki ilk terim toplam atama maliyetini, ikinci terim ise ceza fonksiyonunu temsil etmektedir. Eğer bir çözüm uygun değilse o çözüme ait uygunluk fonksiyonunun ikinci terimi pozitif değer alacak, dolayısıyla araştırma uygun bir çözüme doğru yönlendirilecektir. Diğer taraftan eğer kapasite aşılmamışsa ikinci terim 0 değerini alacaktır. α_j parametresinin değeri, uygun olmayan çözümlerin cezalandırılması ve araştırmayı uygun çözümlere yönlendirebilmek amacıyla programın çalıştırılması süresince değiştirilmekte yani ceza maliyeti uyarlanır olarak kontrol edilmektedir.

3.3.4. Ceza Katsayısının Uyarlanır Kontrolü

Ceza katsayısının uyarlanır kontrolü Yagiura ve ark. [133] çalışması esas alınarak gerçekleştirilmiş olup detaylı adımlar Tablo 3.5'te verilmiştir. Komşuluk yapıları için bir iterasyonun tamamlanmasından sonra α_j değerleri güncellenmektedir. Böylece uygun bir çözüm bulunduğunda bu çözüm etrafında daha detaylı bir arama yapılabilmesi sağlanmaktadır.

Tablo 3.5. α_j parametresinin uyarlanır kontrolü.

1. İzci arı komşuluklarında uygun bir çözüm bulunamamışsa aşağıdaki formülasyonla bütün $j \in J$ için α_j değerlerini artır

$$\alpha_j = \begin{cases} \alpha_j \left(1 + \Delta . q_j^{artış}(\sigma)\right), & \alpha_j > 0 \\ \Delta . \dfrac{q_j^{artış}(\sigma) min_{h \in J}\{b_h \alpha_h | b_h \alpha_h > 0\}}{b_j}, & diğer\ durumlarda \end{cases}$$

$$\Delta = \begin{cases} \dfrac{adımbüyüklüğüartışı}{maks_{j \in J} \left| q_j^{artış}(\sigma) \right|}, & eğer\ maks_{j \in J} \left| q_j^{artış}(\sigma) \right| > 0 \\ 0, & diğer\ durumlarda \end{cases}$$

$$q_j^{artış}(\sigma) = p_j(\sigma)/b_j$$

2. İzci arı komşuluklarında en az bir uygun çözüm bulunmuşsa bütün α_j değerlerini, $q^{artış}(\sigma)$ değeri $q^{azalış}(\sigma)$ ve *adımbüyüklüğüartışı* değeri *adımbüyüklüğüazalışı* olacak şekilde yukarıdaki formülasyonları kullanılarak azalt.

,

$$q_j^{azalış}(\sigma) = \begin{cases} -1, & eğer\ p_j(\sigma) = 0 \\ 0, & diğer\ durumlarda \end{cases}$$

3.4. Genelleştirilmiş Atama Problemi için Geliştirilmiş Yapay Arı Kolonisi Algoritması

GAP'nin çözümü için çıkarım zinciri komşuluk mekanizmasına sahip YAK algoritması geliştirilmiştir. Tablo 3.6'da Geliştirilmiş YAK algoritmasının detaylı adımları verilmiştir.

Tablo 3.6. GAP için geliştirilmiş YAK algoritması adımları.

1. Parametreleri başlangıç değerlerine ata
2. ARUAP algoritması ile *P* adet başlangıç görevli arı çözümü oluştur (σ^p)
3. Görevli arı çözümlerinin uygunluk fonksiyonlarını değerlendir

$$fit(\sigma^p) = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} + \left(\sum_{j=1}^{m} \alpha_j maks\left\{0, \sum_{i=1}^{n} a_{ij}x_{ij} - b_j\right\}\right)$$

4. *I*=0
5. *Do*

   Uygunluk değerleriyle ilişkili olasılıkları hesapla (en küçükleme için)

$$p_p = \frac{\left(\sum 1/_{fit(\sigma^p)}\right)^{-1}}{fit(\sigma^p)}$$

   Hesaplanan olasılıklara göre, görevli arıların yiyecek kaynaklarına gönderilecek izci arı sayısını belirle, p_p*P
   k=0
   Do
      Kaydırma
         Eğer $fit(\sigma^{kaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{kaydırma}$
      Çift Kaydırma
         Eğer $fit(\sigma^{çiftkaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{çiftkaydırma}$
      Görevli arılara atanmış her bir izci arı için
      {
         Çıkarım Zinciri
            Eğer $fit(\sigma^{çıkarımzinciri}) < fit(\sigma^p)$ ise $\sigma^{eniyiizci} = \sigma^{çıkarımzinciri}$
      }
      Eğer $fit(\sigma^{eniyiizci}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{eniyiizci}$ ve $LimitSayacı(\sigma^p)$=0,
      Aksi takdirde $LimitSayacı(\sigma^p) = LimitSayacı(\sigma^p)$+1
      En iyi çözümü güncelle
         Eğer (σ^p uygun ve $fit(\sigma^p) < fit(\sigma^{eniyi})$ ise $\sigma^{eniyi} = \sigma^p$
      Eğer ($LimitSayacı(\sigma^p) > limit$) ise ARUAP algoritması ile yeni bir kâşif arı çözümü oluştur
      α_j değerlerini güncelle, $fit(\sigma^p)$ değerlendir
      k=*k*+1
   While (*k*<*P*)
   Kâşif arı çözümleri ile görevli arı çözümlerini karşılaştır
      ARUAP algoritması ile *S* adet kâşif arı çözümü oluştur (σ^s)
      Artan şekilde sırala$_{s=1\ldots S} fit(\sigma^s)$
      Azalan şekilde sırala$_{p=1\ldots P} fit(\sigma^p)$
      r=0
      Repeat
         Eğer $fit(\sigma^{S-r}) < fit(\sigma^{P-r})$ ise $\sigma^{P-r} = \sigma^{S-r}$
         r=*r*+1
      Until (*r*=*S*)
   I=*I*+1
While (*I*=*MaksIter*)

Geliştirilmiş YAK algoritması, parametrelerin başlangıç değerlerine atanması başlar ve ARUAP algoritması ile P adet başlangıç görevli arı çözümü oluşturulması ile devam eder. Görevli arı çözümlerine, çözümün kalitesi ile orantılı sayıda izci arı atanır. Yerel arama için her görevli arıya sırasıyla kaydırma ve çift kaydırma komşuluk mekanizmaları uygulanır. Daha iyi bir çözüm bulunmuşsa görevli arı çözümü güncellenir. Her bir görevli arıya atanan izci arılar, çıkarım zinciri komşuluk yapısı ile komşu çözümler oluşturur. En iyi izci arı orijinal görevli arı ile karşılaştırılır, eğer daha iyiyse görevli arı çözümü güncellenir. Diğer taraftan görevli arı çözümleri o ana kadar bulunan en iyi çözümle karşılaştırılır ve gerekli şartlar sağlanmışsa (uygun ve bir önceki en iyi çözümden daha iyi bir çözüm elde edilmişse) en iyi çözüm güncellenir. Görevli arı çözümü *limit* adet iterasyon boyunca geliştirilememişse ARUAP algoritması kullanılarak yeni bir kâşif arı çözümü oluşturulur. Bütün bu işlemler sonucunda uygun bir çözüm bulunamamışsa Tablo 3.5'te verilen algoritma kullanılarak α_j değerleri artırılır; en az bir uygun çözüm bulunmuşsa α_j değerleri aynı algoritma kullanılarak azaltılır. İzci arılar tarafından gerçekleştirilen yerel aramaya paralel olarak, kâşif arılar da yeni çözümler araştırmaktadırlar. Elde edilen kâşif arı çözümleri en düşük uygunluk değerine sahip görevli arı çözümleri ile karşılaştırılır ve kâşif arılarla daha iyi bir çözüm elde edilmişse görevli arı çözümleri güncellenir. Son olarak algoritma adımları önceden belirlenmiş iterasyon sayısı kadar tekrarlanır.

Geliştirilmiş YAK algoritmasında başlangıç çözümlerinin oluşturulması, kullanılan komşuluk yapıları, uygunluk fonksiyonu ve ceza katsayısının uyarlanır kontrolü aşamaları Geliştirilmiş AA'ndakiler ile aynıdır.

3.5. Deneysel Çalışma

3.5.1. Problem Tipleri

Test problemleri gap1-gap12 (gap1-gap6/kolay, gap7-gap12/zor) ve gapa-gapd (gapa-gapb/kolay, gapc-gapd/zor) problemlerini içermektedir. Diğer taraftan gap1-12 problemleri, gapa-gapd problemlerine göre daha kolay problemlerdir. gap1-12 problemleri 5 ajan-15 iş ile 10 ajan-60 iş arasında değişen problemler kümesine sahip olup her problem kümesi 5 farklı problem içermekte; dolayısıyla çözülmesi gereken 60 problem bulunmaktadır. Bu problem kümeleri en büyükleme formunda olup optimum

değerleri bilinmektedir. gapa-gapd problemleri ise 5 ajan-100 iş ile 20 ajan-200 iş arasında değişen daha zor problemler olup her problem kümesi 6 farklı problem içermektedir. Dolayısıyla çözülmesi gereken 24 problem vardır. Bu gruptaki problemler en küçükleme formunda olup sadece en iyi değerleri bilinmektedir. Bütün test problemleri OR-Kütüphanesinden (http://mscmga.ms.ic.ac.uk/jeb/orlib/gapinfo.html) temin edilmiştir. gapa-gapd test problemlerinin detayları aşağıda verilmiştir.

gapa: a_{ij}∈U(5,25) ve tamsayı, c_{ij}∈U(10,50) ve tamsayı, b_j=0.6(n/m)15+0.4R şeklinde hesaplanmaktadır.

$$R = maks_{j\in J}\sum_{i\in I, J_{i=j}} a_{ij} \quad \text{ve } J_i = min\left(j \middle| c_{ij} \le c_{kj}, \quad \forall k \in J\right) \qquad (3.10.)$$

gapb: a_{ij} ve c_{ij} gapa'daki gibi hesaplanırken b_j gapa problemlerinde verilen değerlerin %70'i şeklinde hesaplanmaktadır.

gapc: a_{ij} ve c_{ij} gapa'daki gibi hesaplanırken $b_j = 0.8\sum_{i=I} a_{ij}/m$ şeklinde hesaplanmaktadır.

gapd: a_{ij}∈U(1,100) ve tamsayı, c_{ij} = 111-a_{ij}+e, e∈U(-10,10) ve tamsayı, $b_j = 0.8\sum_{i=I} a_{ij}/m$

3.5.2. Geliştirilmiş Arı Algoritması Sonuçları

Geliştirilmiş AA, C# programlama dilinde kodlanmış ve 1.6 GHz CPU ve 512 MB RAM özelliklere sahip Intel Pentium CoreDuo PC kullanılarak gap1-12 ve gapa-d problemleri üzerinde analiz edilmiştir.

Geliştirilmiş AA parametreleri aşağıdaki gibi tanımlanmıştır:

- Kâşif arı sayısı (*S*)
- Görevli arı sayısı (*P*)
- En iyi görevli arı sayısı (*e*)
- Her bir *e* adet görevli arıya gönderilecek izci arı sayısı (*nep*)
- Her bir *P-e* adet görevli arıya gönderilecek izci arı sayısı (*nsp*)
- Çıkarım zinciri komşuluk yapısının uzunluğu (*ÇZ-Uzunluğu*)
- Maksimum iterasyon sayısı (*MaksIter*)

- Gelişme olmaksızın geçirilebilecek maksimum iterasyon sayısı (*limit*)
- Ceza katsayısının başlangıç değeri (α_{jb})
- *adımbüyüklüğüazalışı* > 0
- 0 < *adımbüyüklüğüartışı* < 1

Test problemlerinin yapısı gereği Tablo 3.7'de gösterildiği gibi üç farklı parametre kümesi belirlenmiştir. Bu parametreler bilimsel yazında verilen genel öneriler ve deneyimlere göre belirlenmiştir.

Tablo 3.7. GAP için geliştirilmiş AA parametrelerinin değerleri.

Parametreler	gap1-12	gapa, gapb	gapc, gapd
S	50	100	200
P	20	40	80
e	10	20	40
nep	5	5	5
nsp	2	2	2
ÇZ-Uzunluğu	25	50	75
MaksIter	250	500	1000
limit	25	50	100
α_{jb}	1	1	1
adımbüyüklüğüazalışı	0.1	0.1	0.1
adımbüyüklüğüartışı	0.01	0.01	0.01

Bilimsel yazındaki en iyi sonuçlar genellikle algoritmanın 5 kez çalıştırılması sonucu elde edildiğinden, daha âdil bir karşılaştırma için Geliştirilmiş AA da her test problemi için 5 kez çalıştırılmış ve elde edilen minimum, ortalama, maksimum ve standart sapma değerleri ile 5 koşma sonucu elde edilen en iyi çözüm sayısı Tablo 3.8'de verilmiştir. Tablo 3.8'deki sonuçlar gapa-d problemlerini içermekte olup, gap1-12 problemlerine ait sonuçlar karşılaştırma bölümünde verilecektir.

Tablo 3.8. GAP için geliştirilmiş AA ile elde edilen gapa-d sonuçları.

Problem	Minimum	Ortalama	Maksimum	Standart sapma	En iyi çözüm sayısı
gapa1	1698	1698	1698	0.00	5/5
gapa2	3235	3235	3235	0.00	5/5
gapa3	1360	1360	1360	0.00	5/5
gapa4	2623	2623	2623	0.00	5/5
gapa5	1158	1158	1158	0.00	5/5
gapa6	2339	2339	2339	0.00	5/5
gapb1	1843	1843	1843	0.00	5/5
gapb2	3552	3552	3552	0.00	5/5
gapb3	1407	1407	1407	0.00	5/5
gapb4	2827	2827	2827	0.00	5/5
gapb5	1166	1166	1166	0.00	5/5
gapb6	2339	2339	2339	0.00	5/5
gapc1	1931	1931	1931	0.00	5/5
gapc2	3456	3456.60	3457	0.54	2/5
gapc3	1402	1402	1402	0.00	5/5
gapc4	2806	2806.60	2807	0.54	2/5
gapc5	1243	1243.60	1244	0.54	2/5
gapc6	2392	2392.60	2393	0.54	2/5
gapd1	6353	6354.40	6356	1.34	2/5
gapd2	12744	12746.20	12748	1.78	1/5
gapd3	6356	6358.80	6362	2.58	1/5
gapd4	12442	12445.20	12447	2.04	1/5
gapd5	6221	6226.60	6232	4.61	1/5
gapd6	12276	12280	12284	2.91	1/5

3.5.3. Geliştirilmiş Yapay Arı Kolonisi Algoritması Sonuçları

Önerilen YAK algoritmasının performans analizi için C# programlama dili ve aynı özelliklere sahip PC kullanılarak gap1-12 ve gapa-gapd problem grupları değerlendirilmiştir.

Geliştirilmiş YAK algoritmasının parametreleri aşağıdaki gibi tanımlanmıştır.

- Görevli arı sayısı (P)
- Kâşif arı sayısı (S)
- Çıkarım zinciri komşuluk yapısının uzunluğu (*ÇZ-Uzunluğu*)
- Gelişme olmaksızın geçirilebilecek maksimum iterasyon sayısı (*limit*)
- Maksimum iterasyon sayısı (*MaksIter*)
- Ceza katsayısının başlangıç değeri (α_{jb})
- $adımbüyüklüğüazalışı > 0$
- $0 < adımbüyüklüğüartışı < 1$

Test problemlerinin yapısı, bilimsel yazında belirtilen ilkeler ve deneyimlere göre 3 farklı parametre kümesi tanımlanmıştır (Tablo 3.9). Diğer taraftan kolonideki kâşif arı

sayısı, biyolojik araştırmalar baz alınarak [60] görevli arı sayısının %10'u şeklinde belirlenmiştir.

Tablo 3.9. GAP için geliştirilmiş YAK algoritması parametrelerinin değerleri.

Parametreler	gap1-gap12	gapa, gapb	gapc, gapd
P	50	100	250
S	5	10	25
ÇZ-Uzunluğu	25	50	75
limit	25	50	100
MaksIter	250	500	1000
α_{ib}	1	1	1
adımbüyüklüğüazalışı	0.1	0.1	0.1
adımbüyüklüğüartışı	0.01	0.01	0.01

Önerilen YAK algoritması, gapa-gapd problemleri için yine 5 kez çalıştırılarak elde edilen minimum, ortalama, maksimum ve standart sapma değerleri Tablo 3.10'da verilmiştir.

Tablo 3.10. GAP için geliştirilmiş YAK algoritması ile elde edilen gapa-gapd sonuçları.

Problem	Minimum	Ortalama	Maksimum	Standart sapma	En iyi çözüm sayısı
gapa1	1698	1698	1698	0	5/5
gapa2	3235	3235	3235	0	5/5
gapa3	1360	1360	1360	0	5/5
gapa4	2623	2623	2623	0	5/5
gapa5	1158	1158	1158	0	5/5
gapa6	2339	2339	2339	0	5/5
gapb1	1843	1843	1843	0	5/5
gapb2	3552	3552	3552	0	5/5
gapb3	1407	1407	1407	0	5/5
gapb4	2828	2828.40	2829	0.54	3/5
gapb5	1166	1166	1166	0	5/5
gapb6	2339	2339	2339	0	5/5
gapc1	1931	1931	1931	0	5/5
gapc2	3456	3457	3458	0.70	1/5
gapc3	1402	1402	1402	0	5/5
gapc4	2806	2806.40	2807	0.54	3/5
gapc5	1243	1243	1243	0	5/5
gapc6	2392	2392.80	2394	1.09	3/5
gapd1	6357	6358.40	6360	1.34	2/5
gapd2	12750	12750.60	12751	0.54	2/5
gapd3	6362	6362.40	6363	0.54	3/5
gapd4	12454	12455.80	12460	2.38	1/5
gapd5	6235	6237	6239	1.87	1/5
gapd6	12293	12297.40	12299	2.60	1/5

3.5.4. Geliştirilmiş Arı Algoritması ve Yapay Arı Kolonisi Algoritmalarının Bilimsel Yazındaki Algoritmalarla Karşılaştırılması

Tez çalışmasının bu bölümünde öncelikle gap1-12 daha sonra da gapa-d problem kümesi için karşılaştırma sonuçlarına yer verilmiştir. Tablo 3.11 bilimsel yazındaki

farklı algoritmalar ve önerilen AA ve YAK algoritmalarına ait gap1-12 sonuçlarının optimum değerden ortalama sapmalarını vermektedir. En düşük ortalama sapma değerleri koyu renklendirilmiş ve görüldüğü gibi her iki algoritma da bütün problem kümeleri için algoritmanın her çalışmasında optimum değere ulaşmıştır. Bilimsel yazındaki diğer 12 algoritmayla karşılaştırıldığında önerilen algoritmaların daha iyi performans gösterdiği aşağıdaki tabloda açıkça görülmektedir.

Tablo 3.11. Geliştirilmiş AA ve YAK algoritması ile elde edilen gap1-gap12 sonuçlarının karşılaştırılması.

	AA	YAK	MTH	FJVBB	FSA	MTBB	SPH	LT1FA	RSSA	TS6	TS1	GA_b	GA_a	ASH+LS+TS
gap1	**0.00**	**0.00**	5.43	**0.00**	**0.00**	**0.00**	0.08	1.74	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-
gap2	**0.00**	**0.00**	5.02	**0.00**	0.19	**0.00**	0.11	0.89	**0.00**	0.24	0.10	**0.00**	0.01	-
gap3	**0.00**	**0.00**	2.14	**0.00**	**0.00**	**0.00**	0.09	1.26	**0.00**	0.03	**0.00**	**0.00**	0.01	-
gap4	**0.00**	**0.00**	2.35	0.83	0.06	0.18	0.04	0.72	**0.00**	0.03	0.03	**0.00**	0.03	-
gap5	**0.00**	**0.00**	2.63	0.07	0.11	**0.00**	0.35	1.42	**0.00**	0.04	**0.00**	**0.00**	0.10	-
gap6	**0.00**	**0.00**	1.67	0.58	0.85	0.52	0.15	0.82	0.05	**0.00**	0.03	0.01	0.08	-
gap7	**0.00**	**0.00**	2.02	1.58	0.99	1.32	**0.00**	1.22	0.02	0.02	**0.00**	**0.00**	0.08	**0.00**
gap8	**0.00**	**0.00**	2.45	2.48	0.41	1.32	0.23	1.13	0.10	0.14	0.09	0.05	0.33	0.04
gap9	**0.00**	**0.00**	2.18	0.61	1.46	1.06	0.12	1.48	0.08	0.06	0.06	**0.00**	0.17	**0.00**
gap10	**0.00**	**0.00**	1.75	1.29	1.72	1.15	0.25	1.19	0.14	0.15	0.08	0.04	0.27	0.01
gap11	**0.00**	**0.00**	1.78	1.32	1.10	2.01	**0.00**	1.17	0.05	0.02	0.02	**0.00**	0.20	**0.00**
gap12	**0.00**	**0.00**	1.37	1.37	1.68	1.55	0.10	0.81	0.11	0.07	0.04	0.01	0.17	**0.00**

AA: Geliştirilmiş Arı Algoritması, YAK: Geliştirilmiş Yapay Arı Kolonisi Algoritması, MTH: yapısal sezgisel [118], FJVBB: bir üst CPU limiti dâhilinde dal-sınır yöntemi [4], FSA: benzetimli tavlama algoritmasında sabitleme [120], MTBB: bir üst CPU limiti dâhilinde dal-sınır yöntemi [119], SPH: küme bölüntüleme [121], LT1FA: ilk en iyiyi seçme stratejisiyle uzun dönem iniş algoritması [126], RSSA: melez benzetimli tavlama ve tabu arama [126], TS6: ilk en iyiyi seçme stratejisiyle uzun dönem tabu arama [126], TS1: en iyiyi seçme stratejisiyle uzun dönem tabu arama [126], GA_b: sezgisel operatörlerle genetik algoritma [127], GA_a: sezgisel operatör olmadan genetik algoritma [127], ASH+LS+TS: yerel arama ve tabu arama ile birleştirilmiş maks-min karınca sistemi [134].

Daha büyük boyutlu gapa-d problem kümeleri için geliştirilmiş AA ve YAK algoritmalarıyla elde edilen sonuçlar ise bilimsel yazındaki 2000 yılından sonra yapılan çalışmalarla karşılaştırılmıştır. Karşılaştırma çalışmasının yayınların basım yılı esas alınarak gerçekleştirilmesinin sebebi 2000 yılından önce geliştirilen algoritma sonuçlarının kötü performansa sahip olmasıdır. gapa-gapd problemleri için bilimsel yazındaki 10 farklı algoritmayla yapılan karşılaştırma sonuçları hesaplama süreleri (saniye) ile birlikte Tablo 3.12'de verilmiştir. Ayrıca önerilen algoritmalarla elde edilen çözümlerin kalitesinin karşılaştırılabilmesi için kesin bir çözüm yöntemi olan CPLEX 6.5 sonuçları da Tablo 3.12'de yer almaktadır. Ancak verilen CPLEX 6.5 sonuçları, 100 iş için 150 saniye, 200 iş için de 300 saniyelik bir süre kısıtına sahiptir.

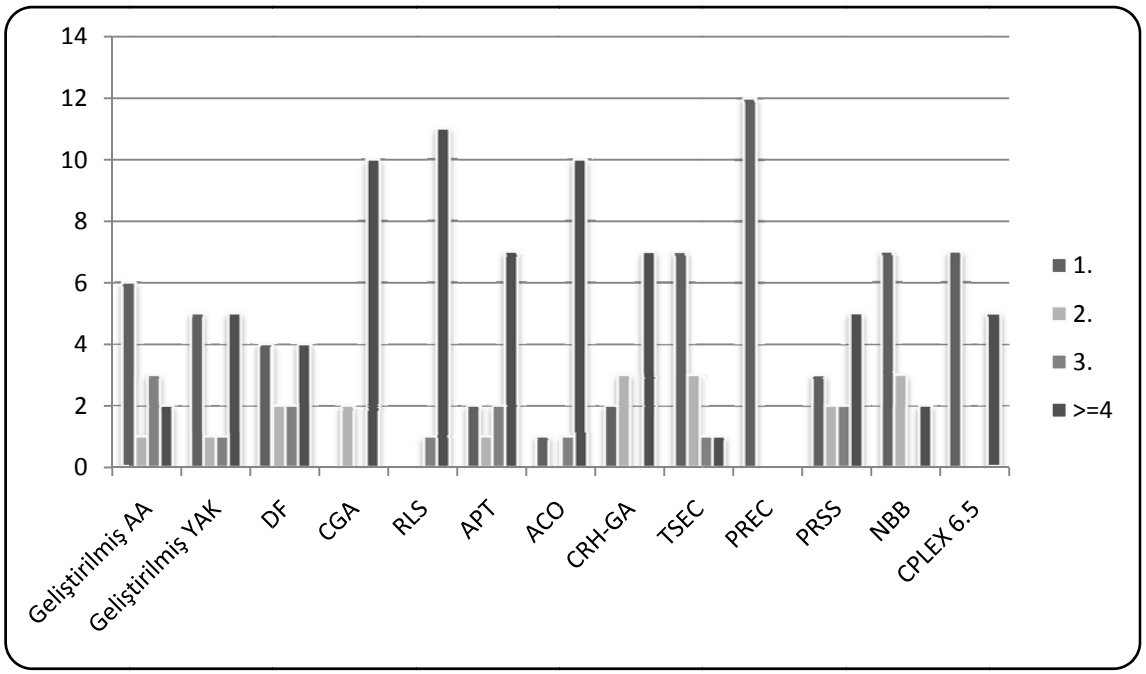
Tablo 3.12. Geliştirilmiş AA ve YAK algoritması ile elde edilen gapa-gapd sonuçlarının karşılaştırılması.

Problem	Geliştirilmiş AA		Geliştirilmiş YAK		DF		CGA		RLS		APT	
	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet
gapa1	0.18	**1698**$^{(1)}$	0.16	**1698**$^{(1)}$	-	-	253	**1698**$^{(1)}$	-	-	-	-
gapa2	0.95	**3235**$^{(1)}$	0.86	**3235**$^{(1)}$	-	-	502	**3235**$^{(1)}$	-	-	-	-
gapa3	0.27	**1360**$^{(1)}$	0.33	**1360**$^{(1)}$	-	-	308	**1360**$^{(1)}$	-	-	-	-
gapa4	1.31	**2623**$^{(1)}$	0.92	**2623**$^{(1)}$	-	-	930	**2623**$^{(1)}$	-	-	-	-
gapa5	0.41	**1158**$^{(1)}$	0.56	**1158**$^{(1)}$	-	-	350	**1158**$^{(1)}$	-	-	-	-
gapa6	1.81	**2339**$^{(1)}$	1.27	**2339**$^{(1)}$	-	-	860	**2339**$^{(1)}$	-	-	-	-
gapb1	5.97	**1843**$^{(1)}$	4.81	**1843**$^{(1)}$	-	**1843**$^{(1)}$	302	**1843**$^{(1)}$	-	-	10.0	**1843**$^{(1)}$
gapb2	45.99	**3552**$^{(1)}$	29.67	**3552**$^{(1)}$	-	**3552**$^{(1)}$	432	3601$^{(4)}$	-	-	121.9	3553$^{(2)}$
gapb3	0.36	**1407**$^{(1)}$	0.54	**1407**$^{(1)}$	-	**1407**$^{(1)}$	165	1410$^{(2)}$	-	-	7.3	**1407**$^{(1)}$
gapb4	315.04	**2827**$^{(1)}$	402.35	2828$^{(2)}$	-	2828$^{(2)}$	949	2831$^{(4)}$	-	-	37.6	2829$^{(3)}$
gapb5	1.12	**1166**$^{(1)}$	1.67	**1166**$^{(1)}$	-	**1166**$^{(1)}$	474	**1166**$^{(1)}$	-	-	11.4	**1166**$^{(1)}$
gapb6	28.65	**2339**$^{(1)}$	17.83	**2339**$^{(1)}$	-	2340$^{(2)}$	683	2347$^{(3)}$	-	-	132.7	2340$^{(2)}$
gapc1	3.61	**1931**$^{(1)}$	3.93	**1931**$^{(1)}$	0.6	**1931**$^{(1)}$	195	1941$^{(2)}$	137.1	1942$^{(3)}$	6.6	**1931**$^{(1)}$
gapc2	18.09	**3456**$^{(1)}$	15.82	**3456**$^{(1)}$	3.7	3457$^{(2)}$	405	3460$^{(4)}$	1693.8	3467$^{(6)}$	146.1	3458$^{(3)}$
gapc3	5.81	**1402**$^{(1)}$	7.35	**1402**$^{(1)}$	3.0	**1402**$^{(1)}$	203	1423$^{(5)}$	178.3	1407$^{(4)}$	31.5	**1402**$^{(1)}$
gapc4	488.89	**2806**$^{(1)}$	368.34	**2806**$^{(1)}$	100.5	2807$^{(2)}$	498	2815$^{(4)}$	1086.0	2818$^{(5)}$	104.3	2810$^{(3)}$
gapc5	23.16	**1243**$^{(1)}$	24.12	**1243**$^{(1)}$	21.6	**1243**$^{(1)}$	479	1244$^{(2)}$	309.6	1247$^{(5)}$	47.5	1244$^{(2)}$
gapc6	646.18	2392$^{(2)}$	562.86	2392$^{(2)}$	137.4	**2391**$^{(1)}$	1059	2397$^{(5)}$	2694.8	2405$^{(6)}$	146.4	2396$^{(4)}$
gapd1	916.03	**6353**$^{(1)}$	828.24	6357$^{(3)}$	62.6	6357$^{(3)}$	259	6479$^{(7)}$	2459.2	6476$^{(6)}$	71.5	6365$^{(5)}$
gapd2	122.41	12744$^{(3)}$	92.65	12750$^{(6)}$	95.5	12747$^{(5)}$	1253	12823$^{(8)}$	11106.9	12923$^{(9)}$	318.1	12747$^{(5)}$
gapd3	538.29	6356$^{(4)}$	416.43	6362$^{(4)}$	107.2	6355$^{(3)}$	497	6390$^{(9)}$	5587.3	6469$^{(10)}$	77.3	6372$^{(5)}$
gapd4	743.95	12442$^{(3)}$	527.92	12454$^{(5)}$	129.2	12457$^{(6)}$	1321	12634$^{(9)}$	47538.7	12746$^{(10)}$	105.2	12457$^{(6)}$
gapd5	1704.46	6221$^{(5)}$	860.43	6235$^{(6)}$	111.0	6220$^{(4)}$	974	6280$^{(9)}$	13656.8	6358$^{(10)}$	108.8	6267$^{(8)}$
gapd6	922.94	12276$^{(3)}$	526.02	12293$^{(5)}$	120.7	12351$^{(8)}$	2158	12471$^{(11)}$	116969.0	12617$^{(12)}$	308.7	12333$^{(6)}$

Problem	ACO		CRH-GA		TSEC		PREC		PRSS		NBB		CPLEX 6.5	
	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet	CPU	Maliyet
gapa1	0.17	**1698**$^{(1)}$	1.0	**1698**$^{(1)}$	--	-	-	-	-	-	0.12	**1698**$^{(1)}$	-	**1698**$^{(1)}$
gapa2	0.61	**3235**$^{(1)}$	25.9	**3235**$^{(1)}$	-	-	-	-	-	-	0.06	**3235**$^{(1)}$	-	**3235**$^{(1)}$
gapa3	0.69	**1360**$^{(1)}$	97.1	**1360**$^{(1)}$	-	-	-	-	-	-	0.05	**1360**$^{(1)}$	-	**1360**$^{(1)}$
gapa4	126.58	**2623**$^{(1)}$	10.2	**2623**$^{(1)}$	-	-	-	-	-	-	0.16	**2623**$^{(1)}$	-	**2623**$^{(1)}$
gapa5	6.71	**1158**$^{(1)}$	22.2	**1158**$^{(1)}$	-	-	-	-	-	-	0.16	**1158**$^{(1)}$	-	**1158**$^{(1)}$
gapa6	158.5	2341$^{(2)}$	24.2	**2339**$^{(1)}$	-	-	-	-	-	-	0.28	**2339**$^{(1)}$	-	**2339**$^{(1)}$
gapb1	191.25	1846$^{(2)}$	51.2	**1843**$^{(1)}$	0.81	**1843**$^{(1)}$	0.95	**1843**$^{(1)}$	1.41	**1843**$^{(1)}$	4.18	**1843**$^{(1)}$	-	**1843**$^{(1)}$
gapb2	2605.55	3561$^{(3)}$	165.4	3553$^{(2)}$	3.13	**3552**$^{(1)}$	2.35	**3552**$^{(1)}$	2.59	**3552**$^{(1)}$	14.66	**3552**$^{(1)}$	-	**3552**$^{(1)}$
gapb3	115.78	**1407**$^{(1)}$	30.9	**1407**$^{(1)}$	0.22	**1407**$^{(1)}$	0.33	**1407**$^{(1)}$	0.18	**1407**$^{(1)}$	0.11	**1407**$^{(1)}$	-	**1407**$^{(1)}$
gapb4	5482.81	2849$^{(5)}$	381.8	2829$^{(3)}$	16.00	**2827**$^{(1)}$	18.97	**2827**$^{(1)}$	91.64	**2827**$^{(1)}$	235.85	**2827**$^{(1)}$	-	**2827**$^{(1)}$
gapb5	613.28	**1166**$^{(1)}$	189.3	**1166**$^{(1)}$	9.00	**1166**$^{(1)}$	3.64	**1166**$^{(1)}$	7.24	**1166**$^{(1)}$	0.17	**1166**$^{(1)}$	-	**1166**$^{(1)}$
gapb6	2865.04	2350$^{(4)}$	378.5	2340$^{(2)}$	5.19	**2339**$^{(1)}$	11.53	**2339**$^{(1)}$	18.21	**2339**$^{(1)}$	88.98	2340$^{(2)}$	-	**2339**$^{(1)}$
gapc1	192.13	**1931**$^{(1)}$	39.2	**1931**$^{(1)}$	0.6	**1931**$^{(1)}$	0.52	**1931**$^{(1)}$	1.34	**1931**$^{(1)}$	0.05	**1931**$^{(1)}$	2	**1931**$^{(1)}$
gapc2	429.09	3464$^{(5)}$	320.0	3457$^{(2)}$	3.7	**3456**$^{(1)}$	12.09	**3456**$^{(1)}$	2.12	**3456**$^{(1)}$	30.43	**3456**$^{(1)}$	35	**3456**$^{(1)}$
gapc3	122.43	1406$^{(3)}$	54.1	1403$^{(2)}$	3.0	**1402**$^{(1)}$	3.20	**1402**$^{(1)}$	1.86	**1402**$^{(1)}$	7.25	**1402**$^{(1)}$	9	**1402**$^{(1)}$
gapc4	1112.31	2825$^{(6)}$	304.6	2807$^{(2)}$	403.8	**2806**$^{(1)}$	2710.87	**2806**$^{(1)}$	48.47	2807$^{(2)}$	312.64	**2806**$^{(1)}$	249	**2806**$^{(1)}$
gapc5	193.85	1246$^{(4)}$	289.6	**1243**$^{(1)}$	22.5	**1243**$^{(1)}$	35.83	**1243**$^{(1)}$	17.99	1245$^{(3)}$	90.19	**1243**$^{(1)}$	8	**1243**$^{(1)}$
gapc6	1119.25	2411$^{(7)}$	1140.1	2396$^{(4)}$	301.8	**2391**$^{(1)}$	1268.83	**2391**$^{(1)}$	191.32	2394$^{(3)}$	968.66	**2391**$^{(1)}$	296	**2391**$^{(1)}$
gapd1	1.81	6625$^{(8)}$	327.4	6365$^{(5)}$	649.2	**6353**$^{(1)}$	83.62	**6353**$^{(1)}$	14.24	6356$^{(2)}$	349.93	**6353**$^{(1)}$	43	6358$^{(4)}$
gapd2	3.31	13197$^{(10)}$	814.7	12767$^{(7)}$	3564.8	12743$^{(2)}$	5712.49	**12742**$^{(1)}$	249.89	12745$^{(4)}$	2937.03	12745$^{(4)}$	62	12750$^{(6)}$
gapd3	174.46	6613$^{(11)}$	472.6	6373$^{(6)}$	2440.7	6349$^{(2)}$	988.07	**6348**$^{(1)}$	74.12	6373$^{(6)}$	2831.47	6349$^{(2)}$	132	6381$^{(8)}$
gapd4	1155.21	13024$^{(11)}$	1909.7	12536$^{(8)}$	5829.9	12440$^{(2)}$	5520.51	**12433**$^{(1)}$	246.32	12468$^{(7)}$	1896.80	12447$^{(4)}$	27	12457$^{(6)}$
gapd5	189.92	6484$^{(11)}$	902.3	6259$^{(7)}$	1591.9	6206$^{(3)}$	2254.88	**6192**$^{(1)}$	129.33	6235$^{(6)}$	2829.38	6200$^{(2)}$	60	6280$^{(9)}$
gapd6	1270.43	12951$^{(13)}$	3825.7	12386$^{(9)}$	1757.7	12277$^{(4)}$	5777.10	**12245**$^{(1)}$	518.30	12334$^{(7)}$	2375.42	12263$^{(2)}$	297	12393$^{(10)}$

DF: tabu arama [132], CGA: yapısal genetik algoritma [144], RLS: yerel arama ve tabu arama ile birleştirilmiş maks-min karınca sistemi [134], APT: yol birleştirme algoritması [136], ACO: karınca koloni optimizasyonu [141], CRH-GA: kısıt-oran sezgiseline dayalı genetik algoritma [142], TSEC: çıkarım zincirine dayalı tabu arama [133], PREC: çıkarım zincirine dayalı yol birleştirme yaklaşımı [140], PRSS: kaydırma ve çift kaydırma komşuluklarına dayalı yol birleştirme yaklaşımı [140], NBB: uygun çözüm üreten dal-sınır algoritması [125], CPLEX 6.5: [133].

Çevresel etkenler, test şartları, parametre optimizasyonu gibi birçok sebepten, farklı algoritmaların performansını direkt olarak karşılaştırmak kolay değildir. Önerilen algoritmaların GAP'nin çözümü üzerindeki performansı hakkında bir fikir edinebilmek için basit bir sıralama yöntemi kullanılmıştır. Tablo 3.12'de en iyi çözümler koyu renklendirilmiş ve sıralama parantez içinde belirtilmiştir. Bu bilgi, bir algoritmanın kaç kez 1, 2, 3 ve ≥4 sıralamasına girdiğini gösteren Şekil 3.4'te birleştirilmiştir.

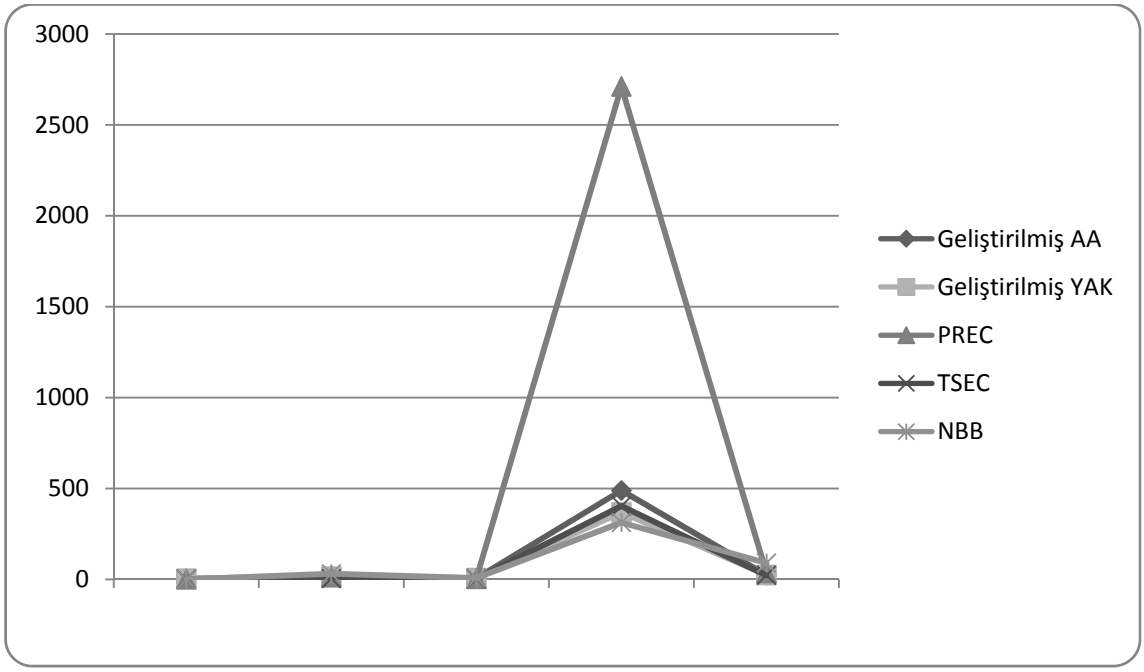


Şekil 3.4. Çözüm kalitesine göre algoritmaların sıralanması.

Şekil 3.4'teki sonuçlara göre PREC en iyi performansa sahip algoritmadır. Geliştirilmiş AA, Geliştirilmiş YAK, TSEC ve NBB ise çözüm kalitesi bakımından en iyi ikinci performansa sahip algoritmalar olarak sınıflandırılabilir. CPLEX 6.5 kesin çözüm yönteminin performansı ise gapc problemleri için iyi sonuçlar verse de gapd problemi üzerindeki performansı düşüktür.

Diğer taraftan bahsedilen algoritmaların (PREC, AA, YAK, TSEC, NBB) hesaplama süresi performansları aynı maliyet değerlerinin bulunduğu gapc1-gapc5 problemleri için Şekil 3.5'te sunulmuş olup görüldüğü gibi Geliştirilmiş AA ve YAK algoritması düşük CPU süresine sahiptir. Belirtmek gerekir ki PREC ve TSEC algoritmalarına ait CPU süreleri α_j parametresinin optimizasyonu sonrasında hesaplanmaktadır. Ancak Geliştirilmiş AA'na ait CPU süreleri α_j parametresinin en iyi değerine getirilme süresini

de içermektedir. Dolayısıyla PREC ve TSEC algoritmalarının CPU süresi gereksiniminin Geliştirilmiş AA'ndan çok daha yüksek olduğunu söyleyebiliriz.



Şekil 3.5. En iyi performansa sahip algoritmaların işlem süresi performansı.

Deneysel çalışmalar sonucunda önerilen algoritmaların gapa, gapb ve gapc (gapc6 hariç) problem kümelerinin çözümünde oldukça etkin olduğu ve koşmaların neredeyse tamamında en iyi çözüme ulaşılabildiği görülmektedir. Diğer taraftan önerilen algoritmalar gapd problem kümesinin çözümünde de diğer birçok algoritmaya göre oldukça başarılı sonuçlar elde etmiştir. Sonuç olarak yapılan geniş deneysel çalışmalar, Geliştirilmiş AA ve YAK algoritmasının karmaşık atama problemlerinin çözümünde büyük bir potansiyele sahip olduğunu göstermektedir.

3.6. Genelleştirilmiş Atama Problemi'nin Arı Algoritması ile Çözümünde Farklı Komşuluk Yapılarının Karşılaştırılması

Bu bölümde GAP için geliştirilmiş olan AA dört farklı komşuluk yapısı kullanılarak test edilmiş ve bu komşuluk yapılarının AA'nın performansı üzerindeki etkileri incelenmiştir.

Yapılan bilimsel yazın araştırması neticesinde kaydırma, çift kaydırma, değiştirme ve çıkarım zinciri komşulukları olmak üzere GAP'ne özel dört tip komşuluk yapısı belirlenmiştir. İlgili komşuluk yapılarının algoritmaya tek tek dâhil edilmesiyle algoritma çalıştırılmış ve her bir komşuluk yapısının bireysel performansı incelenmiştir.

3.6.1. Deneysel Çalışma

Geliştirilmiş AA C# dilinde kodlanarak 2.20 GHz CPU, 2.00 GB RAM özelliklere sahip Intel Pentium CoreDuo PC kullanılarak gapa, gapb test problemlerine uygulanmıştır. Algoritma parametreleri bilimsel yazında verilen genel öneriler ve deneyimlere göre belirlenmiş olup Tablo 3.14'te verildiği gibidir.

Tablo 3.13. GAP'nde komşuluk yapısı karşılaştırması için geliştirilmiş AA parametrelerinin değerleri.

Parametre	Değer
S	500
P	50
e	10
nep	10
nsp	5
ÇZ-Uzunluğu	70
MaksIter	1000
limit	50
α_{jb}	1
adımbüyüklüğüazalışı	0.1
adımbüyüklüğüartışı	0.01

Her komşuluk yapısı için test problemlerinin 10 kez çalıştırılması sonucunda elde edilen minimum, ortalama, maksimum ve standart sapma değerleri, optimum CPLEX 6.5 sonuçları ile birlikte Tablo 3.15'te; bu verilere ait CPU sürelerinin (sn.) minimum, ortalama, maksimum ve standart sapma değerleri ise Tablo 3.16'da verilmiştir.

Tablo 3.14. Farklı komşuluk yapılarının karşılaştırılması.

	Kaydırma				Çift kaydırma				CPLEX 6.5
	min()	ort	maks	ss	min()	ort	maks	ss	
gapa1	1698 (10)	1698.0	1698	0.0	1698 (10)	1698.0	1698	0.0	1698
gapa2	3235 (10)	3235.0	3235	0.0	3235 (9)	3235.1	3236	0.3	3235
gapa3	1360 (10)	1360.0	1360	0.0	1360 (10)	1360.0	1360	0.0	1360
gapa4	2623 (10)	2623.0	2623	0.0	2623 (10)	2623.0	2623	0.0	2623
gapa5	1158 (10)	1158.0	1158	0.0	1158 (10)	1158.0	1158	0.0	1158
gapa6	2339 (10)	2339.0	2339	0.0	2339 (10)	2339.0	2339	0.0	2339
gapb1	1855 (1)	1866.2	1879	7.8	1843 (2)	1857.7	1881	12.1	1843
gapb2	3590 (1)	3610.8	3673	24.8	3557 (1)	3565.4	3571	4.4	3552
gapb3	1407 (2)	1409.6	1413	2.3	1407 (3)	1409.6	1412	2.0	1407
gapb4	2859 (1)	2886.2	2919	18.3	2842 (1)	2853.5	2865	6.4	2827
gapb5	1167 (2)	1170.1	1174	2.4	1166 (1)	1169.5	1171	1.7	1166
gapb6	2345 (1)	2349.8	2355	3.4	2343 (2)	2345.4	2348	1.7	2339

	Değiştirme				Çıkarım zinciri			
	min()	ort	maks	ss	min()	ort	maks	ss
gapa1	1703 (1)	1711.6	1721	5.1	1698 (4)	1699.4	1702	1.4
gapa2	3252 (1)	3264.3	3285	10.0	3235 (7)	3235.5	3237	0.8
gapa3	1394 (1)	1406.0	1421	9.2	1360 (8)	1360.2	1361	0.4
gapa4	2710 (1)	2728.3	2753	16.0	2635 (1)	2665.7	2698	23.4
gapa5	1247 (1)	1264.9	1290	12.8	1158 (10)	1158.0	1158	0.0
gapa6	2511 (1)	2559.3	2613	32.9	2400 (1)	2442.0	2485	26.7
gapb1	1874 (1)	1889.8	1906	10.0	1857 (1)	1888.5	1915	17.4
gapb2	3588 (1)	3604.0	3616	9.7	3628 (1)	3648.5	3677	14.4
gapb3	1438 (1)	1470.0	1504	20.5	1407 (6)	1407.8	1411	1.3
gapb4	2988 (1)	3010.2	3070	24.1	2846 (1)	2865.2	2881	12.2
gapb5	1249 (1)	1275.4	1307	19.4	1167 (2)	1168.9	1172	1.5
gapb6	2586 (1)	2653.1	2698	37.8	2346 (1)	2352.5	2357	3.5

* min() sütunundaki parantez içindeki değerler algoritmanın 10 kez çalıştırılması sonucunda ilgili minimum değerin kaç kez bulunduğunu ifade etmektedir.

Tablo 3.15. Farklı komşuluk yapılarının işlem süresi bakımından karşılaştırılması.

	Kaydırma				Çift kaydırma			
	min	ort	maks	ss	min	ort	maks	ss
gapa1	3.23	4.08	4.59	0.48	6.65	96.04	231.81	71.19
gapa2	8.21	12.69	16.67	2.67	44.81	286.03	923.00	279.89
gapa3	4.07	5.61	8.95	1.48	5.31	25.56	90.10	32.42
gapa4	11.71	54.30	141.85	41.46	37.84	168.02	341.89	114.71
gapa5	5.53	8.13	10.34	1.56	5.42	7.58	10.25	1.26
gapa6	14.76	31.57	52.23	11.27	54.84	87.68	156.37	31.56
gapb1	19.21	60.07	103.95	26.48	23.75	111.69	202.09	63.62
gapb2	6.46	81.50	193.12	61.33	113.59	592.79	1036.93	324.82
gapb3	12.40	387.89	1106.04	333.10	11.90	123.87	333.17	95.78
gapb4	17.68	351.59	805.62	289.03	92.25	399.81	904.64	242.25
gapb5	71.87	421.04	799.14	236.44	17.89	107.01	209.89	82.32
gapb6	204.12	676.22	1577.96	455.78	103.92	694.48	1162.64	322.97
	Değiştirme				Çıkarım zinciri			
	min	ort	maks	ss	min	ort	maks	ss
gapa1	32.37	148.41	534.67	146.31	60.59	273.52	436.43	129.78
gapa2	53.98	166.86	328.20	88.12	180.54	628.99	1119.48	307.40
gapa3	34.37	164.77	357.71	88.45	73.12	237.13	498.65	131.99
gapa4	83.40	139.95	214.64	45.53	268.53	653.85	1159.01	322.72
gapa5	60.00	102.31	170.65	33.69	14.78	68.00	160.04	42.60
gapa6	101.53	211.02	406.03	93.70	260.25	642.56	1174.50	303.52
gapb1	40.32	101.61	191.20	45.09	54.85	668.43	1576.18	396.06
gapb2	66.00	235.14	448.14	108.23	446.62	838.03	1591.14	359.32
gapb3	49.65	193.90	468.75	123.50	98.53	352.28	683.46	210.83
gapb4	89.70	155.79	237.20	49.10	934.93	1847.56	2925.28	731.27
gapb5	53.75	144.96	248.07	58.23	183.31	564.74	929.70	232.63
gapb6	67.06	161.28	216.82	45.72	452.45	1299.56	2919.26	841.56

Elde edilen sonuçların anlamlı bir şekilde karşılaştırılabilmesi için öncelikle uygun bir hipotez testinin seçilmesi gerekmektedir. Uygun hipotez testinin seçimi aşağıdaki durumlara bağlıdır (Öztuna ve Elhan):

- Verilerin ölçüm biçimi: Hipotez testinin seçimini etkileyen en önemli faktörlerden biri olan ölçüm biçimi, nitel (sayımla) ya da nicel (ölçümle) olarak ikiye ayrılmaktadır.
- İncelenen grupların bağımlı ya da bağımsız olması: İncelenen grupların bağımsız olması, grupların ayrı bireylerden oluşması, diğer bir deyişle bir grupta bulunan bir bireyin diğer grupta bulunmaması demektir. Bir birey üzerinde birden çok gözlem yapıldığında gruplar bağımlı olmaktadır.

- Verilerin dağılımı: İstatistiksel analiz yapılırken dağılımın özelliği çok önemlidir. Çünkü parametrik testlerin uygulanabilmesi için dağılımın normal ya da normale yakın olması gerekmektedir.
- Örneklem büyüklüğü: Gruplardaki birey sayısı arttıkça kullanılan testin gücü ve güvenilirliği de artar. Gruplardaki birey sayısı fazla ise verilerin normal dağılıma uyma ihtimali artar, dolayısıyla parametrik test kullanma şansı artmış olur. Gruplardaki birey sayısı az olduğunda ise (30'un altında) genellikle parametrik olmayan testler tercih edilir.

Bilimsel yazında iki bağımsız örnek grubunun karşılaştırılması için farklı yöntemler geliştirilmiştir. Bu yöntemler parametrik ve parametrik olmayan yöntemler olmak üzere iki gruba ayrılmaktadır. Veri dağılımının normal, varyansların eşit, birbirinden bağımsız ve rastgele seçilmiş olan örnek grubu sayısının 30'dan fazla olduğu durumlarda parametrik testler (t-testi, varyans analizi gibi), veri dağılımının normal olmadığı ve veri sayısının 30'dan az olduğu nicel verilerin varlığında ise parametrik olmayan testler (Wilcoxon sıralı toplam testi, Mann-Whitney U testi, Kruskal-Wallis varyans analizi gibi) kullanılmaktadır.

Yukarıda verilen bilgiler ışığında, komşuluk karşılaştırması için elde edilen sonuçların nicel ve bağımlı bir yapıya sahip olması, verilerin normal dağılıma uymaması ve örneklem büyüklüğünün 30'un altında olması sebebiyle komşuluk yapılarının karşılaştırılmasında Wilcoxon sıralı toplam testinin kullanılmasına karar verilmiştir. Wilcoxon sıralı toplam testi Wilcoxon [146] tarafından geliştirilmiş olup iki grubun ortanca değerleri arasında fark olup olmadığını araştırmaktadır. Ancak ortanca değerler arasındaki farkın istatistiksel olarak anlamlı olup olmadığını söyleyebilmek için p değerlerine bakmak gerekir. Eğer p değeri α=0,05 önemlilik değerinden küçükse, karşılaştırılan ortanca değerler arasında istatistiksel olarak anlamlı bir farklılık var demektir.

Her problem tipi için komşuluk yapıları çiftler halinde ele alınarak elde edilen eşitsizlik ilişkisine dayalı Wilcoxon sıralı toplam testi sonuçları Tablo 3.17'de verilmiştir.

Tablo 3.16. Wilcoxon sıralı toplam testi ile komşuluk yapılarının değerlendirilmesi.

		Kaydırma, Çift kaydırma	Kaydırma, Değiştirme	Kaydırma, Çıkarım zinciri	Çift kaydırma, Değiştirme	Çift kaydırma, Çıkarım zinciri	Değiştirme, Çıkarım zinciri
gapa1	ortanca-1	-	-	-	-	-	1711.5
	ortanca-2	-	-	-	-	-	1699.5
	p	-	-	-	-	-	0.0002
gapa2	ortanca-1	-	-	-	3235	3235	3263
	ortanca-2	-	-	-	3263	3235	3235
	p	-	-	-	0.0002	0.4274	0.0002
gapa3	ortanca-1	-	-	-	-	-	1406
	ortanca-2	-	-	-	-	-	1360
	p	-	-	-	-	-	0.0002
gapa4	ortanca-1	-	-	-	-	-	2722.5
	ortanca-2	-	-	-	-	-	2664.5
	p	-	-	-	-	-	0.0002
gapa5	ortanca-1	-	-	-	-	-	-
	ortanca-2	-	-	-	-	-	-
	p	-	-	-	-	-	-
gapa6	ortanca-1	-	-	-	-	-	2554.5
	ortanca-2	-	-	-	-	-	2445
	p	-	-	-	-	-	0.0002
gapb1	ortanca-1	1864.5	1864.5	1864.5	1859	1859	1890.5
	ortanca-2	1859	1890.5	1890	1890.5	1890	1890
	p	0.0963	0.0006	0.0052	0.0003	0.0017	0.9097
gapb2	ortanca-1	3603.5	3603.5	3603.5	3565.5	3565.5	3605
	ortanca-2	3565.5	3605	3646	3605	3646	3646
	p	0.0002	0.7913	0.0022	0.0002	0.0002	0.0002
gapb3	ortanca-1	1409	1409	1409	1410	1410	1471.5
	ortanca-2	1410	1471.5	1407	1471.5	1407	1407
	p	0.9698	0.0002	0.0539	0.0002	0.0696	0.0002
gapb4	ortanca-1	2881	2881	2881	2854.5	2854.5	3005.5
	ortanca-2	2854.5	3005.5	2867.5	3005.5	2867.5	2867.5
	p	0.0003	0.0002	0.0073	0.0002	0.0588	0.0002
gapb5	ortanca-1	1170	1170	1170	1170	1170	1275
	ortanca-2	1170	1275	1169	1275	1169	1169
	p	0.6232	0.0002	0.3258	0.0002	0.3075	0.0002
gapb6	ortanca-1	2350	2350	2350	2345.5	2345.5	2658.5
	ortanca-2	2345.5	2658.5	2353	2658.5	2353	2353
	p	0.0082	0.0002	0.1212	0.0002	0.0006	0.0002

* '-' komşuluk yapıları ile elde edilen değerlerin kendi içinde farklılık göstermemesi nedeniyle Wilcoxon sıralı toplam testinin gerçekleştirilemediğini göstermektedir.
* α=0.05

Yapılan analizler neticesinde gapa1, gapa3, gapa4, gapa6 problemleri için çıkarım zinciri komşuluk yapısının değiştirmeye göre daha üstün olduğu belirlenmiştir. gapa2

problemi için önceki problemler için yapılan tespitin yanı sıra çift kaydırma komşuluk yapısının değiştirmeye göre daha iyi performans gösterdiği belirlenmiştir. gapa5 problem tipi için kaydırma, çift kaydırma ve çıkarım zinciri komşuluklarıyla elde edilen çözümlerin kendi içinde farklılık göstermemesi sebebiyle Wilcoxon sıralı toplam testi gerçekleştirilememiştir. Wilcoxon sıralı toplam testinin yapılamadığı ikili karşılaştırmalar için grup ortalamalarına bakıldığında ise gapa1, gapa3, gapa4 ve gapa6 problemleri için en iyi performans gösteren komşuluk yapısının kaydırma ve çift kaydırma, ikinci en iyi performansa sahip yapının ise çıkarım zinciri olduğu görülmektedir. gapa2 problemi için kaydırma komşuluk yapısın en iyi, çift kaydırma komşuluğunun ikinci en iyi ve son olarak çıkarım zinciri komşuluk yapısının üçüncü en iyi performansa sahip olduğu görülmüştür. Hiçbir ikili karşılaştırmada Wilcoxon sıralı toplam testinin gerçekleştirilemediği gapa5 problemi için ise kaydırma, çift kaydırma ve çıkarım zinciri komşuluk yapılarının eşit performansa sahip olup değiştirme komşuluk yapısından daha üstün olduğu gözlemlenmiştir.

Gapb problemlerini tek tek ele alacak olursak; gapb1 problem tipi için kaydırma komşuluk yapısının değiştirme ve çıkarım zinciri komşuluk yapılarına göre daha üstün; çift kaydırma komşuluk yapısının ise yine değiştirme ve çıkarım zinciri komşuluk yapılarına göre daha üstün olduğu belirlenmiş; ancak kaydırma ve çift kaydırma komşuluk yapıları arasında ve değiştirme ve çıkarım zinciri komşuluk yapıları arasında anlamlı bir farklılık tespit edilememiştir. gapb2 problemleri için çift kaydırma komşuluk yapısının diğer komşuluk yapılarına göre daha iyi performans gösterdiği belirlenmiştir. gapb3 ve gapb5 problemleri için bütün komşuluk yapılarının değiştirme komşuluk yapısından daha üstün olduğu tespit edilmiş ancak bu komşuluklar arasında anlamlı bir farklılık belirlenememiştir. gapb4 problemleri için çift kaydırma ve çıkarım zinciri komşuluk yapılarının kaydırma ve değiştirme komşuluk yapılarından daha iyi olduğu belirlenmiş ancak çift kaydırma ve çıkarım zinciri komşuluk yapıları arasında anlamlı bir fark tespit edilememiştir. gapb6 problemi için bütün komşuluk yapılarının değiştirme komşuluk yapısından daha üstün olduğu tespit edilmiş, diğer taraftan çift kaydırma komşuluk yapısının kaydırma ve çıkarım zincirine göre daha iyi performans gösterdiği belirlenmiştir.

Sonuç olarak ele alınan test problemleri için çift kaydırma komşuluk yapısının en iyi performansa, kaydırma ve çıkarım zinciri komşuluk yapılarının ikinci en iyi

performansa, değiştirme komşuluk yapısının ise en düşük performansa sahip olduğu tespit edilmiştir.

4. BÖLÜM
ÇİFT TARAFLI MONTAJ HATTI DENGELEME PROBLEMİ ÇÖZÜM YAKLAŞIMLARI

4.1. Giriş

Tez çalışmasının bu bölümünde ÇTMHDP'nin çözümü için YAK ve AA algoritmalarından faydalanılacaktır. ÇTMHDP'nin detayları ve bilimsel yazındaki çözüm yaklaşımları verildikten sonra probleme ait farklı kısıtlar altında ilgili algoritmalarla elde edilen geniş deneysel çalışma sonuçları sunulacaktır.

Diğer taraftan AA, bulanık çok amaçlı ÇTMHDP'nin çözümü için kullanılacak ve bulanık amaçlar farklı teknikler altında incelenerek bu tekniklerin algoritma performansı üzerindeki etkisi incelenecektir.

4.2. Montaj Hattı Dengeleme Problemi

İşletmelerin temel amaçları verimlilik düzeyini yükseltmek, kapasite ve kaliteyi artırmak, maliyetleri düşürmek ve çalışma ortamını insancıllaştırmaktır. Bu amaçlara ulaşmak için ise kullanılan işgücü, makine, malzeme ve teçhizattan oluşan iş yöntemlerinin yeniden tasarlanması gerekmektedir. Sürekli üretim sistemlerinde, üretimin birimler halinde gerçekleştirildiği ve kitle talebin olduğu durumlarda, yüksek üretim hızıyla talebi karşılamanın en makul yolu montaj hatlarının yapılandırılmasıdır. Montaj hattı tasarımındaki ana amaçlardan biri, her iş istasyonuna eşit miktarda iş dağıtımını yapabilmektir. Bu amaç doğrultusunda işler, istasyon süreleri birbirine eşit ya da çok yakın olacak şekilde istasyonlar arasında paylaştırılır. Montaj hattı dengeleme ile işler gruplandırılarak istasyonlar kurulur, istasyonların işlem süreleri birbirine yakın hale getirilir ve bu şartlar altında montaj hattının aksamadan çalışması sağlanarak kaynaklardan maksimum fayda elde edilir. Dengenin sağlanamadığı durumda ise bazı

istasyonlarda diğerlerinden daha fazla iş yükü olacağı için, verimlilikte düşüşlerin olması ve bir takım kayıpların ortaya çıkması kaçınılmazdır.

Montaj Hattı Dengeleme Problemi (MHDP) çevrim süresi ve iş sayısının verildiği varsayımı altında öncelik ilişkileri ve çevrim süresi kısıtlarına uyarak bir veya daha fazla amacı eniyileyecek şekilde, işlerin istasyonlara atanması problemidir. Öncelik ilişkisi, montaj prosesindeki işlerin hangi sıra ile gerçekleştirileceğini gösterirken; çevrim süresi, bir istasyonda yapılması gereken işlerin tamamlanabilmesi için ürünün o istasyonda kalabileceği en uzun süreyi temsil etmektedir. MHDP, Karp [147] tarafından ispatlandığı üzere NP-zor yapıya sahip kombinatoryal bir problemdir.

MHDP çeşitli kriterlere göre aşağıdaki gibi sınıflandırılabilir:

- Amaç fonksiyonuna göre [148]
  - Tip-1: Belirli bir çevrim süresi dâhilinde istasyon sayısını minimize etmeyi amaçlar.
  - Tip-2: Belirli bir istasyon sayısı dâhilinde çevrim süresini minimize etmeyi amaçlar.
  - Tip-3: İşyükleri arasındaki dengeyi maksimize etmeyi amaçlar.
  - Tip-4: İşler arasındaki öncelik ilişkilerini dikkate alarak birbiriyle ilişkili işlerin aynı istasyonda yer almasını sağlamayı amaçlar.
  - Tip-5: Tip-3 ve tip-4'te kullanılan amaçları aynı zamanda maksimize etmeyi amaçlar.
  - Tip-E: Çevrim süresi ve istasyon sayısını aynı zamanda minimize ederek hattın verimliliğini maksimize etmeyi amaçlar.
  - Tip-F: Belirli bir iş sayısı ve çevrim süresi için uygun bir hat dengelemesi olup olmadığını bulmayı amaçlar.
- Problemin yapısına göre
  - Scholl [149] ve Becker ve Scholl [150]'e göre
    - Tek modelli montaj hattı dengeleme: Tek tip ürün ya da modelin üretildiği hatlardır.
    - Karma modelli montaj hattı dengeleme: Aynı anda birden fazla benzer tipteki modelin karma olarak üretildiği hatlardır. Karma

modelli üretimin en önemli faydası, müşteri isteklerini karşılamak üzere değişik modellerin sürekli olarak üretilmesi ve büyük bitmiş ürün stoklarını gerektirmemesidir.

— Çok modelli montaj hattı dengeleme: Değişik model veya ürünlerin üretildiği hatlardır. Belirli bir zamanda bir model parti halinde üretilir ve arkadan diğer modellerin üretimine geçilir.

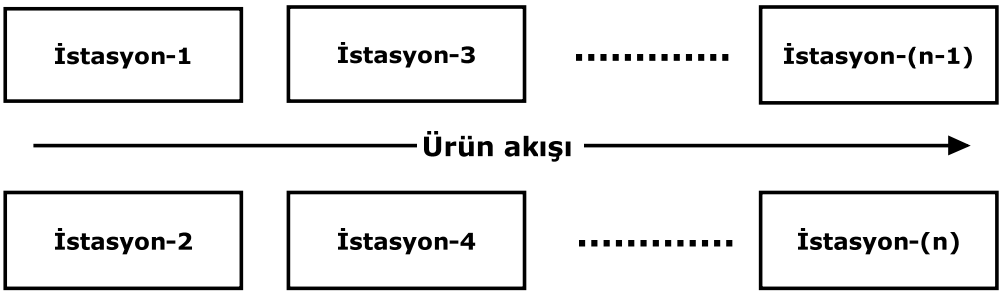
- Baybars [151]'a göre
  - — Basit montaj hattı dengeleme: Kesin ve birbirinden bağımsız işlem sürelerine, seri yerleşime, tek yönlü ve özdeş istasyonlara sahip tek bir ürünün üretildiği hatlardır.
  - — Genel montaj hattı dengeleme: Basit montaj hattı dengelemeye göre daha gerçekçi olup paralel, U tipi ve çift taraflı montaj hattı dengeleme gibi basit montaj hattı dengelemeye girmeyen bütün problemleri içerir.

Diğer taraftan bilimsel yazında MHDP üzerine yapılan bazı çalışmalar şu şekilde özetlenebilir. Salveson [152] doğrusal programlama, Jackson [153] dal ve sınır yaklaşımı, Bowman [154] tam sayılı programlama, Held ve ark. [155] dinamik programlama üzerinde çalışırken Dar-El [156], Dar-El ve Rubinovitch [157], Baybars [158] çeşitli sezgisel yöntemler geliştirmişlerdir. Literatürdeki meta-sezgisel tekniklere bakıldığında ise benzetimli tavlama [159], tabu arama [160, 161], genetik algoritma [162-166] yaklaşımlarını kullanarak montaj hattı dengeleme problemini çözmüşlerdir. Montaj hattı problemlerinin sınıflandırılması ise Baybars [151], Ghosh ve Gagnon [167], Kim ve ark. [148], Erel ve Sarin [168], Scholl [149], Becker ve Scholl [150], Boysen ve ark. [169] çalışmalarının konusu olmuştur.

4.3. Çift Taraflı Montaj Hattı Dengeleme Problemi

Montaj hatları üzerindeki diğer bir sınıflandırma ise hattın tek ya da çift yönünün kullanılmasına bağlı olarak tek ve çift taraflı hatlar olarak karşımıza çıkmaktadır. Tek taraflı montaj hattı, montaj hattının sadece bir yönünün kullanıldığı, hat dengeleme probleminin temel ve basit hali olup en yaygın incelenen tipidir. Çift taraflı montaj hattında ise aynı ürün üzerindeki farklı montaj işleri hattın sağ ve sol yönünde paralel olarak gerçekleştirilmektedir. Bu tip montaj hatları genel olarak otobüs ve kamyon gibi

büyük boyutlu ürünlerin üretildiği işletmelerde görülmektedir. Tek Taraflı Montaj Hattı Dengeleme Problemi'nin (TTMHDP) çözümü için birçok algoritma ve sezgisel yöntem önerilmiştir, fakat ÇTMHDP'nin çözümüne yönelik çalışmalar oldukça azdır. Bunun nedeninin ÇTMHDP'nin, tek taraflı olana göre çok daha zor olmasından kaynaklandığı düşünülmektedir [170]. Nitekim TTMHDP'nde önemli olan hangi işin hangi istasyonda işleneceği iken ÇTMHDP'nde hem hangi işin hangi istasyonda işleneceği hem de hangi sırayla işleneceği belirlenmelidir. Diğer bir deyişle ÇTMHDP'nde aralarında yakın öncelik ilişkisi olan iki iş karşılıklı istasyonlara atandığında biri tamamlanmadan diğeri başlayamayacağından boş zaman ortaya çıkabilecektir. Dolayısıyla hattı dengelerken işlerin sırasına bağlı tamamlanma zamanları dikkate alınmalıdır. Böylece problem daha karmaşık ve çözülmesi zor bir hâl almaktadır. Bartholdi [5] çalışmasında da belirtildiği gibi ÇTMHDP NP-Tam bir yapıya sahiptir. Diğer taraftan çift taraflı hatların kullanımı, tek taraflı hatlara göre hat uzunluğunun kısalması, üretim süresinin kısalması, araçların her iki yönden paylaşılması sebebiyle daha az araç maliyeti, elde bulundurma maliyetinin, işçi hareketlerinin ve kurulum süresinin kısalması gibi birçok avantaja sahiptir [5]. Şekil 4.1'de bir çift taraflı montaj hattı şematik olarak gösterilmiştir.



Şekil 4.1. Çift taraflı montaj hattı.

4.3.1. Çift Taraflı Montaj Hattı Dengeleme Problemi Kısıtları

Çift taraflı montaj hattının tek taraflı montaj hattından temel farkı işlerin operasyon yönü kısıtı olmasıdır. Bazı montaj işleri iki yönden birini tercih ederken bazıları ise herhangi bir yönde yapılabilmektedir. Bu durumda işler üç şekilde sınıflandırılır: sağ(R), sol(L) ve herhangi biri (E). Örneğin bir kamyon montaj hattında benzin deposu ve hava filtresinin montajı sadece sol yönde; akü, hava tankı ve egzoz montajı sadece sağ yönde yapılabilecek işler iken aks, pervane ve radyatör montajı hattın herhangi bir yönünde yapılabilmektedir [170].

TTMHDP sadece öncelik ilişkileri ve çevrim süresi kısıtı olmak üzere iki temel kısıt içerirken ÇTMHDP için yalnızca bu kısıtları ele almak yeterli değildir. Çift taraflı montaj hattında kullanılan diğer kısıtlar şöyle sıralanabilir:

- Konumsal kısıtlar: Belirli bir işin işletmenin yerleşiminden dolayı önceden belirlenmiş bir istasyona atanması gerektiğini belirtir. Kamyon ve otobüs gibi büyük boyutlu ürünlerin üretildiği montaj hatlarında, bazı istasyonlar ağır makineler içerebilir ve tasarım aşamasında kurulduktan sonra yerinin değiştirilmesi çok zordur. Bundan dolayı bu makinelerde yapılacak işler mutlaka bu istasyonlara atanmalıdır.
- Bölgesel kısıtlar: Hangi işlerin aynı istasyona atanması (pozitif bölgesel) ve hangi işlerin aynı istasyona atanmaması (negatif bölgesel) gerektiğini gösterir. Pozitif bölgesel kısıta sahip işler ortak bir teçhizat ya da beceri gerektirirken, negatif bölgesel kısıta sahip işler aynı istasyona atandığında istasyon alanı yetersizliği ortaya çıkabilir. Pozitif bölgesel kısıta sahip işler karşılıklı iki istasyona da atanabilirken, negatif bölgesel kısıtlar karşılıklı iki istasyona da atanmamalıdır.
- Senkronizasyon kısıtları: Hattın her iki yönünde aynı anda gerçekleştirilmesi gereken, aynı süreye sahip işleri tanımlar. Bu kısıt türüne örnek olarak kamyon kabinine ait üst panelin aynı anda her iki yönden yerleştirilmesi verilebilir.

4.4. Çift Taraflı Montaj Hattı Dengeleme Problemi Çözüm Yaklaşımları

Çift taraflı montaj hatlarının tasarımı ve hattın dengelenmesi ilk olarak Bartholdi [5] tarafından incelenerek İlk Uygun Kural adı verilen basit bir atama kuralı önerilmiş ve temel olarak interaktif bir program üzerine yoğunlaşılmıştır. Kim ve ark. [171] ise ÇTMHDP'ne ait ilk matematiksel modelleri sunarak hat uzunluğunun, istasyon sayısının ve iş yükü sapmalarının en küçüklenmesi gibi farklı amaçları değerlendirmişlerdir.

Kim ve ark. [170] verilen bir çevrim süresi dâhilinde istasyon sayısını minimize etmeyi amaçlayan konumsal kısıtlara sahip ÇTMHDP'nin çözümü için genetik algoritmayı kullanmışlardır. Uygun olmayan çözümlere izin veren algoritmada genetik algoritma operatörleri olarak ÇTMHDP'ne özel olarak geliştirilmiş, yapılandırılmış tek nokta çaprazlama ve mutasyon kullanılmıştır. Önerilen algoritma otomotiv sektöründe faaliyet

gösteren bir firmaya uygulanmış, diğer taraftan algoritmanın performansı bilimsel yazındaki çeşitli test problemleri üzerinde de denenmiştir. Küçük boyutlu problemler için, Kim ve ark. [172] çalışmasındaki tamsayılı programlama yaklaşımı ile elde edilen optimum çözümlerle karşılaştırma yapılmış ve önerilen yaklaşımla optimum çözümlerin daha kısa sürede bulunduğu belirtilmiştir. Büyük boyutlu problemler için ise tamsayılı programlama yaklaşımıyla optimum çözümler bulunamadığından Bartholdi [5] çalışmasında verilen İlk Uygun Kural sezgiseli ile karşılaştırma yapılmış ve yine önerilen algoritmanın daha iyi performans gösterdiği belirtilmiştir.

Lee ve ark. [173] geliştirdikleri, tek bir işi atamak yerine işleri gruplar halinde atamaya dayanan, grup atama prosedürü ile aynı kurulum ya da teçhizat gerektiren birbiriyle ilişkili işlerin aynı istasyona atanmasını sağlamaya çalışırken aynı zamanda da bu işler arasındaki boş zamanı minimize etmeyi amaçlamaktadırlar. TTMHDP için geliştirilen en uzun işlem süresine sahip işin, kendinden sonra gelen iş sayısı en fazla olan işin, kendinden hemen sonra gelen iş sayısı en fazla olan işin ve maksimum sıralı konumsal ağırlık değerine sahip işin seçimi şeklinde belirlenen atama kuralları ÇTMHDP için güncellenerek büyük boyutlu test problemleri üzerinde önerilen yöntemle karşılaştırılmıştır. Karşılaştırma sonucunda geliştirilen yöntem birbiriyle ilişkili işlerin aynı istasyona atanması ve bu işler arasındaki boş zamanın minimize edilmesi amacına göre bütün problem örneklerinde diğer sezgisel kurallardan daha iyi; istasyon sayısı, çevrim süresi ve hattın verimliliği açısından ise bazı problemler için daha iyi performans göstermiştir.

Lapierre ve Ruiz [174] çift taraflı montaj hatlarının yönetimi ve dengelenmesi için öncelik tabanlı bir sezgisel yöntem geliştirerek endüstriyel bir uygulama gerçekleştirmişlerdir. Firmaya özel olarak oluşturulan ÇTMHDP'nin bilimsel yazındaki problemlerden farkı, iki farklı yükseklik ve alt montaj hatlarına sahip olmasıdır. Bu özelliklere uygun 248 işe sahip yeni bir veri kümesi oluşturulmuş ve yapılan deneysel çalışmada, geliştirilen yöntemle elde edilen çözümler firmanın deneyimlerine dayanarak oluşturduğu çözümle karşılaştırılmıştır. İstasyon sayısı açısından önerilen yöntemin daha iyi sonuçlar verdiği ancak firmaya ait çözümde daha iyi bir işyükü dağılımının söz konusu olduğu görülmüştür.

Simaria ve Vilarinho [175] karınca koloni optimizasyonu algoritmasını kullanarak bölgesel ve senkronizasyon kısıtlarına sahip Çift Taraflı Karma Model Montaj Hattı Dengeleme Problemi'ni (ÇTKMMHDP) çözen ilk çalışmayı gerçekleştirmişlerdir. Problemin amacı istasyon sayısını minimize etmek olup, ek amaçlar istasyonlar arasındaki işyükünü dengelemek, farklı modeller için istasyonlardaki işyükünü dengelemek ve montaj planlayıcısının gruplama tercihlerini sağlamaktır. Gruplama tercihleri, montaj hattını planlayan kişinin tercihleri doğrultusunda bazı işlerin aynı istasyonda gruplandırılması şeklinde tanımlanmış olup pozitif bölgesel kısıtlarda olduğu gibi bir zorunluluk söz konusu değildir. Simaria ve Vilarinho [176] bölgesel ve senkronizasyon kısıtlarına sahip ÇTKMMHDP'nin yapısına uygun bir matematiksel model sunarak önceki çalışmalarında önerdikleri karınca koloni optimizasyonu algoritması üzerinde geniş bir deneysel çalışma gerçekleştirmişlerdir. Aynı hat üzerinde eşzamanlı olarak montajı yapılacak iki modele ve 14 işe sahip sayısal bir örnek oluşturularak algoritmanın performansı incelenmiştir. Ayrıca büyük boyutlu test problemleri için tek modelli, bölgesel ve senkronizasyon kısıtları içermeyen ÇTMHDP çözülmüş, elde edilen sonuçlar grup atama prosedürü [173] ile karşılaştırılmış ve çok daha iyi sonuçlar elde edilmiştir.

Hu ve ark. [177] ÇTMHDP'nin çözümü için, geliştirdikleri istasyon tabanlı atama prosedürü ile Hoffmann sezgiselinin [178] kombinasyonunu önermişlerdir. Küçük boyutlu problemlerle gerçekleştirilen deneysel çalışmada 13 problemin 10'unda alt sınır değerine ulaşılmıştır.

Baykasoglu ve Dereli [179] çalışmalarında bölgesel kısıtlara sahip ÇTMHDP'nin çözümü için karınca koloni optimizasyonu algoritmasını kullanmışlardır. Problemin amacı verilen çevrim süresi dâhilinde istasyon sayısını minimize etmek ve mümkünse birbiriyle ilişkili işlerin aynı istasyona atanmasını sağlamaktır. Genetik algoritma [170] ve grup atama prosedürü [173] ile küçük ve büyük boyutlu test problemleri üzerinde bölgesel kısıtlar olmadan yapılan karşılaştırmalar önerilen yöntemin daha iyi sonuçlar elde ettiğini göstermiştir.

Wu ve ark. [180] ÇTMHDP'nde istasyon sayısının en küçüklenmesini amaçlayan bir matematiksel model önererek dal-sınır algoritmasıyla çözüm yoluna gitmişlerdir. TTMHDP için geliştirilen iş-tabanlı dal sınır algoritması ÇTMHDP'ne uyarlanmış,

ancak ağaç yapısının aynı boyuttaki bir TTMHDP'ndeki yapıya göre çok büyük olması sebebiyle iş atama kuralları ile ağacın boyutu küçültülmüştür. Küçük ve büyük boyutlu bazı test problemleri üzerinde algoritmanın etkinliği test edilmiş ve problemlerin tamamında optimum çözüm elde edilmiştir.

Kim ve ark. [181] verilen istasyon sayısı dâhilinde çevrim süresini minimize edecek bir matematiksel model geliştirerek genetik algoritmaya dayalı bir çözüm yöntemi önermişlerdir. Geliştirilen matematiksel model ile küçük boyutlu test problemlerinin optimum çözümleri elde edilmiş, ancak daha büyük boyutlu problemlerin matematiksel modelle çözümü mümkün olmadığından genetik algoritma kullanılmıştır. Büyük boyutlu problemler için genetik algoritma ile elde edilen sonuçlar Bartholdi [5] tarafından geliştirilen İlk Uygun Kural ve Kim ve ark. [170] çalışmasında önerilen genetik algoritmanın performansıyla karşılaştırılmıştır. Çözüm kalitesi ve yakınsama hızı açısından önerilen algoritmanın daha iyi performans gösterdiği belirtilmiştir.

Özcan ve Toklu [182] istasyon sayısının en küçüklenmesini ve iş yükünün istasyonlar arasında mümkün olduğunca eşit paylaştırılmasını amaçlayan ÇTMHDP'nin çözümü için tabu arama algoritmasını kullanmışlardır. Önerilen algoritma küçük ve büyük boyutlu test problemleri üzerinde çalıştırılmış ve sonuçlar genetik algoritma [170], grup atama prosedürü [173], karınca koloni optimizasyonu algoritması [179] ve istasyon tabanlı atama prosedürü [177] ile karşılaştırılmıştır. Karşılaştırma sonucunda önerilen algoritmanın p205 problem kümesi dışında diğer test problemlerinin hepsinde karşılaştırılan sonuçlara göre daha iyi ya da aynı sonuçlara ulaşıldığı ancak diğer algoritmalara göre daha çok hesaplama süresi gerektirdiği belirtilmiştir.

Özcan ve Toklu [183] öncelikle Kim ve ark. [181] çalışmasında verilen karma tamsayılı programlama modelini istasyon sayısının en küçüklenmesi amacına uygun olarak düzenlemiş ve modele bölgesel kısıtları eklemişlerdir. Önerilen model ile küçük boyutlu test problemleri çözülerek optimum çözümler elde edilmiştir. Diğer taraftan ÇTMHDP'nin çözümü için ilk kez çok-kriterli karar-verme yaklaşımı kullanılmış; deterministik ve bulanık olmak üzere iki karma tamsayılı amaç programlama modeli önerilmiştir. Bahsedilen modellere ait amaçlar karşılıklı istasyon sayısı, çevrim süresi ve her istasyona atanan iş sayısı olarak belirlenmiştir. Amaç programlama modellerinin

deneysel sonuçları önerilen modellerin geçerli olduğunu ve karar vericinin çeşitli şartlar altında birçok senaryoyu değerlendirebileceğini göstermiştir.

Özcan ve Toklu [184] ÇTKMMHD probleminin çözümü için bir matematiksel model ve benzetimli tavlama yaklaşımı önermişlerdir. Önerilen matematiksel model istasyon sayısının en küçüklenmesini amaçlarken, benzetimli tavlama yaklaşımı ise ağırlıklandırılmış hat etkinliğinin en büyüklenmesi ve birbiriyle ilişkili işler arasındaki boş zamanın en küçüklenmesi ile ilgilenmektedir.

Tablo 4.1'de bilimsel yazında ÇTMHDP ile ilgili yapılmış çalışmalar problemin yapısı, kullanılan amaç, yöntem ve kısıtlara göre özetlenmiştir. Görüldüğü gibi çalışmaların çoğu belirli bir çevrim süresi dâhilinde istasyon sayısını minimize etmeyi amaçlamaktadır. Her ne kadar probleme ait matematiksel modeller önerilse de problemin NP-zor yapısı sebebiyle birçok metasezgisel yöntem de önerilmiştir. Diğer taraftan ÇTMHDP'ne özel kısıtlar çözüm yaklaşımına dâhil edildikçe problem daha karmaşıklaşmaktadır. Tablo 4.1'de belirtildiği gibi bir çalışmada sadece konumsal kısıtlara sahip ÇTMHDP, iki çalışmada sadece bölgesel kısıtlara sahip ÇTMHDP, iki çalışmada ise bölgesel ve senkronizasyon kısıtlarına sahip ÇTKMMHDP ele alınmıştır. Tablo 4.1'den görüldüğü gibi bilimsel yazında konumsal, bölgesel ve senkronizasyon kısıtlarının tamamının aynı anda probleme dahil edildiği bir çalışma yoktur.

Tablo 4.1. ÇTMHDP çalışmalarının sınıflandırılması.

Araştırmacılar	Problem	Amaç	Kullanılan Yöntem	Kullanılan Kısıtlar		
				Konumsal	Bölgesel	Senkronizasyon
Kim ve ark. [170]	ÇTMHDP	Tip-1	Genetik algoritma	√	-	-
Lee ve ark. [173]	ÇTMHDP	Tip-5	Grup atama prosedürü	-	-	-
Lapierre ve Ruiz [174]	ÇTMHDP	Tip-1	Öncelik tabanlı bir sezgisel yöntem	-	-	-
Simaria ve Vilarinho [175]	ÇTKMMHDP	Tip-1	Karınca koloni optimizasyonu algoritması	-	√	√
Simaria ve Vilarinho [176]	ÇTKMMHDP ve ÇTMHDP	Tip-1	Matematiksel model ve karınca koloni optimizasyonu algoritması	-	√	√
Hu ve ark. [177]	ÇTMHDP	Tip-1	İstasyon tabanlı atama prosedürü	-	-	-
Baykasoğlu ve Dereli [179]	ÇTMHDP	Tip-1	Karınca koloni optimizasyonu algoritması	-	√	-
Wu ve ark. [180]	ÇTMHDP	Tip-1	Matematiksel model ve dal-sınır algoritması	-	-	-
Kim ve ark. [181]	ÇTMHDP	Tip-2	Matematiksel model ve genetik algoritma	-	-	-
Özcan ve Toklu [182]	ÇTMHDP	Tip-1	Tabu arama	-	-	-
Özcan ve Toklu [183]	ÇTMHDP	Tip-1	Matematiksel model, deterministik ve bulanık amaç programlama	-	√	-
Özcan ve Toklu [184]	ÇTKMMHDP	Tip-1	Matematiksel model ve benzetimli tavlama	-	-	-

4.5. Çift Taraflı Montaj Hattı Dengeleme Problemi için Önerilen Matematiksel Model

Bu bölümde sunulan matematiksel model Simaria ve Vilarinho (2007) çalışmasında önerilen matematiksel modele dayanmaktadır. Simaria ve Vilarinho (2007) çalışmasında sunulan matematiksel model, bölgesel ve senkronizasyon kısıtlarına sahip karma model ÇTMHDP için geliştirilmiş iken, bu çalışmada ilgili matematiksel model tek modele indirgenmiş, amaç fonksiyonu farklılaştırılarak herhangi bir özel kısıta sahip olmayan ÇTMHDP, bölgesel kısıta sahip ÇTMHDP ve konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için güncellenmiştir.

4.5.1. Özel Kısıt İçermeyen Çift Taraflı Montaj Hattı Dengeleme Problemi için Önerilen Matematiksel Model ve Sonuçları

Önerilen karma tamsayılı doğrusal olmayan programlama modeline ait parametreler, tamsayılı ve ikili değişkenler, kısıtlar ve amaç fonksiyonu aşağıdaki gibi tanımlanmıştır.

Parametreler:

N = işler kümesi (i = 1,…,N)
K = istasyonlar kümesi (k = 1,…,N)
B = yönler kümesi (b = 1,2)
t_i = i işinin işlem süresi
ct = çevrim süresi
M = büyük bir sayı
S_L= sol yönde yapılacak işler kümesi
S_R= sağ yönde yapılacak işler kümesi
Suc_{ij} = i işinden sonra gelen j işleri kümesi

Tamsayılı değişkenler:

d_i = i işinin başlangıç zamanı
s_{kb} = k. istasyonun b. yönünde kullanılan toplam süre
$istsay$ = istasyon sayısı

İkili değişkenler:

$$x_{ikb} = \begin{cases} 1, & i.iş\ k.\ istasyonun\ b.yönüne\ atanırsa \\ 0, & diğer\ durumlarda \end{cases}$$

$$y_{kb} = \begin{cases} 1, & k.istasyonun\ b.yönü\ kullanılıyorsa \\ 0, & diğer\ durumlarda \end{cases}$$

$$u_{ij} = \begin{cases} 1, & i\ işi\ j\ işinden\ sonra\ atanmışsa \\ 0, & diğer\ durumlarda \end{cases}$$

Kısıtlar:

- $\sum_{K.B} x_{ikb} = 1 \qquad \forall i$

  Bir işin sadece bir istasyona atanmasını sağlar.

- $\sum_{K} x_{ik1} = 1 \qquad \forall i \in SL_i$

  Sol yönde yapılması gereken işleri sol yöne atamayı sağlar.

- $\sum_{K} x_{ik2} = 1 \qquad \forall i \in SR_i$

  Sağ yönde yapılması gereken işleri sağ yöne atamayı sağlar.

- $x_{ikb}(d_i + t_i) \leq ct \qquad \forall i, k, b$

  Bir istasyona atanmış işlerin toplam sürelerinin çevrim süresini aşmasını sağlar. Diğer bir deyişle işin başlangıç zamanı ve süresinin toplamı çevrim süresini geçmemelidir.

- $\sum_{K,B} x_{ikb}(d_i + (k-1)ct) + t_i - \sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) \leq 0 \qquad \forall i, j \in Suc_{ij}$

  Öncelik ilişkilerini sağlar (*i* işinden sonra *j* işinin geldiğini varsayalım; *j* işinin başlangıç zamanı, *i* işinin başlangıç zamanı ile süresinin toplamından küçük olmamalıdır).

- $\sum_{K,B} x_{ikb}(d_i + (k-1)ct) + t_i - \sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) \leq Mu_{ij} \qquad \forall i, j \notin Suc_{ij}, i \neq j$

- $\sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) + t_j - \sum_{K,B} x_{ikb}(d_i + (k-1)ct) \leq M(1 - u_{ij}) \qquad \forall i, j \notin Suc_{ij}, i \neq j$

  Arasında öncelik ilişkisi olmayan işler için çakışmayı önler. *i* ve *j* olmak üzere iki iş için ya *i* işi *j* işinden sonra ya da *j* işi *i* işinden sonra atanmıştır. Yukarıdaki kısıt kümesindeki ilk kısıt *j* işinin *i* işinden sonra, ikinci kısıt ise *i* işinin *j* işinden sonra atanmış olması durumları için geçerlidir.

- $s_{kb} = \sum_N x_{ikb} t_i \qquad \forall k, b$

  Her yöndeki toplam işlem süresini hesaplar.
- $\sum_N x_{ikb} \geq y_{kb} \qquad \forall k, b$
- $\sum_N x_{ikb} \leq M y_{kb} \qquad \forall k, b$

  Sadece açılan istasyonlara iş atanmasını sağlar.
- $istsay = \sum_{K,B} y_{kb}$

  Açılan istasyonlara göre istasyon sayısını hesaplar.

Amaç fonksiyonu:

- $\min 2\sqrt{\frac{\sum_{K,B}(y_{kb}(ct - s_{kb}))^2}{istsay}} + \frac{\sum_{K,B} y_{kb}(ct - s_{kb})}{istsay}$

  Bilimsel yazın araştırmasından görüldüğü gibi ÇTMHDP çalışmaları genellikle belirli bir çevrim süresi dâhilinde istasyon sayısının en küçüklenmesi (tip-1) amacına yöneliktir. Bu tez çalışmasında da önerilen algoritmaların performansını değerlendirmek açısından aynı amaç kullanılmıştır. Ancak aynı istasyon sayısına sahip iki çözümün biri diğerinden daha dengeli bir iş dağılımına sahip olabileceğinden amaç fonksiyonu iki parçaya ayrılmıştır. Amaç fonksiyonunun ilk parçası aynı istasyon sayısına sahip çözümler arasında en iyi dengeye sahip olanı bulmayı amaçlarken ikinci kısım çözümdeki istasyon sayısını minimize etmektedir. İlk amacın diğerine göre daha önemli olduğu düşünüldüğünden 2 ile çarpılmıştır [185].

$$\min 2\sqrt{\frac{\sum_{K,B}\left(y_{kb}(ct - s_{kb})\right)^2}{istsay}} + \frac{\sum_{K,B} y_{kb}(ct - s_{kb})}{istsay}$$

Kısıtlar

$$\sum_{K.B} x_{ikb} = 1 \qquad \forall i \quad (1)$$

$$\sum_{K} x_{ik1} = 1 \qquad \forall i \in SL_i \quad (2)$$

$$\sum_{K} x_{ik2} = 1 \qquad \forall i \in SR_i \quad (3)$$

$$x_{ikb}(d_i + t_i) \leq ct \qquad \forall i,k,b \quad (4)$$

$$\sum_{K,B} x_{ikb}(d_i + (k-1)ct) + t_i - \sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) \leq 0 \qquad \forall \{i,j\} \in Suc_{ij} \quad (5)$$

$$\sum_{K,B} x_{ikb}(d_i + (k-1)ct) + t_i - \sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) \leq Mu_{ij} \qquad \forall \{i,j\} \notin Suc_{ij} \ i \neq j \quad (6)$$

$$\sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) + t_j - \sum_{K,B} x_{ikb}(d_i + (k-1)ct) \leq M(1 - u_{ij}) \qquad \forall \{i,j\} \notin Suc_{ij} \ i \neq j \quad (7)$$

$$s_{kb} = \sum_{N} x_{ikb} t_i \qquad \forall k,b \quad (8)$$

$$\sum_{N} x_{ikb} \geq y_{kb} \qquad \forall k,b \quad (9)$$

$$\sum_{N} x_{ikb} \leq M y_{kb} \qquad \forall k,b \quad (10)$$

$$istsay = \sum_{K,B} y_{kb} \qquad (11)$$

$$d_i \geq 0 \qquad \forall i \quad (12)$$

$$s_{kb} \geq 0 \qquad \forall k,b \quad (13)$$

$$istsay \geq 0 \qquad tamsayı \quad (14)$$

$$x_{ikb} \in \{0,1\} \qquad \forall i,k,b \quad (15)$$

$$y_{kb} \in \{0,1\} \qquad \forall k,b \quad (16)$$

Önerilen matematiksel model GAMS 22.7.2 optimizasyon paket programı ile küçük boyutlu test problemlerinin bir kısmı için sonuçlar elde edebilmiş, ancak daha büyük boyutlu test problemleri için hesaplama süresi çok uzun sürdüğünden sonuç alınamamıştır. Herhangi bir özel kısıt içermeyen p9 ve p12 test problemlerine ait optimum sonuçlar hesaplama süreleri ile birlikte Tablo 4.2'de verilmiştir.

Tablo 4.2. Özel kısıt içermeyen ÇTMHDP için önerilen karma tamsayılı doğrusal olmayan matematiksel model sonuçları.

Problem	Çevrim süresi	İstasyon sayısı	CPU (sn.)
p9	3	6	28588.61
	4	5	10456.81
	5	4	2549.20
	6	3	799.31
p12	4	-	7200
	5	-	7200
	6	-	7200
	7	-	7200
	8	7	7200

* p12 problem kümesine ait istasyon sayısı değerleri, programın 7200 saniye çalıştırılması sonucu elde edilen uygun çözüme ait istasyon sayısını vermektedir. '-' 7200 saniye içerisinde uygun bir çözüm bulunamadığını göstermektedir.

4.5.2. Bölgesel Kısıta Sahip Çift Taraflı Montaj Hattı Dengeleme Problemi için Önerilen Matematiksel Model ve Sonuçları

Bölgesel kısıta sahip ÇTMHDP'nin çözümü için önerilen matematiksel modele iki parametre ve iki kısıt eklenmiştir.

Eklenen Parametreler:

ZP_{ij}=aynı istasyona atanması gereken işler kümesi

ZN_{ij}=aynı istasyona atanmaması gereken işler kümesi

Eklenen Kısıtlar:

- $\sum_K k(x_{ik1} + x_{ik2}) - \sum_K k(x_{jk1} + x_{jk2}) = 0 \qquad \forall i,j \in ZP_{ij}$

  Pozitif bölgesel kısıta sahip işlerin aynı istasyona atanmasını sağlar.

- $\sum_K k(x_{ik1} + x_{ik2}) - \sum_K k(x_{jk1} + x_{jk2}) \neq 0 \qquad \forall i,j \in ZN_{ij}$

  Negatif bölgesel kısıta sahip işlerin aynı istasyona atanmamasını sağlar.

$$\min 2\sqrt{\frac{\sum_{K,B}\left(y_{kb}(ct - s_{kb})\right)^2}{istsay}} + \frac{\sum_{K,B} y_{kb}(ct - s_{kb})}{istsay}$$

Kısıtlar

(1)-(17)

$$\sum_{K} k(x_{ik1} + x_{ik2}) - \sum_{K} k\left(x_{jk1} + x_{jk2}\right) = 0 \qquad \forall\{i,j\} \in ZP_{ij} \quad (18)$$

$$\sum_{K} k(x_{ik1} + x_{ik2}) - \sum_{K} k\left(x_{jk1} + x_{jk2}\right) \neq 0 \qquad \forall\{i,j\} \in ZN_{ij} \quad (19)$$

Önerilen matematiksel model GAMS 22.7.2 optimizasyon paket programı ile küçük boyutlu test problemlerinin bir kısmı için sonuçlar elde edebilmiş, ancak daha büyük boyutlu test problemleri için hesaplama süresi çok uzun sürdüğünden sonuç alınamamıştır. Bölgesel kısıta sahip p9 ve p12 test problemlerine ait optimum sonuçlar hesaplama süreleri ile birlikte aşağıdaki tabloda (Tablo 4.3) verilmiştir.

Tablo 4.3. Bölgesel kısıta sahip ÇTMHDP için önerilen karma tamsayılı doğrusal olmayan matematiksel model sonuçları.

Problem	Çevrim süresi	İstasyon sayısı	CPU (sn.)
p9	3	6	15225.09
	4	5	6399.62
	5	4	3806.94
	6	3	3109.53
p12	5	-	7200
	6	-	7200
	7	-	7200
	8	7	7200

* p12 problem kümesine ait istasyon sayısı değerleri, programın 7200 saniye çalıştırılması sonucu elde edilen uygun çözüme ait istasyon sayısını vermektedir. '-' 7200 saniye içerisinde uygun bir çözüm bulunamadığını göstermektedir.

4.5.3. Konumsal, Bölgesel ve Senkronizasyon Kısıtlarına Sahip Çift Taraflı Montaj Hattı Dengeleme Problemi için Önerilen Matematiksel Model ve Sonuçları

Konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP'nin çözümü için önerilen matematiksel modele aşağıdaki parametre ve kısıtlar eklenmiştir.

Eklenen Parametreler:

Pos_{ik}=konumsal kısıta sahip işler kümesi

ZP_{ij}=aynı istasyona atanması gereken işler kümesi

ZN_{ij}=aynı istasyona atanmaması gereken işler kümesi

SC_{ij}= hattın her iki yönünde aynı anda yapılması gereken işler kümesi

Eklenen Kısıtlar:

- $\sum_{K,B} x_{ikb}(d_i + (k-1)ct) + t_i - \sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) \leq Mu_{ij} \qquad \forall i,j \notin Suc_{ij}, SC_{ij}, i \neq j$
- $\sum_{K,B} x_{jkb}\left(d_j + (k-1)ct\right) + t_j - \sum_{K,B} x_{ikb}(d_i + (k-1)ct) \leq M(1 - u_{ij}) \qquad \forall i,j \notin Suc_{ij}, SC_{ij}, i \neq j$

  Arasında öncelik ilişkisi ve senkronizasyon kısıtı olmayan işler için çakışmayı önler.
- $\sum_B x_{ikb} = 1 \qquad \forall i,k \in Pos_{ik}$

  Konumsal kısıta sahip işlerin belirlenen istasyonlara atanmasını sağlar.
- $\sum_K k(x_{ik1} + x_{ik2}) - \sum_K k\left(x_{jk1} + x_{jk2}\right) = 0 \qquad \forall i,j \in ZP_{ij}$

  Pozitif bölgesel kısıta sahip işlerin aynı istasyona atanmasını sağlar.
- $\sum_K k(x_{ik1} + x_{ik2}) - \sum_K k\left(x_{jk1} + x_{jk2}\right) \neq 0 \qquad \forall i,j \in ZN_{ij}$

  Negatif bölgesel kısıta sahip işlerin aynı istasyona atanmamasını sağlar.
- $\sum_K x_{ik1}(d_i + (k-1)ct) - \sum_K x_{jk2}(d_j + (k-1)ct) = 0 \qquad \forall i,j \in SC_{ij}$

  Senkronizasyon kısıtına sahip işlerin aynı başlangıç zamanıyla aynı istasyonun karşılıklı yönlerine atanmasını sağlar.

$$\min 2\sqrt{\frac{\sum_{K,B}\left(y_{kb}(ct-s_{kb})\right)^2}{istsay}}+\frac{\sum_{K,B} y_{kb}(ct-s_{kb})}{istsay}$$

Kısıtlar

(1)-(5), (8)-(17)

$$\sum_{K} k(x_{ik1}+x_{ik2})-\sum_{K} k\left(x_{jk1}+x_{jk2}\right)=0 \qquad \forall\{i,j\}\in ZP_{ij} \quad (18)$$

$$\sum_{K} k(x_{ik1}+x_{ik2})-\sum_{K} k\left(x_{jk1}+x_{jk2}\right)\neq 0 \qquad \forall\{i,j\}\in ZN_{ij} \quad (19)$$

$$\sum_{B} x_{ikb}=1 \qquad \forall\{i,k\}\in Pos_{ik} \quad (20)$$

$$\sum_{K} x_{ik1}(d_i+(k-1)ct)-\sum_{K} x_{jk2}(d_j+(k-1)ct)=0 \qquad \forall\{i,j\}\in SC_{ij} \quad (21)$$

$$\sum_{K,B} x_{ikb}(d_i+(k-1)ct)+t_i-\sum_{K,B} x_{jkb}\left(d_j+(k-1)ct\right)\leq Mu_{ij} \qquad \forall\{i,j\}\notin Suc_{ij},SC_{ij} \quad i\neq j \quad (22)$$

$$\sum_{K,B} x_{jkb}\left(d_j+(k-1)ct\right)+t_j-\sum_{K,B} x_{ikb}(d_i+(k-1)ct)\leq M(1-u_{ij}) \qquad \forall\{i,j\}\notin Suc_{ij},SC_{ij} \quad i\neq j \quad (23)$$

Önerilen matematiksel model GAMS 22.7.2 optimizasyon paket programı ile küçük boyutlu test problemlerinin bir kısmı için sonuçlar elde edebilmiş, ancak daha büyük boyutlu test problemleri için hesaplama süresi çok uzun sürdüğünden sonuç alınamamıştır. Konumsal, bölgesel ve senkronizasyon kısıtlarına sahip p9 ve p12 test problemlerine ait optimum sonuçlar hesaplama süreleri ile birlikte aşağıdaki tabloda (Tablo 4.4) verilmiştir.

Tablo 4.4. Konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için önerilen karma tamsayılı doğrusal olmayan matematiksel model sonuçları.

Problem	Çevrim süresi	İstasyon sayısı	CPU (sn.)
p9	4	5	421.09
	5	4	200.09
	6	4	292.77
p12	5	6	53173.70
	6	5	17337.06
	7	5	46576.48
	8	4	11225.56

4.6. Çift Taraflı Montaj Hattı Dengeleme Problemi için Arı Algoritması

ÇTMHDP'nin çözümünde kullanılan AA'nın detaylı adımları Tablo 4.5'te verilmektedir.

Tablo 4.5. ÇTMHDP'nin çözümü için AA adımları.

1. Parametreleri başlangıç durumuna getir
2. Sezgisel kurallarla kâşif arı çözümlerini oluştur
3. Kâşif arı çözümlerinin uygunluk fonksiyonlarını değerlendir

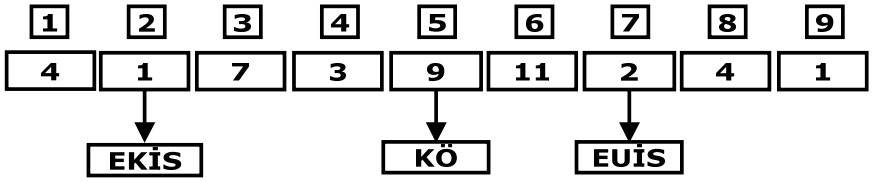
$$fit(\sigma^s) = 2\sqrt{\frac{\sum_{k=1}^{n}(ct-S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(ct-S_k)}{n}$$

4. I=0
5. *Do*
Artan şekilde sırala$_{s=1....S}$ *fit*(σ^s) ve en iyi *P* adet çözümü görevli arı olarak belirle
En iyi *e* adet görevli arıyı seç
En iyi *e* adet görevli arının her birine *nep* adet izci arı ata
Kalan *P-e* adet görevli arının her birine *nsp* adet izci arı ata
k=0
Do
Her izci arıya komşuluk yapılarından birini %50 olasılıkla uygula
Eğer $fit(\sigma^{kaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{kaydırma}$
Eğer $fit(\sigma^{değiştirme}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{değiştirme}$
En iyi çözümü güncelle
Eğer $min_{p=1,...,P} fit(\sigma^p) < fit(\sigma^{eniyi})$ ise $\sigma^{eniyi} = \sigma^p$
k=k+1
While (*k*<*P*)
Sezgisel kurallar ile *S-P* adet yeni kâşif arı çözümleri oluştur
I=*I*+1
While (*I*<*MaksIter*)

ÇTMHDP için AA, parametrelerin başlangıç değerlerine atanması ile başlar (*S, P, e, nep, nsp, MaksIter*) ve 4.6.1 bölümünde detayları verilecek olan sezgisel kurallarla *S* adet başlangıç kâşif arı çözümü oluşturulması ile devam eder. Oluşturulan çözümler kümesinden *P* adet iyi çözüm görevli arı çözümleri olarak; *P* adet iyi çözüm arasından seçilen *e* adet çözüm ise en iyi çözümler olarak belirlenir. Daha detaylı bir komşuluk araması için en iyi çözümlere *nep* adet izci arı gönderilir. Daha az sayıda izci arı ise kalan *P-e* adet çözüme gönderilir. Yerel arama için her görevli arıya kaydırma ve değiştirme komşuluk mekanizmalarından biri uygulanır. Daha iyi bir çözüm bulunmuşsa görevli arı çözümü güncellenir. Diğer taraftan en iyi görevli arı çözümü o ana kadar bulunan en iyi çözümle karşılaştırılır ve daha iyi bir çözüm bulunmuşsa en iyi çözüm güncellenir. Global arama için sezgisel kurallarla *S-P* adet kâşif arı çözümü oluşturularak bir sonraki iterasyona geçecek popülasyon sayısı tamamlanır. Bahsedilen adımlar *MaxIter* sayısınca tekrarlanır. ÇTMHDP'nin çözümünde kullanılan AA'nın önemli adımlarının detayları sonraki bölümlerde verilmiştir.

4.6.1. Arı Kolonisinin Oluşturulması

ÇTMHDP'nin çözümü için önerilen AA'nda çözüm dizilerinin gösterim şekli olarak sezgisel tabanlı gösterim kullanılmıştır. Bu gösterim şeklinde çözümler dolaylı bir yolla temsil edilmekte yani çözüm dizisinin her değeri bir sezgisel kuralı ifade etmektedir. Şekil 4.2'de 9 işe sahip bir veri kümesi için örnek bir çözüm dizisi gösterilmiştir. Verilen örnekte 2. sırada atanacak işin en kısa işlem süresine sahip iş kuralına göre, 5. sırada atanacak işin konumsal kısıta sahip bir işin önceki işine öncelik verme kuralına göre ve 7. sırada atanacak işin en uzun işlem süresine sahip iş kuralına göre seçileceği ifade edilmektedir. Yani çözüm dizisinin *i*. elemanı *i*. sırada atanacak işi atamak için kullanılacak sezgisel kuralı temsil etmektedir.



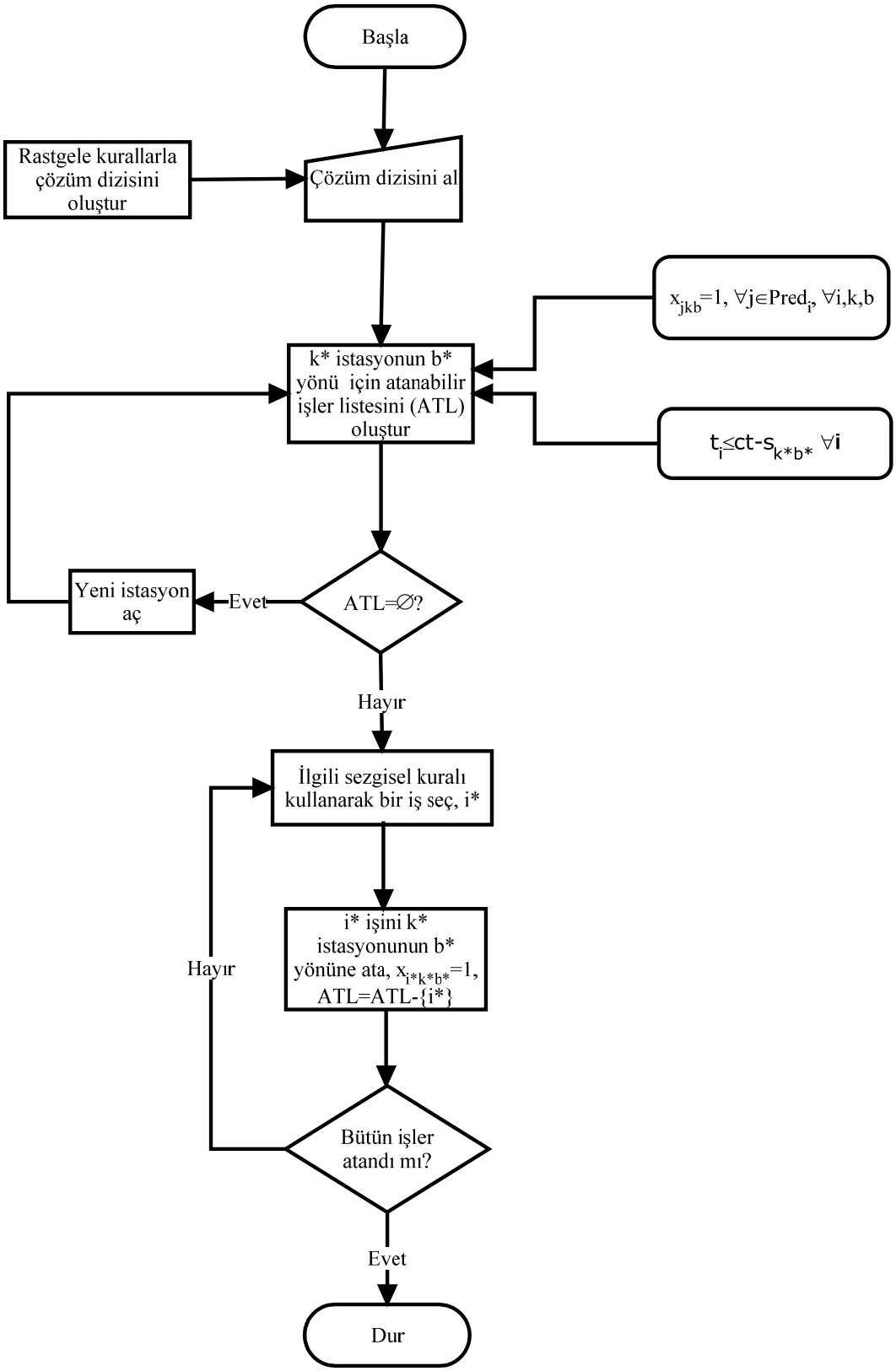
Şekil 4.2. Çözüm dizisi örneği.

Algoritmaya dâhil edilen sezgisel kurallar aşağıdaki gibidir:

1) En kısa işlem süresi (EKİS): Minimum işlem süresine sahip işi seçer.
2) En uzun işlem süresi (EUİS): Maksimum işlem süresine sahip işi seçer.
3) Kendinden sonraki iş sayısı toplamının minimumu (MiİSa): Kendinden sonra gelen iş sayısı en az olan işi seçer.
4) Kendinden sonraki iş sayısı toplamının maksimumu (MaİSa): Kendinden sonra gelen iş sayısı en çok olan işi seçer.
5) Kendinden sonraki işlerin toplam işlem süresinin minimumu (MiİSü): Kendinden sonra gelen işlerin işlem süresi toplamı en az olan işi seçer.
6) Kendinden sonraki işlerin toplam işlem süresinin maksimumu (MaİSü): Kendinden sonra gelen işlerin işlem süresi toplamı en çok olan işi seçer.
7) Maksimum sıralı konumsal ağırlık (MaSKA): Sıralı konumsal ağırlık değeri en büyük olan işi seçer. Bir işe ait sıralı konumsal ağırlık değeri ise o işin işlem süresi ile kendinden sonra gelen işlerin işlem sürelerinin toplamı ile elde edilir.

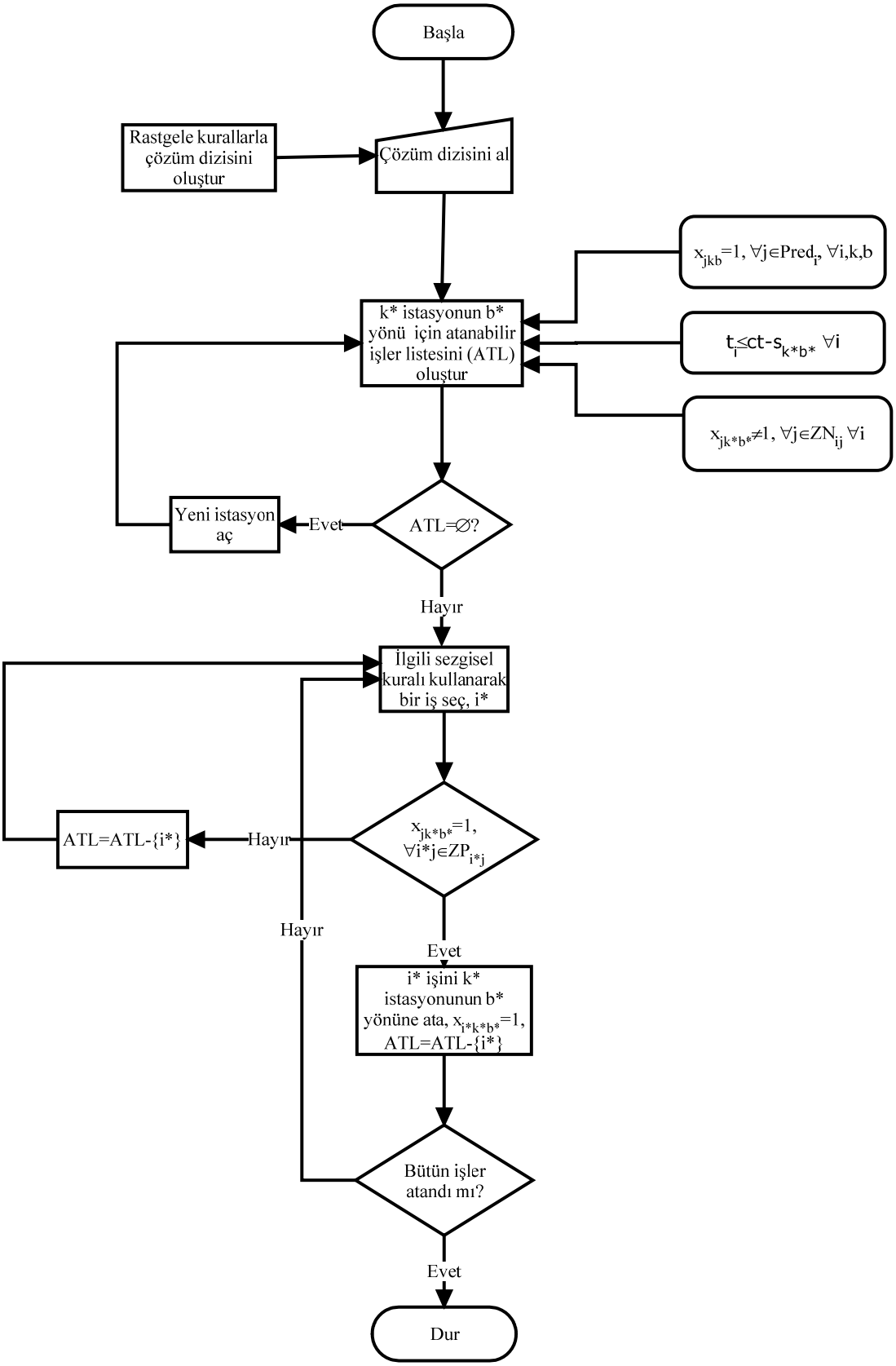
8) Maksimum ortalama sıralı konumsal ağırlık (MaOSKA): Ortalama sıralı konumsal ağırlık değeri en büyük olan işi seçer. Bir işe ait ortalama sıralı konumsal ağırlık değeri ise o işin işlem süresi ile kendinden sonra gelen işlerin işlem sürelerinin toplamının iş sayısına bölünmesi ile elde edilir.
9) Pozitif bölgesel kısıt önceliği (PBÖ): Pozitif bölgesel kısıta sahip bir işin önceki işini seçer. Eğer atanabilir işler listesinde bu özelliğe sahip bir iş yoksa rastgele seçim yapılır.
10) Konumsal kısıt önceliği (KÖ): Konumsal kısıta sahip bir işin önceki işini seçer. Eğer atanabilir işler listesinde bu özelliğe sahip bir iş yoksa rastgele seçim yapılır.
11) Senkronizasyon kısıtı önceliği (SÖ): Senkronizasyon kısıtına sahip bir işin önceki işini seçer. Eğer atanabilir işler listesinde bu özelliğe sahip bir iş yoksa rastgele seçim yapılır.

Herhangi bir özel kısıta sahip olmayan ÇTMHDP için çözüm dizilerinin oluşturulması esnasında (1)-(8) kuralları kullanılır ve çözüm dizisi uzunluğu iş sayısına eşittir. Bahsedilen problem için başlangıç çözümlerinin oluşturulması aşaması Şekil 4.3'te açıklanmaktadır. Öncelikle ilgili istasyonun kapasitesini aşmayan ve varsa önceki işleri bir istasyona atanmış işlerden ($Pred_i$, i işinden hemen önceki işler kümesini göstermek üzere) atanabilir işler listesi oluşturulur. Eğer atanabilir işler listesi boş ise yeni bir istasyon açılır, aksi takdirde verilen çözüm dizisinde sıradaki kural kullanılarak bir iş seçilir ve ataması gerçekleştirilir. Atanan iş, atanabilir işler listesinden çıkarılır. Bu adımlar bütün işler bir istasyona atanana kadar tekrarlanır.



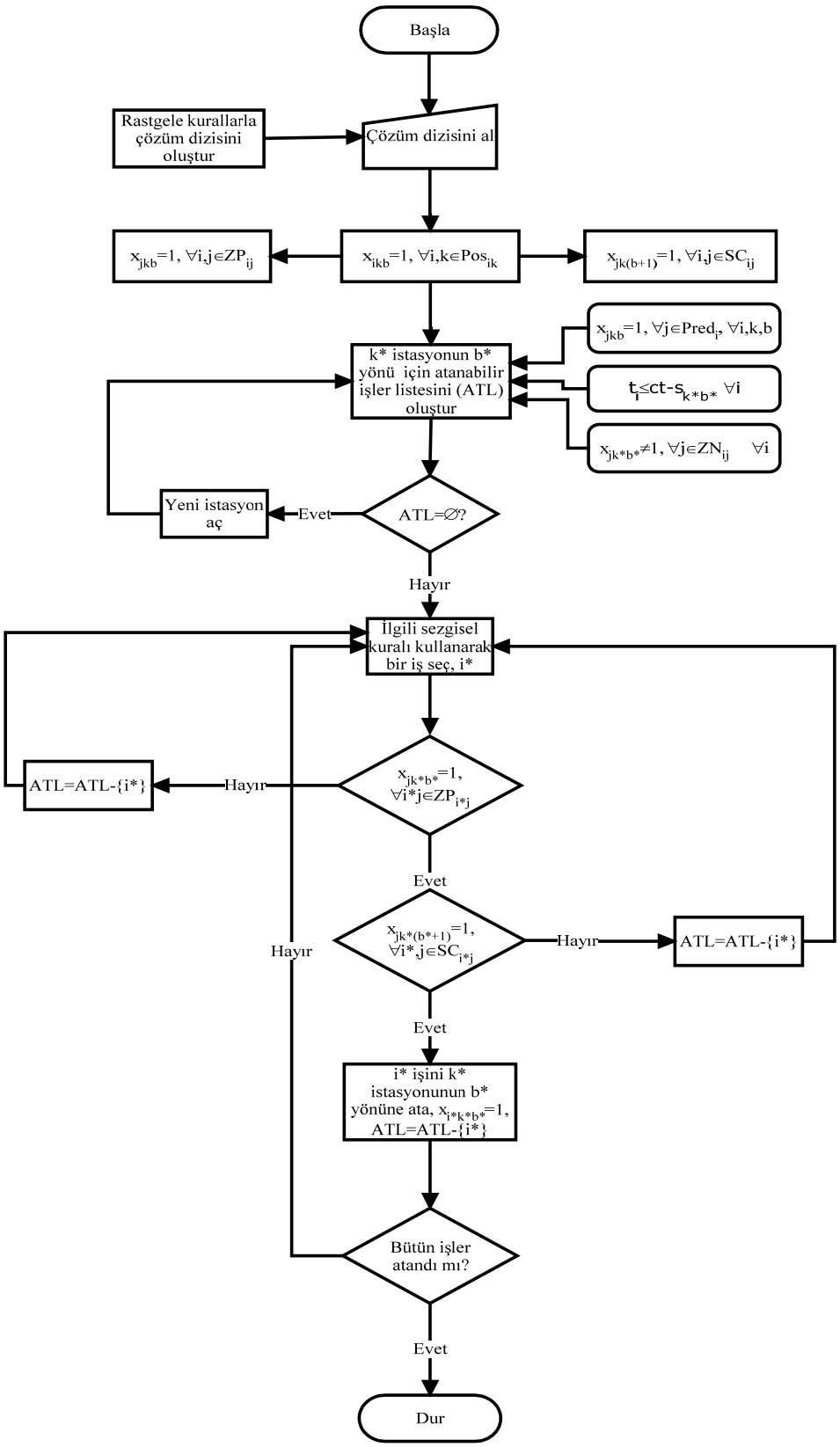
Şekil 4.3. Özel kısıta sahip olmayan ÇTMHDP için başlangıç çözümlerinin oluşturulması.

Bölgesel kısıta sahip ÇTMHDP için çözüm dizilerinin oluşturulması esnasında ise (1)-(9) kuralları kullanılır ve çözüm dizisi uzunluğu, iş sayısından pozitif bölgesel kısıt sayısının çıkarılması ile hesaplanır. Çünkü pozitif bölgesel kısıta sahip bir iş atandığında karşılık işin de aynı istasyona atanması gerekir ve bu atama esnasında atama kurallarından biri kullanılmaz. Bahsedilen problem için başlangıç çözümlerinin oluşturulması aşaması Şekil 4.4'te açıklanmaktadır. Öncelikle ilgili istasyonun kapasitesini aşmayan, varsa önceki işleri bir istasyona atanmış ve atama yapılacak istasyonda negatif bölgesel kısıt karşılığı olmayan işlerden atanabilir işler listesi oluşturulur. Eğer atanabilir işler listesi boş ise yeni bir istasyon açılır, aksi takdirde verilen çözüm dizisinde sıradaki kural kullanılarak bir iş seçilir. Seçilen işin ataması gerçekleştirilmeden önce bu işe ait bir pozitif bölgesel kısıt olup olmadığı kontrol edilir. Eğer varsa ve her iki iş de aynı istasyona atanabiliyorsa (öncelik, çevrim süresi ve ilgili istasyonda negatif bölgesel kısıt karşılığı olmama şartları sağlanıyorsa) atama gerçekleştirilir ve karşılık iş de aynı istasyona atanır, eğer atama gerçekleştirilemiyorsa seçilen iş atanabilir işler listesinden çıkarılır. Bu adımlar bütün işler atanana kadar tekrarlanır.



Şekil 4.4. Bölgesel kısıta sahip ÇTMHDP için başlangıç çözümlerinin oluşturulması.

Konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için çözüm dizilerinin oluşturulması esnasında (1)-(11) kuralları kullanılmakta ve çözüm dizisi uzunluğu, iş sayısından konumsal kısıt, pozitif bölgesel kısıt ve senkronizasyon kısıtı sayısı çıkarılarak hesaplanmaktadır. Bahsedilen problem için başlangıç çözümlerinin oluşturulması aşaması Şekil 4.5'te açıklanmaktadır. Diğer modellerden farklı olarak öncelik konumsal kısıta sahip işlere verilmektedir. Konumsal kısıta sahip iş, belirlenen istasyona atandıktan sonra bu işe ait bir pozitif bölgesel kısıt olup olmadığı kontrol edilerek eğer varsa karşılık iş de aynı istasyona atanır. Diğer taraftan yine konumsal kısıta sahip işe ait bir senkronizasyon kısıtı olup olmadığı kontrol edilir ve karşılık iş de aynı başlangıç süresiyle karşılık istasyona atanır. Daha sonra başlangıç istasyonuna dönülerek ilgili istasyonun kapasitesini aşmayan, varsa önceki işleri bir istasyona atanmış ve atama yapılacak istasyonda negatif bölgesel kısıt karşılığı olmayan işlerden atanabilir işler listesi oluşturulur. Eğer atanabilir işler listesi boş ise yeni bir istasyon açılır, aksi takdirde verilen çözüm dizisinde sıradaki kural kullanılarak bir iş seçilir. Seçilen işin ataması gerçekleştirilmeden önce bu işe ait bir pozitif bölgesel kısıt olup olmadığı kontrol edilir. Eğer varsa ve her iki iş de aynı istasyona atanabiliyorsa (öncelik, çevrim süresi ve ilgili istasyonda negatif bölgesel kısıt karşılığı olmama şartları sağlanıyorsa) atama gerçekleştirilir ve karşılık iş de aynı istasyona atanır, eğer atama gerçekleştirilemiyorsa seçilen iş atanabilir işler listesinden çıkarılır. Aynı şekilde seçilen işin ataması gerçekleştirilmeden önce bu işe ait bir senkronizasyon kısıtı olup olmadığı kontrol edilir ve eğer varsa karşılık iş de aynı başlangıç süresiyle karşılık istasyona atanır. Eğer atama gerçekleştirilemiyorsa (öncelik, çevrim süresi ve ilgili istasyonda negatif bölgesel kısıt karşılığı olmama şartları sağlanamıyorsa) seçilen iş atanabilir işler listesinden çıkarılır. Bu adımlar bütün işler atanana kadar tekrarlanır.



Şekil 4.5. Konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için başlangıç çözümlerinin oluşturulması.

4.6.2. Uygunluk Fonksiyonu ve Komşuluk Yapıları

Uygunluk fonksiyonu, ÇTMHDP için önerilen matematiksel modeldeki gibi ele alınmış olup n istasyon sayısını, s_k k. istasyondaki toplam işlem süresini göstermek üzere aşağıdaki gibi hesaplanmaktadır.

$$fit(\sigma^s) = 2\sqrt{\frac{\sum_{k=1}^{n}(ct - S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(ct - S_k)}{n} \qquad (4.1.)$$

Komşuluk yapısı olarak ise algoritmanın performansını öne çıkarabilmek için basit yapılar kullanılmıştır. Bu amaçla kaydırma ve değiştirme komşuluk yapıları %50 olasılıkla değişimli olarak kullanılmıştır. Diğer taraftan her bir komşuluk aramasında dizi uzunluğunun %10'u kadar kaydırma ya da değiştirme yapılmıştır.

4.7. Çift Taraflı Montaj Hattı Dengeleme Problemi için Yapay Arı Kolonisi Algoritması

ÇTMHDP'nin çözümünde kullanılan YAK algoritmasının detaylı adımları Tablo 4.6'da verilmektedir.

YAK algoritması, parametrelerin başlangıç değerlerine atanması ile başlar ve sezgisel kurallarla P adet başlangıç görevli arı çözümü oluşturulması ile devam eder. Görevli arı çözümlerine, çözümün kalitesi ile orantılı sayıda izci arı atanır. Yerel arama için her görevli arıya kaydırma ve çift kaydırma komşuluk mekanizmalarından biri uygulanır. Daha iyi bir çözüm bulunmuşsa görevli arı çözümü güncellenir. Diğer taraftan görevli arı çözümleri o ana kadar bulunan en iyi çözümle karşılaştırılır ve daha iyi bir çözüm bulunmuşsa en iyi çözüm güncellenir. Görevli arı çözümü *limit* parametresi ile belirlenen sayıda iterasyon boyunca geliştirilememişse sezgisel kurallarla yeni bir kâşif arı çözümü oluşturulur. Son olarak algoritma adımları önceden belirlenmiş iterasyon sayısı kadar tekrarlanır.

YAK algoritmasında başlangıç çözümlerinin oluşturulması, kullanılan komşuluk yapıları ve uygunluk fonksiyonu AA'ndakiler ile aynıdır.

Tablo 4.6. ÇTMHDP için YAK algoritması adımları.

1. Parametreleri başlangıç değerlerine ata
2. Sezgisel kurallarla başlangıç görevli arı çözümlerini oluştur (σ^p)
3. Görevli arı çözümlerinin uygunluk fonksiyonlarını değerlendir

$$fit(\sigma^p) = 2\sqrt{\frac{\sum_{k=1}^{n}(ct - S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(ct - S_k)}{n}$$

4. I=0
5. *Do*
Uygunluk değerleriyle ilişkili olasılıkları hesapla

$$p_p = \frac{\left(\sum fit(\sigma^p)^{-1}\right)^{-1}}{fit(\sigma^p)}$$

Olasılıklara göre izci arıları görevli arılara ata
Hesaplanan olasılıklara göre, görevli arıların yiyecek kaynaklarına gönderilecek izci arı sayısını belirle, p_p*P
k=0
Do
Her izci arıya komşuluk yapılarından birini %50 olasılıkla uygula
Eğer $fit(\sigma^{kaydırma}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{kaydırma}$, $LimitSayacı(\sigma^p)$=0
Aksi takdirde $LimitSayacı(\sigma^p)=LimitSayacı(\sigma^p)$+1
Eğer $fit(\sigma^{değiştirme}) < fit(\sigma^p)$ ise $\sigma^p = \sigma^{değiştirme}$,
$LimitSayacı(\sigma^p)$=0
Aksi takdirde $LimitSayacı(\sigma^p)=LimitSayacı(\sigma^p)$+1
En iyi çözümü güncelle
Eğer $min_{p=1,\ldots,P} fit(\sigma^p) < fit(\sigma^{eniyi})$ ise $\sigma^{eniyi}=\sigma^p$
Eğer ($LimitSayacı(\sigma^p)>limit$) ise sezgisel kurallarla yeni bir kâşif arı çözümü oluştur
$k=k$+1
While ($k<P$)
$I=I$+1
While ($I=MaksIter$)

4.8. Deneysel Çalışma

Bilimsel yazındaki çalışmaların sonuçlarıyla birebir karşılaştırma yapabilmek için AA ve YAK algoritmaları özel kısıt içermeyen, bölgesel kısıta sahip ve konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için ayrı ayrı çalıştırılıp sonuçlar ilgili çalışmalarla karşılaştırılmıştır.

Test problemleri küçük boyutlu (p9, p12, p16, p24) ve büyük boyutlu (p65, p148, p205) problemleri içermekte olup problem adlarındaki sayılar ilgili verideki iş sayısını göstermektedir. p9, p12 ve p24 problemleri Kim ve ark. (2000) çalışmasından, p16, p65, p205 problemleri Lee ve ark. (2001) çalışmasından ve son olarak p148 problemi

ise Bartholdi (1993) çalışmasından alınmıştır. p148 problemindeki 79 ve 108 nolu işlere ait işlem süreleri diğer işlem sürelerine göre çok büyük olduğundan ve çevrim süresi üzerinde bir kısıt oluşturduğundan karşılaştırma yapılan diğer çalışmalarda olduğu gibi bu işlere ait süreler sırasıyla 2.81 yerine 1.11 ve 3.83 yerine 0.43 olarak alınmıştır. p9, p12, p24, p65, p148 problemlerine ait konumsal kısıt verileri Kim ve ark. (2000) çalışmasından alınmışken, p16 ve p205 problemlerine ait konumsal kısıt verileri ise bu çalışmada belirlenmiştir. p9, p12, p24, p65, p148, p205 problemlerine ait bölgesel kısıt verileri Baykasoğlu ve Dereli (2008) çalışmasından, p16 problemine ait bölgesel kısıt verileri ise Özcan ve Toklu (2009b) çalışmasından alınmıştır. Bütün test problemlerine ait senkronizasyon kısıtları ise verinin özelliklerine bağlı olarak bu çalışmada belirlenmiştir.

4.8.1. Arı Algoritması Sonuçları

ÇTMHDP'nin çözümü için önerilen AA, C# programlama dilinde kodlanarak 2.2 GHz CPU ve 2GB RAM özelliklere sahip Intel Core 2 Duo PC kullanılarak test problemleri üzerinde analiz edilmiştir. Kullanılan kontrol parametresi değerlerinin belirlenmesi aşamasında {*S,P,e,nep,nsp*}: {20,10,5,4,2}, {15,5,3,4,2}, {10,5,3,2,1}, {5,3,2,2,1} olmak üzere 4 farklı parametre kombinasyonu oluşturulmuştur. Yapılan analizler neticesinde 3. parametre kombinasyonu en iyi performansı gösterdiğinden Tablo 4.7'de verildiği gibi algoritmanın parametreleri olarak bu değerler kullanılmıştır.

Tablo 4.7. ÇTMHDP için AA parametrelerinin değerleri.

Parametre	Değer
S	10
P	5
e	3
nep	2
nsp	1
MaksIter	100

AA her problem kümesinin bütün çevrim süreleri için 10'ar kez çalıştırılmıştır. Minimum, ortalama ve maksimum istasyon sayıları, minimum CPU süreleri ile birlikte ilgili tablolarda sunulmuştur. CPU süresi olarak minimum değerlerin dikkate alınmasının sebebi karşılaştırma yapılan çalışmalarda da minimum CPU sürelerinin verilmesidir. Daha önce bahsedildiği gibi deneysel çalışma 3 kategoride ele alınmıştır. Bunlardan ilki olan özel kısıt içermeyen ÇTMHDP sonuçları Tablo 4.8, bölgesel kısıta

sahip ÇTMHDP sonuçları Tablo 4.9, konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP sonuçları ise Tablo 4.10'da verilmiştir.

Tablo 4.8. AA ile özel kısıt içermeyen ÇTMHDP sonuçları.

	Çevrim süresi	min	ort	maks	CPU (sn.)
p9	3	6	6	6	<0.02
	4	5	5	5	<0.02
	5	4	4	4	<0.02
	6	3	3	3	<0.02
p12	4	7	7	7	<0.02
	5	6	6	6	<0.02
	6	5	5	5	<0.02
	7	4	4	4	<0.02
	8	4	4	4	<0.02
p16	15	6	6	6	<0.02
	16	6	6	6	<0.02
	18	6	6	6	<0.02
	19	5	5	5	<0.02
	20	5	5	5	<0.02
	21	5	5	5	<0.02
	22	4	4	4	<0.02
p24	18	8	8	8	<0.02
	20	8	8	8	<0.02
	24	6	6.4	7	0.031
	25	6	6	6	<0.02
	30	5	5	5	<0.02
	35	4	4	4	<0.02
	40	4	4	4	<0.02
p65	326	17	17	17	0.250
	381	14	14.5	15	1.078
	435	12	12.9	13	5.125
	490	11	11.3	12	2.203
	544	10	10	10	0.312
p148	204	26	26	26	0.406
	255	21	21	21	0.375
	306	17	17.6	18	0.390
	357	15	15	15	0.359
	408	13	13.3	14	0.453
	459	12	12	12	0.171
	510	11	11	11	0.406
p205	1133	22	23.7	24	33.390
	1322	20	20	20	32.953
	1510	17	17.9	18	2906.859
	1699	16	16	16	5.578
	1888	14	14.3	15	12.390
	2077	12	12.8	13	149.703
	2266	12	12	12	7.359
	2454	11	11.9	12	921.682
	2643	10	10	10	21.437
	2832	10	10	10	8.640

* CPU süresi olarak bulunan <0.02 değeri, karşılık gelen istasyon sayısının bir iterasyonda bulunduğunu göstermektedir.

Tablo 4.9. AA ile bölgesel kısıta sahip ÇTMHDP sonuçları.

	Çevrim süresi	min	ort	maks	CPU (sn.)		Çevrim süresi	min	ort	maks	CPU (sn.)
p9	3	7	7	7	<0.02	p65	326	17	17.3	18	0.250
	4	5	5	5	<0.02		381	14	14.9	15	9.328
	5	4	4	4	<0.02		435	13	13	13	0.265
	6	3	3	3	<0.02		490	11	11.8	12	6.015
p12	5	6	6	6	<0.02		544	10	10	10	0.687
	6	5	5	5	<0.02	p148	204	26	26.7	27	6.421
	7	4	4	4	<0.02		255	21	21.2	22	1.218
	8	4	4	4	<0.02		306	18	18	18	2.000
p16	15	6	6.4	7	3.657		357	15	15.6	16	14.531
	16	6	6.3	7	2.625		408	14	14	14	0.734
	18	6	6	6	<0.02		459	12	12.4	13	2.625
	19	6	6	6	<0.02		510	11	11.4	12	4.359
	20	5	5	5	<0.02	p205	1133	23	23.9	24	41.593
	21	5	5	5	0.093		1322	21	21.3	22	1.843
	22	5	5.2	6	0.512		1510	18	18	18	11.609
p24	18	8	8	8	<0.02		1699	17	17.8	18	13.25
	20	8	8	8	<0.02		1888	16	16	16	5.015
	24	6	6.2	7	0.109		2077	14	14.9	15	260.031
	25	6	6	6	<0.02		2266	14	14	14	12.078
	30	5	5	5	<0.02		2454	14	14	14	5.843
	35	4	4	4	<0.02		2643	13	13.8	14	94.578
	40	4	4	4	<0.02		2832	12	12	12	9.692

Tablo 4.10. AA ile konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP sonuçları.

	Çevrim süresi	min	ort	maks	CPU (sn.)
p9	4	5	5	5	<0.02
	5	4	4	4	<0.02
	6	4	4	4	<0.02
p12	5	6	6	6	<0.02
	6	5	5	5	<0.02
	7	5	5	5	<0.02
	8	4	4	4	<0.02
p16	15	7	7	7	<0.02
	16	7	7	7	<0.02
	18	7	7	7	<0.02
	19	7	7	7	<0.02
	20	7	7	7	<0.02
	21	7	7	7	<0.02
	22	6	6	6	<0.02
p24	18	8	8	8	<0.02
	20	8	8	8	<0.02
	24	7	7.2	8	0.062
	25	7	7	7	<0.02
	30	6	6	6	<0.02
	35	6	6	6	<0.02
	40	6	6	6	<0.02

	Çevrim süresi	min	ort	maks	CPU (sn.)
p65	326	18	18	18	0.234
	381	15	15.1	16	0.625
	435	14	14	14	0.390
	490	13	13	13	0.281
	544	12	12	12	0.578
p148	204	26	26.9	27	19.093
	255	23	23	23	2.421
	306	20	20.7	21	4.359
	357	18	18.1	19	2.015
	408	17	18	19	13.593
	459	17	17.6	18	2.078
	510	17	17	17	2.468
p205	1133	25	25.7	26	51.843
	1322	23	23	23	10.875
	1510	21	21	21	16.984
	1699	19	19.9	20	926.781
	1888	19	19.5	20	287.109
	2077	19	19	19	34.218
	2266	18	18.2	19	41.734
	2454	18	18	18	48.312
	2643	18	18	18	55.828
	2832	18	18	18	66.093

4.8.2. Yapay Arı Kolonisi Algoritması Sonuçları

ÇTMHDP'nin çözümü için önerilen YAK algoritması, C# programlama dilinde kodlanmış ve aynı özelliklere sahip PC kullanılarak test problemleri üzerinde analiz edilmiştir. Parametre değerlerinin belirlenmesinde AA parametreleri temel alınarak algoritma içerisinde aynı sayıda arı olmasına dikkat edilmiştir. YAK algoritması parametrelerinin değerleri Tablo 4.11'deki gibi belirlenmiştir.

Tablo 4.11. ÇTMHDP için YAK algoritması parametrelerinin değerleri.

Parametre	Değer
P	10
limit	25
MaksIter	100

YAK algoritması da yine AA'nda olduğu gibi her problem kümesinin bütün çevrim süreleri için 10'ar kez çalıştırılmıştır. Minimum, ortalama ve maksimum istasyon sayıları, minimum CPU süreleri ile birlikte ilgili tablolarda sunulmuştur. Bunlardan ilki olan özel kısıt içermeyen ÇTMHDP sonuçları Tablo 4.12, bölgesel kısıta sahip

ÇTMHDP sonuçları Tablo 4.13, konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP sonuçları ise Tablo 4.14'te verilmiştir.

Tablo 4.12. YAK algoritması ile özel kısıt içermeyen ÇTMHDP sonuçları.

	Çevrim süresi	min	ort	maks	CPU (sn.)
p9	3	6	6	6	<0.02
	4	5	5	5	<0.02
	5	4	4	4	<0.02
	6	3	3	3	<0.02
p12	4	7	7	7	<0.02
	5	6	6	6	<0.02
	6	5	5	5	<0.02
	7	4	4	4	<0.02
	8	4	4	4	<0.02
p16	15	6	6	6	<0.02
	16	6	6	6	<0.02
	18	6	6	6	<0.02
	19	5	5	5	<0.02
	20	5	5	5	<0.02
	21	5	5	5	<0.02
	22	4	4	4	<0.02
p24	18	8	8	8	<0.02
	20	8	8	8	<0.02
	24	6	6.3	7	0.031
	25	6	6	6	<0.02
	30	5	5	5	<0.02
	35	4	4	4	<0.02
	40	4	4	4	<0.02
p65	326	17	17	17	0.203
	381	14	14.5	15	0.343
	435	13	13	13	0.281
	490	11	11.4	12	1.015
	544	10	10	10	0.250
p148	204	26	26	26	0.343
	255	21	21	21	0.484
	306	17	17.9	18	14.328
	357	15	15	15	0.156
	408	13	13.6	14	1.359
	459	12	12	12	5.968
	510	11	11	11	0.343
p205	1133	22	23.3	24	71.578
	1322	20	20	20	23.312
	1510	18	18	18	4.375
	1699	16	16	16	15.625
	1888	14	14.6	15	40.031
	2077	14	14	14	5.125
	2266	12	12	12	12.328
	2454	12	12	12	60.687
	2643	10	10.1	11	17.109
	2832	10	10	10	7.390

Tablo 4.13. YAK algoritması ile bölgesel kısıta sahip ÇTMHDP sonuçları.

	Çevrim süresi	min	Ort	maks	CPU (sn.)		Çevrim süresi	min	Ort	maks	CPU (sn.)
p9	3	7	7	7	<0.02	p65	326	18	18	18	0.218
	4	5	5	5	<0.02		381	14	14.9	15	3.687
	5	4	4	4	<0.02		435	13	13	13	0.390
	6	3	3	3	<0.02		490	12	12	12	0.265
p12	5	6	6	6	<0.02		544	10	10	10	0.296
	6	5	5	5	<0.02	p148	204	26	26.7	27	2.203
	7	4	4	4	<0.02		255	21	21.1	22	2.125
	8	4	4	4	<0.02		306	18	18	18	0.546
p16	15	7	7	7	<0.02		357	15	15.7	16	0.578
	16	7	7	7	<0.02		408	14	14	14	0.703
	18	6	6	6	<0.02		459	12	12.3	13	0.265
	19	6	6	6	<0.02		510	11	11.8	12	2.406
	20	5	5	5	<0.02	p205	1133	24	24.2	25	8.031
	21	5	5.2	5	0.078		1322	21	21.5	22	24.187
	22	5	5	5	<0.02		1510	18	19.1	20	78.390
p24	18	8	8	8	<0.02		1699	18	18	18	4.390
	20	8	8	8	<0.02		1888	16	16	16	4.703
	24	6	6.2	7	<0.02		2077	14	14.8	15	83.625
	25	6	6	6	<0.02		2266	14	14	14	4.328
	30	5	5	5	<0.02		2454	14	14	14	4.750
	35	5	5	5	<0.02		2643	14	14	14	5.671
	40	4	4	4	<0.02		2832	13	13	13	12.156

Tablo 4.14. YAK algoritması ile konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP sonuçları.

	Çevrim süresi	min	ort	maks	CPU (sn.)		Çevrim süresi	min	ort	maks	CPU (sn.)
p9	4	5	5	5	<0.02	p65	326	18	18	18	0.203
	5	4	4	4	<0.02		381	15	15.1	16	1.593
	6	4	4	4	<0.02		435	14	14	14	0.437
p12	5	6	6	6	<0.02		490	13	13	13	0.375
	6	5	5	5	<0.02		544	12	12	12	0.296
	7	5	5	5	<0.02	p148	204	26	26.4	27	1.328
	8	4	4	4	<0.02		255	23	23	23	0.578
p16	15	7	7	7	<0.02		306	20	20.9	21	1.437
	16	7	7	7	<0.02		357	18	18.2	19	3.046
	18	7	7	7	<0.02		408	17	17.9	19	2.000
	19	7	7	7	<0.02		459	17	17.8	18	20.375
	20	7	7	7	<0.02		510	17	17	17	1.359
	21	7	7	7	<0.02	p205	1133	24	25.2	26	115.640
	22	6	6	6	<0.02		1322	23	23.1	24	24.937
p24	18	8	8	8	<0.02		1510	20	20.9	21	242.500
	20	8	8	8	<0.02		1699	20	20	20	55.296
	24	7	7	7	<0.02		1888	19	19.8	20	22.328
	25	7	7	7	<0.02		2077	18	18.9	19	960.812
	30	6	6.4	6	<0.02		2266	17	17.3	18	352.734
	35	6	6	6	<0.02		2454	17	17	17	56.937
	40	6	6	6	<0.02		2643	17	17	17	42.046
							2832	17	17	17	55.640

4.8.3. Arı Algoritması ve Yapay Arı Kolonisi Algoritmasının Bilimsel Yazındaki Algoritmalarla Karşılaştırılması

AA ve YAK algoritmasının ÇTMHDP'nin çözümü için kullanıldığı bir uygulama niteliğinde olan bu bölümde, geniş bir deneysel çalışma neticesinde AA ve YAK algoritması sonuçları bilimsel yazında sunulan sonuçlarla karşılaştırılmıştır. Değerlendirmeye bilimsel yazındaki sezgisel ve matematiksel model sonuçları dâhil edilmiş ancak problemin karmaşıklığı sebebiyle karşılaştırma yapılan sonuçların çoğu sezgisel arama algoritmalarından elde edilmiştir. Karşılaştırma bölümü, deneysel çalışma bölümünde olduğu gibi 3 kategoride ele alınmıştır. Tablo 4.15 özel kısıta sahip olmayan ÇTMHDP, Tablo 4.16 bölgesel kısıta sahip ÇTMHDP, Tablo 4.17 konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için elde edilen sonuçların bilimsel yazındaki sonuçlarla karşılaştırılmasını içermektedir. Tablolardaki koyu değerler bilinen en iyi istasyon sayısını göstermektedir.

Tablo 4.15. AA ve YAK algoritması ile elde edilen özel kısıta sahip olmayan ÇTMHDP sonuçlarının karşılaştırılması.

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Hu ve ark. (2008)		Özcan ve Toklu (2009a)	Özcan ve Toklu (2009b)		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı	CPU	İst. sayısı	CPU
p9	3	**6**	<0.02	**6**	<0.02	**6**	<1	-	-	**6**	**6**	0.093	**6**	28588.61
	4	**5**	<0.02	**5**	<0.02	**5**	<1	**5**	0.047	**5**	**5**	0.296	**5**	10456.81
	5	**4**	<0.02	**4**	<0.02	**4**	<1	**4**	0.047	**4**	**4**	0.140	**4**	2549.20
	6	**3**	<0.02	**3**	<0.02	**3**	<1	-	-	**3**	**3**	0.109	**3**	799.31

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Hu ve ark. (2008)		Wu ve ark. (2008)		Özcan ve Toklu (2009a)	Özcan ve Toklu (2009b)		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı	CPU	İst. sayısı	CPU
p12	4	**7**	<0.02	**7**	<0.02	-	-	-		**7**	0.010	-	-		-	-
	5	**6**	<0.02	**6**	<0.02	**6**	<1	-		**6**	0.210	**6**	**6**	22.609	-	-
	6	**5**	<0.02	**5**	<0.02	**5**	<1	**5**	0.078	**5**	0.010	**5**	**5**	12.042	-	-
	7	**4**	<0.02	**4**	<0.02	**4**	<1	**4**	0.141	**4**	0.001	**4**	**4**	0.203	-	-
	8	**4**	<0.02	**4**	<0.02	-	-	**4**	0.250	-	-	**4**	**4**	1.125	7	7200

	Çevrim süresi	AA		YAK		Hu ve ark. (2008)		Wu ve ark. (2008)		Özcan ve Toklu (2009a)	Özcan ve Toklu (2009b)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı	CPU
p16	15	**6**	<0.02	**6**	<0.02	-	-	7	0.121	-	**6**	132.859
	16	**6**	<0.02	**6**	<0.02	**6**	0.204	-	-	**6**	**6**	2.031
	18	**6**	<0.02	**6**	<0.02	-	-	**6**	0.100	-	**6**	153.328
	19	**5**	<0.02	**5**	<0.02	6	0.219	-	-	**5**	**5**	18.125
	20	**5**	<0.02	**5**	<0.02	-	-	**5**	4.756	-	**5**	156.609
	21	**5**	<0.02	**5**	<0.02	**5**	0.094	-	-	**5**	**5**	399.640
	22	**4**	<0.02	**4**	<0.02	**4**	0.156	**4**	0.161	**4**	**4**	0.671

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Hu ve ark. (2008)		Wu ve ark. (2008)		Özcan ve Toklu (2009a)	Özcan ve Toklu (2009b)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı	CPU
p24	18	**8**	<0.02	**8**	<0.02	-	-	**8**	0.828	-	-	**8**	**8**	<7200
	20	**8**	<0.02	**8**	<0.02	**8**	<1	**8**	1.218	-	-	**8**	**8**	<7200
	24	**6**	0.031	**6**	0.031	**-**	-	7	5.938	-	-	**6**	**6**	1621.437
	25	**6**	<0.02	**6**	<0.02	**6**	<1	**6**	8.627	**6**	0.130	**6**	**6**	<7200
	30	**5**	<0.02	**5**	<0.02	**5**	<1	-	-	**5**	0.010	**5**	**5**	<7200
	35	**4**	<0.02	**4**	<0.02	5	<1	-	-	**4**	21.010	**4**	**4**	259.671
	40	**4**	<0.02	**4**	<0.02	**4**	<1	-	-	**4**	0.010	**4**	**4**	<7200

	Çevrim süresi	AA		YAK		Lee ve ark. (2001)		Baykasoğlu ve Dereli (2008)		Wu ve ark. (2008)		Simaria ve Vilarinho (2009)	Özcan ve Toklu (2009a)
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı
p65	326	**17**	0.250	**17**	0.203	**17**	<3	**17**	<1	-	-	**17**	**17**
	381	**14**	1.078	**14**	0.343	15	<3	15	<1	**14**	0.047	**14**	15
	435	**12**	5.125	13	0.281	13	<3	13	<1	-	-	13	13
	490	**11**	2.203	**11**	1.015	12	<3	12	<1	**11**	2.187	12	**11**
	544	**10**	0.312	**10**	0.250	**10**	<3	**10**	2.48	**10**	5.230	**10**	**10**

	Çevrim süresi	AA		YAK		Lee ve ark. (2001)		Baykasoğlu ve Dereli (2008)		Wu ve ark. (2008)		Simaria ve Vilarinho (2009)	Özcan ve Toklu (2009a)
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı
p148	204	**26**	0.406	**26**	0.343	27	<3	**26**	4.39	**26**	3.235	**26**	**26**
	255	**21**	0.375	**21**	0.484	21	<3	**21**	15.64	**21**	11.063	**21**	**21**
	306	**17**	0.390	**17**	14.328	18	<3	18	50.91	-	-	18	18
	357	**15**	0.359	**15**	0.156	**15**	<3	**15**	3.78	**15**	68.152	**15**	**15**
	408	**13**	0.453	**13**	1.359	14	<3	14	2.19	**13**	35.014	14	**13**
	459	**12**	0.171	**12**	5.968	13	<3	**12**	180.76	**12**	9.703	**12**	**12**
	510	**11**	0.406	**11**	0.343	**11**	<3	**11**	15.05	**11**	10.657	**11**	**11**

	Çevrim süresi	AA		YAK		Lee ve ark. (2001)		Baykasoğlu ve Dereli (2008)		Simaria ve Vilarinho (2009)	Özcan ve Toklu (2009a)
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	İst. sayısı
p205	1133	**22**	33.390	**22**	71.578	23	<3	24	451.14	**22**	24
	1322	**20**	32.953	**20**	23.312	**20**	<3	22	449.27	**20**	21
	1510	**17**	2906.859	18	4.375	20	<3	18	288.20	**17**	18
	1699	16	5.578	16	15.625	16	<3	18	448.28	**15**	17
	1888	14	12.390	14	40.031	16	<3	15	177.84	**13**	16
	2077	**12**	149.703	14	5.125	14	<3	14	7.06	**12**	14
	2266	**12**	7.359	**12**	12.328	13	<3	**12**	131.30	**12**	13
	2454	11	921.682	12	60.687	12	<3	12	6.99	**10**	12
	2643	**10**	21.437	**10**	17.109	12	<3	11	68.54	**10**	11
	2832	**10**	8.640	**10**	7.390	**10**	<3	**10**	303.63	**10**	**10**

Tablo 4.15'ten görüldüğü gibi p9, p16 ve p24 problem kümelerinin bütün çevrim süreleri için AA ve YAK algoritmaları bilinen en iyi çözüme daha düşük CPU süresi ile ulaşmıştır. p12 problem kümesi için ise her iki algoritma ile de bilinen en iyi çözüm tüm çevrim süreleri için oldukça kısa CPU süreleri ile bulunmuştur. p65 problem kümesi için AA 5 çevrim süresinden 4'ünde bilinen en iyi çözüme; 1'inde ise (*ct*= 435) bilinen en iyi çözümden daha iyi bir çözüme ulaşmıştır. CPU süresi olarak ise karşılaştırma yapılan algoritmalarla benzer süreler elde edilmiştir. Aynı problem kümesi için YAK algoritması değerlendirildiğinde 4 problem için bilinen en iyi çözüme ulaşılmış ve CPU süresi açısından AA'na göre daha düşük değerler elde edilmiştir. p148 problem kümesi için AA ve YAK algoritmaları bilinen en iyi çözüme daha düşük CPU zamanları ile ulaşmış, diğer taraftan her iki algoritma da çevrim süresinin 306 olduğu problem için bilinen en iyi çözümden daha iyi bir çözüm bulmuştur. Son olarak p205 problem kümesi için AA 10 çevrim süresinin 7'sinde, YAK algoritması ise 5'inde bilinen en iyi çözüme ulaşmıştır. Bütün problem kümelerine genel olarak bakılacak olursa AA 45 problemden 42'sinde, YAK algoritması da 39'unda bilinen en iyi çözümü bulmuştur. Diğer taraftan AA 2 kere, YAK algoritması da 1 kere daha önce elde edilmemiş bir çözüme ulaşmıştır. Her iki algoritmada da basit komşuluk yapılarının kullanıldığı göz önüne alınırsa AA ve YAK algoritmalarının özel kısıta sahip olmayan ÇTMHDP üzerinde etkin bir performansa sahip olduğu söylenebilir.

Tablo 4.16. AA ve YAK algoritması ile elde edilen bölgesel kısıta sahip ÇTMHDP sonuçlarının karşılaştırılması.

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Özcan ve Toklu (2009b)		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst sayısı	CPU
p9	3	7	<0.02	7	<0.02	7	<1	7*	1.421	**6**	15225.09
	4	**5**	<0.02	**5**	<0.02	6	<1	**5**	0.234	**5**	6399.62
	5	**4**	<0.02	**4**	<0.02	**4**	<1	**4**	0.156	**4**	3806.94
	6	**3**	<0.02	**3**	<0.02	**3**	<1	**3**	0.203	**3**	3109.53

* Özcan ve Toklu (2009b) çalışması da matematiksel model sonuçlarını içermesine rağmen GAMS 22.7.2 sonuçları ile farklılık olmasının sebebi, pozitif bölgesel kısıta sahip iki işin aynı istasyonunun farklı yönlerine atanmasına izin verilmemesinden kaynaklanmaktadır.

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Özcan ve Toklu (2009b)		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst sayısı	CPU
p12	5	**6**	<0.02	**6**	<0.02	**6**	<1	**6**	1.640	-	-
	6	**5**	<0.02	**5**	<0.02	**5**	<1	**5**	4.656	-	-
	7	**4**	<0.02	**4**	<0.02	5	<1	**4**	0.281	-	-
	8	**4**	<0.02	**4**	<0.02	-	-	**4**	2.562	7	7200

	Çevrim süresi	AA		YAK		Özcan ve Toklu (2009b)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p16	15	**6**	3.657	7	<0.02	**6**	13.640
	16	**6**	2.625	7	<0.02	**6**	30.234
	18	**6**	<0.02	**6**	<0.02	**6**	0.296
	19	**6**	<0.02	**6**	<0.02	**6**	0.359
	20	**5**	<0.02	**5**	<0.02	**5**	0.875
	21	**5**	0.093	**5**	0.078	**5**	1.046
	22	**5**	0.512	**5**	<0.02	**5**	2.015

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)		Özcan ve Toklu (2009b)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p24	18	**8**	<0.02	**8**	<0.02	-	-	**8**	<7200
	20	**8**	<0.02	**8**	<0.02	**8**	<1	**8**	3999.265
	24	**6**	0.109	**6**	<0.02	-	-	**6**	6604.968
	25	**6**	<0.02	**6**	<0.02	**6**	<1	**6**	<7200
	30	**5**	<0.02	**5**	<0.02	**5**	<1	**5**	<7200
	35	**4**	<0.02	5	<0.02	5	<1	**4**	242.890
	40	**4**	<0.02	**4**	<0.02	**4**	<1	**4**	<7200

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p65	326	**17**	0.250	18	0.218	**17**	3.52
	381	**14**	9.328	**14**	3.687	15	<1
	435	**13**	0.265	**13**	0.390	**13**	2.78
	490	**11**	6.015	12	0.265	12	<1
	544	**10**	0.687	**10**	0.296	**10**	1.85

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p148	204	**26**	6.421	**26**	2.203	**26**	10.32
	255	**21**	1.218	**21**	2.125	**21**	3.64
	306	**18**	2.000	**18**	0.546	**18**	463.39
	357	**15**	14.531	**15**	0.578	18	2.06
	408	**14**	0.734	**14**	0.703	15	2.02
	459	**12**	2.625	**12**	0.265	13	465.92
	510	**11**	4.359	**11**	2.406	**11**	6.76

	Çevrim süresi	AA		YAK		Baykasoğlu ve Dereli (2008)	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p205	1133	**23**	41.593	24	8.031	25	264.32
	1322	**21**	1.843	**21**	24.187	22	264.31
	1510	**18**	11.609	**18**	78.390	19	270.34
	1699	**17**	13.250	18	4.390	18	264.28
	1888	**16**	5.015	**16**	4.703	**16**	263.91
	2077	**14**	260.031	**14**	83.625	16	266.76
	2266	**14**	12.078	**14**	4.328	**14**	259.72
	2454	**14**	5.843	**14**	4.750	**14**	258.44
	2643	**13**	94.578	14	5.671	**13**	259.79
	2832	**12**	9.692	13	12.156	**12**	258.85

Bilimsel yazında bölgesel kısıta sahip ÇTMHDP alanında iki çalışma bulunmaktadır. Bunlardan ilki karınca koloni optimizasyon algoritmasını kullanan Baykasoğlu ve Dereli (2008) çalışması, ikincisi ise küçük boyutlu problemler üzerinde matematiksel modelle çözüm bulan Özcan ve Toklu (2009b) çalışmasıdır. Tablo 4.16'dan görüldüğü gibi p9 problem kümesinde çevrim süresinin 3 olduğu problem için, önerilen matematiksel modelle bulunan çözüm, karşılaştırma yapılan diğer sonuçlardan daha düşük istasyon sayısına sahiptir. Özcan ve Toklu (2009b) çalışması da matematiksel model sonuçlarını içermesine rağmen arada böyle bir fark çıkmasının nedeni, pozitif bölgesel kısıta sahip iki işin aynı istasyonunun farklı yönlerine atanmasına izin verilmemesinden kaynaklanmaktadır. Ancak tez çalışmasında temel alınan ve Simaria ve Vilarinho (2007) çalışmasında önerilen matematiksel modelde, bilimsel yazındaki ilkelere bağlı olarak pozitif bölgesel kısıta sahip iki işin aynı istasyonun farklı yönlerine atanmasına izin verilmiştir. AA ve YAK algoritmalarının p9 problem kümesi üzerindeki performansları incelendiğinde çevrim süresinin 3 olduğu problem hariç en iyi çözüme daha düşük CPU süreleri ile ulaşıldığı görülmektedir. p12 problem kümesi için ise önerilen her iki algoritma ile de en iyi çözüm, daha düşük CPU süreleri ile elde edilmiştir. p16 ve p24 problem kümelerinin bütün çevrim süreleri için AA en iyi çözüme daha düşük CPU süreleri ile ulaşmış ancak YAK algoritması p16 için 2 problemde, p24 için ise 1 problemde daha yüksek istasyon sayısını elde etmiştir. AA p65 problem kümesi için oluşturulan 5 problemin 3'ünde bilinen en iyi çözüme ulaşmış, 2'sinde ise bilinen en iyi çözümden daha iyi bir çözüm (*ct*=381, 490) elde etmiştir. YAK algoritması ise 5 problemin 3'ünde en düşük istasyon sayısını bulmuş ve bunlardan biri yine bilinen en iyi çözümden daha iyi bir çözüm (*ct*=381) olmuştur. p148 problem kümesi için AA ve YAK algoritmaları bütün çevrim sürelerinde en düşük istasyon sayısına sahip çözümleri elde etmiş ve bunlardan 3'ü (*ct*=357, 408, 459) ilk kez bulunmuştur. Son olarak p205 problem kümesi için AA 10 problemin hepsinde, YAK algoritması ise 6'sında en düşük istasyon sayısına sahip çözümleri bulmuştur. Her iki algoritmanın CPU süreleri de karşılaştırma yapılan algoritmaya göre çok daha düşüktür. Bütün problem kümeleri genel olarak incelenecek olursa AA 44 problemden 43'ünde, YAK algoritması da 34'ünde bilinen en iyi çözüme ulaşmıştır. Diğer taraftan AA 10 kere, YAK algoritması da 7 kere daha önce bilinmeyen iyi bir çözüme ulaşmıştır. Netice

olarak AA ve YAK algoritmaları bölgesel kısıta sahip ÇTMHDP üzerinde etkin bir performansa sahiptir

Tablo 4.17. AA ve YAK algoritması ile elde edilen konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP sonuçlarının karşılaştırılması.

	Çevrim süresi	AA		YAK		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p9	4	**5**	<0.02	**5**	<0.02	**5**	421.09
	5	**4**	<0.02	**4**	<0.02	**4**	200.09
	6	**4**	<0.02	**4**	<0.02	**4**	292.77

	Çevrim süresi	AA		YAK		GAMS 22.7.2	
		İst. sayısı	CPU	İst. sayısı	CPU	İst. sayısı	CPU
p12	5	**6**	<0.02	**6**	<0.02	**6**	53173.70
	6	**5**	<0.02	**5**	<0.02	**5**	17337.06
	7	**5**	<0.02	**5**	<0.02	**5**	46576.48
	8	**4**	<0.02	**4**	<0.02	**4**	11225.56

	Çevrim süresi	AA		YAK	
		İst. sayısı	CPU	İst. sayısı	CPU
p16	15	**7**	<0.02	**7**	<0.02
	16	**7**	<0.02	**7**	<0.02
	18	**7**	<0.02	**7**	<0.02
	19	**7**	<0.02	**7**	<0.02
	20	**7**	<0.02	**7**	<0.02
	21	**7**	<0.02	**7**	<0.02
	22	**6**	<0.02	**6**	<0.02

	Çevrim süresi	AA		YAK	
		İst. sayısı	CPU	İst. sayısı	CPU
p24	18	**8**	<0.02	**8**	<0.02
	20	**8**	<0.02	**8**	<0.02
	24	**7**	0.062	**7**	<0.02
	25	**7**	<0.02	**7**	<0.02
	30	**6**	<0.02	**6**	<0.02
	35	**6**	<0.02	**6**	<0.02
	40	**6**	<0.02	**6**	<0.02

	Çevrim süresi	AA		YAK	
		İst. sayısı	CPU	İst. sayısı	CPU
p65	326	**18**	0.234	**18**	0.203
	381	**15**	0.625	**15**	1.593
	435	**14**	0.390	**14**	0.437
	490	**13**	0.281	**13**	0.375
	544	**12**	0.578	**12**	0.296

	Çevrim süresi	AA		YAK	
		İst. sayısı	CPU	İst. sayısı	CPU
p148	204	**26**	19.093	**26**	1.328
	255	**23**	2.421	**23**	0.578
	306	**20**	4.359	**20**	1.437
	357	**18**	2.015	**18**	3.046
	408	**17**	13.593	**17**	2.000
	459	**17**	2.078	**17**	20.375
	510	**17**	2.468	**17**	1.359

	Çevrim süresi	AA		YAK	
		İst. sayısı	CPU	İst. sayısı	CPU
p205	1133	25	51.843	**24**	115.640
	1322	**23**	10.875	**23**	24.937
	1510	21	16.984	**20**	242.500
	1699	**19**	926.781	20	55.296
	1888	**19**	287.109	**19**	22.328
	2077	19	34.218	**18**	960.812
	2266	18	41.734	**17**	352.734
	2454	18	48.312	**17**	56.937
	2643	18	55.828	**17**	42.046
	2832	18	66.093	**17**	55.640

Bilimsel yazın araştırmasında da belirtildiği gibi konumsal, bölgesel ve senkronizasyon kısıtlarının aynı anda ele alındığı bir ÇTMHDP çözüm yaklaşımı bulunmamaktadır. Dolayısıyla bu bölümdeki karşılaştırmalarda AA, YAK algoritması ve 4.5.3 bölümünde verilen GAMS 22.7.2 optimizasyon paket programı sonuçları değerlendirilecektir. p9 ve p12 problem kümesi için bütün çevrim sürelerinde her iki algoritma da GAMS 22.7.2 optimizasyon paket programı ile bulunan optimum çözümleri çok daha kısa sürede elde etmiştir. Ancak problemin boyutu arttıkça hesaplama süresinin de üstel olarak artması sebebiyle daha büyük boyutlu problemler için optimum değerler elde edilememiştir. Bu sebeple bu tür problemler üzerindeki etkinlik analizi için AA ve YAK algoritmaları kendi aralarında karşılaştırılacaktır. p16, p24, p65 ve p148 problem kümeleri için her iki algoritma da aynı değerlere ulaşmıştır. İki algoritma arasındaki farklılık p205 problem kümesinde ortaya çıkmaktadır. AA 10 problemden 3'ünde en düşük istasyon sayısına sahip çözümü bulurken, YAK algoritması 9 problemde en düşük istasyon sayısına sahip çözümü elde etmiştir. Görüldüğü gibi YAK algoritması konumsal, bölgesel ve senkronizasyon kısıtlarına sahip büyük boyutlu ÇTMHDP'inde AA'na göre daha iyi performans göstermektedir. İki algoritma CPU süresi açısından karşılaştırıldığında ise genel olarak YAK algoritmasının daha az CPU süresi gerektirdiği söylenebilir.

ÇTMHDP bilimsel yazındaki zor problemlerden biridir. Probleme özel kısıtların da çözüme dâhil edilmesiyle problem daha da zorlaşmaktadır. Bilimsel yazın araştırmasından da görüldüğü gibi özel kısıtların dâhil olduğu çalışmalar oldukça azdır. AA ve YAK algoritmalarının deneysel çalışma bölümünde bahsedilen her 3 problem türüne de uygulanmasıyla oldukça tatmin edici sonuçlar elde edilmiştir. Algoritmaların ve komşuluk yapıların en basit hâllerinin kullanıldığı da göz önüne alındığında her iki algoritmanın da ÇTMHDP üzerinde etkin bir performansa sahip olduğu görülmektedir.

4.9. Arı Algoritması ile Bulanık Çok Amaçlı Çift Taraflı Montaj Hattı Dengeleme Problemi Çözüm Yaklaşımı

Birçok gerçek hayat problemi dilsel ve/veya kesin olmayan değişkenler, kısıtlar ve amaçlar içerir. Sistem çevresinin sabit olmayan bir yapıya sahip olması ve kesin veriler elde etmenin yüksek maliyet gerektirmesi sebebiyle kesin verileri toplamak genel olarak çok zordur. Gerçek hayat sistemlerindeki bu belirsizliğin aşılabilmesi için genellikle bulanık küme teorisine dayalı bulanık matematiksel programlama yaklaşımı kullanılır [186].

Bulanık matematiksel programlama modelleri bulanık bileşenlerine göre; bulanık amaçlara sahip, bulanık amaç fonksiyonu katsayılarına sahip ve bulanık sağ taraf sabitlerine sahip modeller olmak üzere 3 sınıfa ayrılabilir [186]. Bulanık amaçlara sahip ÇTMHDP ise bulanık çok amaçlı matematiksel programlama olarak adlandırılan ilk sınıflandırmaya dâhil olup amaçlara ait aspirasyon seviyelerinin belirlenmesine dayanan Bulanık Amaç Programlama (BAP) teknikleriyle çözülebilmektedir.

Bulanık çok amaçlı matematiksel programlama problemlerinin çözümü için gerçekleştirilen ilk temel çalışma Zimmermann'a [187] ait olup Bellman ve Zadeh [188] tarafından önerilen maks-min operatörüne dayalı maks-min yöntemini içermektedir. Sinha [189] maks-min yöntemini çok seviyeli programlama modellerinin; Chakraborty ve Gupta [190] ise bulanık çok amaçlı doğrusal kesirli programlama problemlerinin çözümü için kullanmışlardır. Diğer taraftan Chanas [191] bulanık doğrusal programlama problemlerinin çözümü için parametrik programlama tekniği yaklaşımını önermiş; Gen ve ark. [192] bulanık doğrusal olmayan amaç programlama problemlerinin çözümü için genetik algoritmayı kullanmış; Baykasoğlu ve Göçken [193] ise bulanık amaç programlama problemlerinin çözümü için çok amaçlı bir tabu

arama algoritması önermişlerdir. Ayrıca Narasimhan [194] BAP probleminin çözümü için eşit ağırlıklara sahip çoklu amaçlar için bir çözüm yöntemi geliştirmiş, Hannan [195] ise aynı problem için doğrusal ve kesikli üyelik fonksiyonlarını kullanmıştır. Lu ve ark. [196] ise karar vericilerin bulanık amaçları herhangi bir üyelik fonksiyonuyla ifade etmesine izin veren ve çözüm süreci boyunca karar vericilere etkileşimli karar seçenekleri sunan interaktif bulanık amaç eniyileme yöntemini önermişlerdir. Son olarak Baykasoğlu ve Göçken [186] bulanık matematiksel programlama ve çözüm yaklaşımları üzerine geniş bir inceleme ve sınıflandırma çalışması gerçekleştirmişlerdir.

BAP problemi mevcut bulanık amaçları sağlayacak en iyi *D* kararının bulunması şeklinde aşağıdaki gibi tanımlanmaktadır.

$$\sum_{j=1}^{n} c_{pj}x_j \cong g_p \qquad p = 1,2,\dots,k_1 \tag{4.2}$$

$$\sum_{j=1}^{n} c_{pj}x_j \gtrsim g_p \qquad p = k_1+1,\dots,k_2 \tag{4.3}$$

$$\sum_{j=1}^{n} c_{pj}x_j \lesssim g_p \qquad p = k_2+1,\dots,k_3 \tag{4.4}$$

$Kısıtlar$

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad i = 1,2,\dots,m \tag{4.5}$$

$$x_j \geq 0 \qquad j = 1,2,\dots,n \tag{4.6}$$

Yukarıdaki ifadede x_j karar değişkenlerini, g_p amaçlarla ilişkilendirilmiş aspirasyon seviyelerini, c_{pj} ve a_{ij} amaçlara ve kısıtlara ait teknolojik katsayıları, b_i ise mevcut kaynak miktarını göstermektedir.

Bulanık amaç fonksiyonu, üyelik fonksiyonları ile belirlenmekte; üyelik fonksiyonları ise genellikle doğrusal, doğrusal olmayan ve üstel fonksiyonlarla ifade edilmektedir. Bu çalışmada doğrusal üyelik fonksiyonları kullanılmış olup minimizasyon, maksimizasyon ve eşitlik durumlarındaki doğrusal üyelik fonksiyonu ifadeleri aşağıdaki gibi tanımlanmıştır [186].

$$Bulanık\ minimum\ (' \lesssim '):\ \mu(c_{pj}x_j) = \begin{cases} 1 & c_{pj}x_j \le g_p \\ \dfrac{(g_p + d_p^R) - c_{pj}x_j}{d_p^R} & g_p \le c_{pj}x_j \le g_p + d_p^R \\ 0 & c_{pj}x_j \ge g_p + d_p^R \end{cases} \quad (4.7)$$

$$Bulanık\ maksimum\ (' \gtrsim '):\ \mu(c_{pj}x_j) = \begin{cases} 1 & c_{pj}x_j \ge g_p \\ \dfrac{c_{pj}x_j - (g_p - d_p^L)}{d_p^L} & g_p - d_p^L \le c_{pj}x_j \le g_p \\ 0 & c_{pj}x_j \le g_p - d_p^L \end{cases} \quad (4.8)$$

$$Bulanık\ eşitlik\ (' \cong '):\ \mu(c_{pj}x_j) = \begin{cases} 0 & c_{pj}x_j \le g_p - d_p^L \\ \dfrac{c_{pj}x_j - (g_p - d_p^L)}{d_p^L} & g_p - d_p^L \le c_{pj}x_j \le g_p \\ 1 & c_{pj}x_j = g_p \\ \dfrac{(g_p + d_p^R) - c_{pj}x_j}{d_p^R} & g_p \le c_{pj}x_j \le g_p + d_p^R \\ 0 & c_{pj}x_j \ge g_p + d_p^R \end{cases} \quad (4.9)$$

Yukarıdaki ifadelerde d_p^R maksimum sağ tolerans sınırını, d_p^L ise maksimum sol tolerans sınırını göstermektedir.

Ulaşılan bulanık karar ise bulanık amaç fonksiyonlarının kesişimi olarak ifade edilmekte olup bulanık karar kümesine ait üyelik fonksiyonu $\mu_D(x)$ ve maksimizasyon kararı aşağıdaki gibi tanımlanmaktadır [187].

$$\mu_D(x) = \mu_1(c_{1j}x_j) \wedge \mu_2(c_{2j}x_j) \wedge \ldots \wedge \mu_{k_3}(c_{k_3 j}x_j) = \min_p \mu_p(c_{pj}x_j) \quad (4.10)$$

$$\max_x \mu_D(x) = \max_x \min_p \mu_p(c_{pj}x_j) \quad (4.11)$$

Bulanık amaçların analizi için maks-min, öncelik ve toplamsal yöntemler olmak üzere 3 teknik kullanılmıştır. Maks-min yöntemi, BAP problemlerine Narasimhan [194] tarafından uygulanan bir yöntem olup amaçlara ait en düşük üyelik fonksiyonu değerlerinin maksimizasyonunu amaçlamaktadır ($maks - min\ \mu_g(x)$). Öncelik yönteminde ise bulanık amaçlara önem derecelerine göre bir öncelik verilmekte ve düşük önceliğe sahip amaçlar ancak yüksek önceliğe sahip amaçlar sağlandığında ele alınmaktadır. Dolayısıyla öncelikle yüksek önceliğe sahip amaçlara ait üyelik fonksiyonları maksimize edilmektedir [193]. Tiwari ve ark. [197] tarafından önerilen toplamsal yöntemde ise amaçlara ait üyelik fonksiyonu değerlerinin toplamı maksimize

edilmektedir ($max \sum_{i=1}^{m} \mu_i$). Her yöntem için komşu çözümler arasından mevcut en iyi çözümün seçim süreci aşağıdaki gibi gerçekleşmektedir.

Maks-min yöntemi:

Adım 1. Her komşu çözüm için amaçlara ait üyelik fonksiyonu değerleri hesaplanır ve içlerinden en düşük değere sahip olan belirlenir.

Adım 2. Minimum üyelik değerleri içinden en büyük değere sahip olan çözüm, mevcut en iyi çözüm olarak seçilir.

Öncelik yöntemi:

Adım 1. Önceden belirlenen öncelik değerlerine göre her komşu çözüm için amaçlara ait üyelik değerleri hesaplanır.

Adım 2. Her çözüm için en yüksek önceliğe sahip amaca ait üyelik değeri kontrol edilir ve en yüksek değere sahip olan çözüm seçilir. En yüksek önceliğe sahip amaç için birden fazla alternatif komşu çözüm söz konusuysa sonraki önceliğe sahip amacın üyelik değerleri kontrol edilir.

Toplamsal yöntem:

Adım 1. Her komşu çözüm için amaçlara ait üyelik değerleri hesaplanarak toplanır.

Adım 2. En yüksek toplam değere sahip komşu çözüm, mevcut en iyi çözüm olarak seçilir.

4.9.1 Bulanık Çok Amaçlı Çift Taraflı Montaj Hattı Dengeleme Problemi için Arı Algoritması

Çalışmanın bu aşamasında özel kısıta sahip olmayan ÇTMHDP ele alınmış olup ilgili problemin çözümü için Bölüm 4.6'da detayları verilmiş olan AA kullanılmıştır. ÇTMHDP'nin bulanık amaçlar içermesi sebebiyle algoritmada kullanılan uygunluk fonksiyonu farklılaştırılmıştır. Bulanık çok amaçlı ÇTMHDP 3 farklı bulanık amaca sahip bir BAP modeli olarak aşağıdaki gibi tanımlanmıştır.

Amaç-1 Gevşeklik indeksinin maksimizasyonu

Gevşeklik indeksi iki bağlantılı iş arasındaki boş zamanı ifade etmektedir. Bağlantılı işler ise öncelik diyagramında birbiriyle direkt bağlantılı işleri göstermektedir. Çift taraflı montaj hatlarında, aralarında yakın öncelik ilişkisi olan iki iş karşılıklı

istasyonlara atandığında, biri bitmeden diğeri başlayamayacağından iki iş arasında boş zaman oluşabilmekte ve bu durumda gevşeklik indeksinin dikkate alınması gerekmektedir. Gevşeklik indeksi [0,1] arasında değerler almakta olup indeks değeri arttıkça bağlantılı işler arasındaki boş zaman azalmaktadır. Gevşeklik indeksinin maksimizasyonunu içeren ilk amaç aşağıdaki gibi tanımlanmıştır.

$$\frac{\sum_{j=1}^{n}\left\{\sum_{i\in Q_j}\sqrt{t_{ij}^{s}-t_{i'j'}^{f}}+\sum_{i\in I_j-Q_j}\sqrt{CT}\right\}}{m\sqrt{CT}} \gtrsim amaç_{ws} \tag{4.12}$$

Yukarıdaki eşitsizlikte *CT* çevrim süresini, *j* ve j' karşılıklı istasyonları, I_j *j* istasyonuna atanan işler kümesini, Q_j hemen önceki işlerinden biri j' istasyonuna atanan işler kümesini, t_{ij}^{s} ve t_{ij}^{f} sırasıyla *j* istasyonundaki *i* işinin başlangıç ve bitiş sürelerini temsil etmektedir.

Amaç-2 Karşılıklı istasyon sayısının minimizasyonu

$$m_R + m_L \lesssim amaç_{opt} \tag{4.13}$$

m_R sağ yöndeki istasyon sayısını, m_L sol yöndeki istasyon sayısını göstermektedir.

Amaç-3 Hat etkinliğinin maksimizasyonu

$$LE \lesssim amaç_{le} \tag{4.14}$$

LE hat etkinliğini temsil etmekte olup aşağıdaki gibi hesaplanmaktadır.

$$LE = \frac{LB}{m_r + m_l} \tag{4.15}$$

$$LB = 2 * Maks + maks\left\{0, \left\lceil\frac{ETotal - (Max * CT - LTotal) - (Max * CT - RTotal)}{CT}\right\rceil\right\} \tag{4.16}$$

$$Maks = maks\left\{\left\lceil\frac{LTotal}{CT}\right\rceil, \left\lceil\frac{RTotal}{CT}\right\rceil\right\} \tag{4.17}$$

LTotal, *RTotal* ve *ETotal* sırasıyla sol, sağ ve herhangi bir yöne atanabilecek işlerin toplam süresini göstermektedir. Çift taraflı montaj hattı için bir alt sınır elde edebilmek için öncelikle sağ ve sol işlerin atanması için gerekli istasyon sayısı belirlenmektedir. Her istasyon karşılıklı iki istasyona sahip olduğundan *Max* değeri 2 ile çarpılmıştır. Herhangi bir yöne atabilecek işler için ise öncelikle çevrim süresini aşmayan mevcut istasyonlara atama yapılmakta daha sonra kalan işler için yeni istasyonlar açılmaktadır [177].

Netice olarak AA'nda kullanılan uygunluk fonksiyonu, ele alınan yönteme göre aşağıdaki gibi tanımlanmıştır.

$$maks\ \mu_D(\sigma^s) = \min(\mu_{ws}, \mu_{opt}, \mu_{le}) \qquad (maks - min\ yöntemi\ için)$$

$$maks\ \mu_D(\sigma^s) = (\mu_{opt} \gg \mu_{le} \gg \mu_{ws}) \quad (öncelik\ yöntemi\ için)$$

$$maks\ \mu_D(\sigma^s) = \mu_{ws} + \mu_{opt} + \mu_{le} \qquad (toplamsal\ yöntem\ için)$$

4.9.2 Deneysel Çalışma

Bulanık çok amaçlı ÇTMHDP'nin çözümü için kullanılan AA, C# programlama dilinde kodlanarak 2.2 GHz CPU ve 2 GB RAM özelliklere sahip Intel Core 2 Duo PC kullanılarak bilimsel yazındaki test problemlerine uygulanmıştır. Algoritma parametrelerinin değerleri Tablo 4.18'de, amaçlara ait aspirasyon seviyeleri ve tolerans değerleri ise Tablo 4.19'da sunulmaktadır.

Tablo 4.18. Bulanık çok amaçlı ÇTMHDP için AA parametrelerinin değerleri.

Parametre	Değer
S	25
P	15
e	5
nep	3
nsp	2
MaksIter	200

Tablo 4.19. Amaçlara ait aspirasyon seviyeleri ve tolerans değerleri.

Amaç	Aspirasyon seviyesi	Tolerans	Değer
$amaç_{ws}$	0.95	d_{ws}^L	0.1
$amaç_{opt}$	opt	d_{opt}^R	2
$amaç_{le}$	$\frac{LB}{opt}$	d_{le}^L	$\frac{LB}{opt} - \frac{LB}{opt+2}$

Birinci ve üçüncü amaçlar bulanık maksimizasyon formunda iken ikinci amaç bulanık minimizasyon formundadır. Verilen aspirasyon seviyesi ve tolerans değerlerine göre ilgili üyelik fonksiyonları aşağıdaki gibi tanımlanmıştır.

Bulanık gevşeklik amacı için üyelik fonksiyonu,

$$\mu_{ws} = \begin{cases} 1 & z_{ws} \geq 0.95 \\ \dfrac{z_{ws} - 0.85}{0.1} & 0.85 \leq z_{ws} \leq 0.95 \\ 0 & z_{ws} \leq 0.85 \end{cases} \tag{4.18}$$

$$z_{ws} = \frac{\sum_{j=1}^{n} \left\{ \sum_{i \in Q_j} \sqrt{t_{ij}^{s} - t_{i'j'}^{f}} + \sum_{i \in I_j - Q_j} \sqrt{CT} \right\}}{m\sqrt{CT}} \tag{4.19}$$

Bulanık karşılıklı istasyon sayısı amacı için üyelik fonksiyonu,

$$\mu_{opt} = \begin{cases} 1 & z_{opt} \leq opt \\ \dfrac{opt + 2 - z_{opt}}{2} & opt \leq z_{opt} \leq opt + 2 \\ 0 & z_{opt} \geq opt + 2 \end{cases} \tag{4.20}$$

$$z_{opt} = m_R + m_L \tag{4.21}$$

Bulanık hat etkinliği amacı için üyelik fonksiyonu,

$$\mu_{LE} = \begin{cases} 1 & z_{LE} \geq \dfrac{LB}{opt} \\ \dfrac{z_{LE} - \dfrac{LB}{opt+2}}{\dfrac{LB}{opt} - \dfrac{LB}{opt+2}} & \dfrac{LB}{opt+2} \leq z_{LE} \leq \dfrac{LB}{opt} \\ 0 & z_{LE} \leq \dfrac{LB}{opt+2} \end{cases} \tag{4.22}$$

$$z_{LE} = LE \tag{4.23}$$

(4.18)-(4.23) formülasyonlarında; z_{ws}, z_{opt}, z_{LE} amaç fonksiyonu değerlerini, μ_{ws}, μ_{opt}, μ_{LE} ise ilgili amaçlara ait üyelik fonksiyonlarını temsil etmektedir. Diğer taraftan *opt* bilinen en iyi çözümü, *LB* ise karşılıklı istasyon sayısı için alt sınırı göstermektedir.

AA 10 kez çalıştırılmış ve her amaç için maks-min, öncelik ve toplamsal yöntemler ile elde edilen ortalama amaç fonksiyonu ve ortalama üyelik fonksiyonu değerleri sırasıyla Tablo 4.20, Tablo 4.21 ve Tablo 4.22'de verilmiştir.

Tablo 4.20. Amaç-1'e ait deneysel sonuçlar.

	Çevrim süresi	Maks-min		Öncelik		Toplamsal	
		Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri
p9	3	0.953	1	0.953	1	0.953	1
	4	1	1	1	1	1	1
	5	0.956	0.988	0.956	0.988	0.952	0.965
	6	1	1	1	1	1	1
p12	4	1	1	1	1	1	1
	5	0.985	1	0.995	1	0.995	1
	6	0.970	1	0.972	1	0.968	1
	7	0.994	1	0.988	1	0.991	1
	8	0.996	1	0.985	1	0.989	1
p16	15	0.965	1	0.961	1	0.965	1
	16	0.979	1	0.975	1	0.984	1
	18	0.992	1	0.986	1	0.981	1
	19	0.972	1	0.971	1	0.968	1
	20	0.976	1	0.973	1	0.982	1
	21	0.953	0.975	0.946	0.950	0.943	0.925
	22	0.916	0.671	0.914	0.649	0.923	0.739
p24	18	0.972	1	0.968	1	0.974	1
	20	0.969	1	0.962	1	0.960	1
	24	0.936	0.850	0.943	0.880	0.939	0.873
	25	0.961	1	0.966	1	0.961	1
	30	0.969	1	0.965	1	0.969	1
	35	0.970	1	0.970	1	0.972	1
	40	0.963	1	0.960	1	0.962	1
p65	326	0.958	1	0.959	1	0.955	1
	381	0.925	0.758	0.934	0.845	0.939	0.873
	435	0.954	1	0.953	0.994	0.953	0.998
	490	0.918	0.685	0.924	0.753	0.925	0.764
	544	0.929	0.812	0.935	0.855	0.929	0.795
p148	204	0.957	1	0.957	1	0.956	1
	255	0.955	1	0.955	1	0.953	1
	306	0.921	0.723	0.919	0.694	0.923	0.740
	357	0.934	0.849	0.935	0.858	0.936	0.871
	408	0.909	0.600	0.909	0.598	0.913	0.638
	459	0.929	0.794	0.928	0.783	0.928	0.787
	510	0.906	0.568	0.904	0.547	0.910	0.607
p205	1133	0.878	0.286	0.884	0.348	0.871	0.215
	1322	0.896	0.468	0.886	0.364	0.885	0.359
	1510	0.876	0.270	0.878	0.292	0.876	0.269
	1699	0.869	0.197	0.873	0.238	0.877	0.283
	1888	0.815	0.008	0.838	0.017	0.826	0.001
	2077	0.812	0.010	0.823	0	0.862	0.209
	2266	0.856	0.062	0.813	0.019	0.839	0.030
	2454	0.811	0	0.846	0.136	0.860	0.139
	2643	0.796	0	0.793	0	0.791	0
	2832	0.811	0.011	0.788	0	0.790	0

Tablo 4.21. Amaç-2'ye ait deneysel sonuçlar.

	Çevrim süresi	Maks-min		Öncelik		Toplamsal	
		Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri
p9	3	6	1	6	1	6	1
	4	5	1	5	1	5	1
	5	4	1	4	1	4	1
	6	3	1	3	1	3	1
p12	4	7	1	7	1	7	1
	5	6	1	6	1	6	1
	6	5	1	5	1	5	1
	7	4	1	4	1	4	1
	8	4	1	4	1	4	1
p16	15	6	1	6	1	6	1
	16	6	1	6	1	6	1
	18	6	1	6	1	6	1
	19	5	1	5	1	5	1
	20	5	1	5	1	5	1
	21	5	1	5	1	5	1
	22	4	1	4	1	4	1
p24	18	8	1	8	1	8	1
	20	8	1	8	1	8	1
	24	6	1	6	1	6	1
	25	6	1	6	1	6	1
	30	5	1	5	1	5	1
	35	4	1	4	1	4	1
	40	4	1	4	1	4	1
p65	326	17	1	17	1	17	1
	381	14	1	14	1	14	1
	435	13	1	13	1	13	1
	490	11.100	0.950	11	1	11	1
	544	10	1	10	1	10	1
p148	204	26	1	26	1	26	1
	255	21	1	21	1	21	1
	306	17	1	17	1	17	1
	357	15	1	15	1	15	1
	408	13.200	0.900	13	1	13	1
	459	12	1	12	1	12	1
	510	11	1	11	1	11	1
p205	1133	23	0.500	23	0.500	22.900	0.550
	1322	20.900	0.550	20	1	20	1
	1510	18	0.500	18	0.500	17.900	0.550
	1699	16	0.500	16	0.500	16	0.500
	1888	16.200	0.100	14	0.500	14	0.500
	2077	15.500	0	13	0.500	14.800	0.200
	2266	13	0.500	12	1	12	1
	2454	12.800	0	12.600	0.100	12.900	0.050
	2643	12	0	10	1	10	1
	2832	10.600	0.700	10	1	10	1

Tablo 4.22. Amaç-3'e ait deneysel sonuçlar.

	Çevrim süresi	Maks-min		Öncelik		Toplamsal	
		Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri	Ort. amaç fonk. değeri	Ort. üyelik fonk. değeri
p9	3	1	1	1	1	1	1
	4	1	1	1	1	1	1
	5	1	1	1	1	1	1
	6	1	1	1	1	1	1
p12	4	1	1	1	1	1	1
	5	0.833	1	0.833	1	0.833	1
	6	1	1	1	1	1	1
	7	1	1	1	1	1	1
	8	1	1	1	1	1	1
p16	15	1	1	1	1	1	1
	16	1	1	1	1	1	1
	18	0.833	1	0.833	1	0.833	1
	19	1	1	1	1	1	1
	20	1	1	1	1	1	1
	21	0.800	1	0.800	1	0.800	1
	22	1	1	1	1	1	1
p24	18	1	1	1	1	1	1
	20	0.875	1	0.875	1	0.875	1
	24	1	1	1	1	1	1
	25	1	1	1	1	1	1
	30	1	1	1	1	1	1
	35	1	1	1	1	1	1
	40	1	1	1	1	1	1
p65	326	0.941	1	0.941	1	0.941	1
	381	1	1	1	1	1	1
	435	0.923	1	0.923	1	0.923	1
	490	0.991	0.945	1	1	1	1
	544	1	1	1	1	1	1
p148	204	1	1	1	1	1	1
	255	1	1	1	1	1	1
	306	1	1	1	1	1	1
	357	1	1	1	1	1	1
	408	0.985	0.892	1	1	1	1
	459	1	1	1	1	1	1
	510	1	1	1	1	1	1
p205	1133	0.913	0.478	0.913	0.478	0.917	0.530
	1322	0.861	0.528	0.900	1	0.900	1
	1510	0.888	0.472	0.888	0.472	0.893	0.524
	1699	0.875	0.468	0.875	0.468	0.875	0.468
	1888	0.808	0.092	0.928	0.464	0.928	0.464
	2077	0.776	0	0.923	0.461	0.819	0.184
	2266	0.846	0.461	0.916	1	0.916	1
	2454	0.785	0	0.800	0.090	0.780	0.045
	2643	0.750	0	0.900	1	0.900	1
	2832	0.853	0.690	0.900	1	0.900	1

Deneysel çalışma sonuçlarının karşılaştırılmasında üyelik fonksiyonları kullanılmıştır. Bütün amaçlar için de en iyi üyelik fonksiyonu değerleri toplamsal yöntemle elde edilmiştir. Öncelik yönteminin performansının ise toplamsal yönteme yakın olduğu tespit edilmiştir. 3 yöntem içinde en kötü üyelik fonksiyonu değerleri ise maks-min yöntemi ile elde edilmiştir.

AA'nın bulanık çok amaçlı ÇTMHDP üzerindeki performansının değerlendirilme aşamasında ise elde edilen karşılıklı istasyon sayıları, bilimsel yazındaki bilinen en iyi değerlerle karşılaştırılmıştır. Tablo 4.23'te her 3 yöntemle elde edilen karşılıklı istasyon sayıları ve bilinen en iyi istasyon sayıları; minimum, ortalama CPU süreleri ve standart sapma değerleri ile birlikte verilmiştir.

Tablo 4.23. Minimum karşılıklı istasyon sayıları ve CPU süresi sonuçları.

	Çevrim süresi	Bilinen en iyi karşılıklı istasyon sayısı	Minimum karşılıklı istasyon sayısı			CPU(sn)								
						Minimum			Ortalama			Standart sapma		
			Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal
p9	3	6	6	6	6	0.015	0.015	0.015	0.019	0.021	0.021	0.014	0.019	0.019
	4	5	5	5	5	0.015	0.015	0.015	0.019	0.021	0.021	0.014	0.019	0.019
	5	4	4	4	4	0.015	0.015	0.015	0.165	0.085	0.121	0.144	0.082	0.144
	6	3	3	3	3	0.015	0.015	0.015	0.019	0.019	0.022	0.014	0.014	0.024
p12	4	7	7	7	7	0.015	0.015	0.015	0.021	0.021	0.021	0.019	0.019	0.019
	5	6	6	6	6	0.015	0.015	0.015	0.021	0.019	0.019	0.019	0.014	0.014
	6	5	5	5	5	0.015	0.015	0.015	0.021	0.019	0.019	0.019	0.014	0.014
	7	4	4	4	4	0.015	0.015	0.015	0.030	0.032	0.026	0.029	0.029	0.029
	8	4	4	4	4	0.015	0.015	0.015	0.022	0.021	0.021	0.024	0.019	0.019
p16	15	6	6	6	6	0.015	0.015	0.015	0.066	0.088	0.091	0.054	0.092	0.135
	16	6	6	6	6	0.015	0.015	0.015	0.065	0.050	0.068	0.045	0.049	0.074
	18	6	6	6	6	0.015	0.015	0.015	0.051	0.040	0.043	0.041	0.039	0.057
	19	5	5	5	5	0.015	0.015	0.015	0.029	0.030	0.038	0.025	0.036	0.058
	20	5	5	5	5	0.031	0.031	0.031	0.147	0.096	0.149	0.191	0.072	0.120
	21	5	5	5	5	0.015	0.015	0.015	0.493	0.594	0.354	0.460	0.613	0.639
	22	4	4	4	4	0.062	0.046	0.015	0.321	0.151	0.476	0.361	0.117	0.529
p24	18	8	8	8	8	0.015	0.015	0.015	0.093	0.055	0.062	0.080	0.055	0.055
	20	8	8	8	8	0.015	0.015	0.015	0.022	0.022	0.022	0.024	0.024	0.024
	24	6	6	6	6	0.078	0.281	0.109	1.101	1.146	1.201	0.666	0.655	0.812
	25	6	6	6	6	0.015	0.015	0.015	0.132	0.055	0.105	0.155	0.044	0.081
	30	5	5	5	5	0.015	0.015	0.015	0.069	0.046	0.068	0.073	0.040	0.070
	35	4	4	4	4	0.015	0.015	0.015	0.063	0.087	0.069	0.075	0.055	0.052
	40	4	4	4	4	0.015	0.015	0.015	0.032	0.029	0.029	0.038	0.027	0.024

	Çevrim süresi	Bilinen en iyi karşılıklı istasyon sayısı	Minimum karşılıklı istasyon sayısı			CPU(sn)								
						Minimum			Ortalama			Standart sapma		
			Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal	Maks-min	Öncelik	Toplamsal
p65	326	17	17	17	17	1.203	3.609	0.750	15.421	14.580	11.316	13.566	7.920	8.202
	381	14	14	14	14	11.140	22.171	5.875	44.618	67.304	60.474	19.369	27.839	27.300
	435	13	13	13	13	7.812	5.781	13.453	50.670	31.869	39.185	27.316	20.801	24.360
	490	11	11	11	11	0.375	18.781	11.546	69.165	60.010	70.865	32.206	26.910	32.990
	544	10	10	10	10	36.703	11.250	5.109	77.083	59.130	50.237	24.129	34.090	35.700
p148	204	26	26	26	26	1.265	2.109	1.296	3.730	4.335	4.918	3.260	2.186	2.773
	255	21	21	21	21	1.203	10.031	1.265	51.976	34.587	47.151	42.673	21.037	40.568
	306	17	17	17	17	9.390	15.125	15.140	76.769	78.660	84.951	38.058	49.022	41.431
	357	15	15	15	15	16.343	9.796	24.515	102.558	52.302	90.351	54.871	43.833	45.926
	408	13	13	13	13	0.531	52.375	3	86.371	92.618	90.082	58.033	44.813	52.455
	459	12	12	12	12	3.296	34.531	14.734	86.340	101.451	92.963	71.960	35.413	45.129
	510	11	11	11	11	11.015	8.265	6.453	96.702	87.697	82.921	54.366	52.556	51.759
p205	1133	22	23	23	**22**	151.750	92.546	54.656	747.263	742.557	715.909	426.397	391.590	430.616
	1322	20	20	20	20	837.421	351.062	23.875	1339.126	1190.586	898.960	381.860	558.150	611.382
	1510	17	18	18	**17**	144.062	412.171	76.546	1121.315	1134.218	1267.214	573.696	534.308	769.849
	1699	15	16	16	16	266.453	380.531	430.265	1077.115	1033.017	1088.767	650.224	514.496	530.091
	1888	13	14	14	14	2315.218	41.421	87.515	2529.403	714.208	779.487	285.713	521.125	605.364
	2077	12	14	**13**	**13**	2357.421	102.796	112.843	2389.973	510.971	1260.771	17.491	489.187	1003.693
	2266	12	13	**12**	**12**	240.734	8.203	7.843	1184.706	434.285	709.605	827.492	886.861	836.005
	2454	10	12	**11**	**11**	2878.968	411.875	412.171	2919.115	1610.444	1561.831	26.721	859.235	752.507
	2643	10	12	**10**	**10**	2965.171	8.265	24.078	2999.575	216.743	152.904	22.610	195.708	112.440
	2832	10	10	10	10	2278.484	8.015	8.843	3204.028	8.946	9.466	347.679	0.544	0.728

Deneysel çalışmanın gerçekleştirildiği 45 test probleminin, toplamsal yöntemle 41'inde, öncelik yöntemiyle 39'unda ve maks-min yöntemiyle 37'sinde bilinen en iyi istasyon sayısına ulaşılmıştır. CPU süresi açısından kullanılan yöntemler değerlendirildiğinde ise toplamsal ve öncelik yöntemlerinin maks-min yöntemine göre daha kısa sürelere sahip olduğu görülmüştür. Diğer taraftan AA ile hem bilinen en iyi karşılıklı istasyon sayılarına büyük ölçüde ulaşılmış hem de diğer amaçlar etkin bir şekilde sağlanmıştır. Sonuç olarak AA bulanık çok amaçlı ÇTMHDP'ni çözmede etkin bir performansa sahiptir.

5. BÖLÜM
SONUÇ VE ÖNERİLER

5.1. Çalışmanın Katkıları

Tez çalışmasının bu bölümünde çalışmanın orijinalliğinden ve bilimsel yazına olan katkılarından bahsedilecektir.

Birçok gerçek hayat probleminin kombinatoryal optimizasyon problemi olarak modellenebilmesi, tez çalışmanın çıkış noktasını oluşturmuş olup bu tür problemlerin çözümünde hızlı ve etkin olarak kullanılabilecek araçların geliştirilmesi ya da mevcut araçların kullanılması ele alınmıştır. Zor kombinatoryal optimizasyon problemlerinin kesin yöntemlerle çözümünün mümkün olamayabilmesi ve problemin boyutu arttıkça hesaplama süresinin büyük bir sorun teşkil etmesi sebebiyle sürü zekâsına dayalı algoritmalardan yararlanılmıştır.

Bu doğrultuda, bilgisayar ve iletişim ağları, yerleştirme problemleri, araç rotalama, grup teknolojisi ve çizelgeleme gibi gerçek hayat uygulamalarına sahip zor kombinatoryal optimizasyon problemlerinden GAP'nin çözümü için sürü zekâsı tabanlı algoritmalardan AA ve YAK algoritmasından faydalanılmıştır. İlgili algoritmalar GAP'nin çözümü için farklılaştırılmış ve önerilen her iki algoritmanın da GAP üzerinde özellikle de karşılaştırma yapılan algoritmalara göre iyi bir performansa sahip olduğu görülmüştür.

Çalışmanın diğer bir uygulama alanında ise bilimsel yazında önerilen farklı komşuluk yapılarının GAP için geliştirilmiş olan AA'nın performansı üzerindeki etkileri incelenmiş ve bu komşuluk yapılarının farklılıkları ortaya konmuştur.

Diğer taraftan otobüs ve kamyon gibi büyük boyutlu ürünlerin üretildiği işletmelerde ortaya çıkan ve yine NP-zor bir yapıya sahip olan ÇTMHDP ele alınmış ve özel kısıtlar çözüm yaklaşımına dâhil edildikçe problemin daha da zorlaştığı belirtilmiştir. Yapılan

bilimsel yazın taraması sonucunda özel kısıtların dâhil olduğu çözüm yaklaşımlarının oldukça az olduğu; özellikle konumsal, bölgesel ve senkronizasyon kısıtlarının aynı anda çözüm yaklaşımına dahil edildiği bir çalışmanın bulunmadığı görülmüştür. Bu amaçla yine AA ve YAK algoritmalarından faydalanılarak farklı kısıtlar altında algoritmaların performansı incelenmiştir. Her iki algoritmada da basit komşuluk yapılarının kullanıldığı da göz önüne alınırsa AA ve YAK algoritmasının ÇTMHDP üzerinde üstün bir performansa sahip olduğu görülmüştür. Ayrıca bilimsel yazında ilk kez konumsal, bölgesel ve senkronizasyon kısıtlarına sahip ÇTMHDP için karma tamsayılı doğrusal olmayan bir matematiksel model önerilmiştir.

Diğer taraftan AA, bulanık çok amaçlı ÇTMHDP'nin çözümü için kullanılmış ve bulanık amaçlar farklı teknikler altında incelenerek bu tekniklerin algoritma performansı üzerindeki etkisi incelenmiştir.

Sonuç olarak genellikle sürekli optimizasyon problemlerine uygulanmış olan AA ve YAK algoritmasının karmaşık tamsayılı optimizasyon problemleri üzerindeki performansı incelenmiş; geniş deneysel çalışmalar sonucunda GAP ve ÇTMHDP için bilimsel yazındaki kesin ve sezgisel çözüm yöntemleriyle yapılan karşılaştırmalar, önerilen algoritmaların etkin bir performansa sahip olduğunu göstermiştir.

Tez çalışmasında elde edilen sonuçlardan iki SCI'e giren [198-200], bir de SCI'e indeksine girmeyen [201] dergilere makaleler gönderilmiş olup, bu makalelerden ilki basılmış diğerleri ise inceleme aşamasındadır. Ayrıca bir kitap bölümü [202] çalışması gerçekleştirilmiş olup, bir uluslararası kongre [203] ve üç ulusal kongre bildirisi [204-207] sunulmuştur.

5.2.İleriye Yönelik Öneriler

Zor kombinatoryal optimizasyon problemlerine arıların yiyecek arama davranışına dayalı metasezgisel yöntemlerle çözüm aramaya dayanan bu tez çalışmasında, uygulama alanı olarak GAP ve ÇTMHDP seçilmiştir. Bu problemlere ek olarak atama esnasında kullanılan kaynak kullanım miktarlarının ve ajanlara ait kaynak kapasitelerinin stokastik değerlere sahip olduğu Stokastik GAP, işlem sürelerinin stokastik değerlere sahip olduğu Stokastik ÇTMHDP ve diğer birçok zor kombinatoryal optimizasyon probleminin çözümünde de arı sistemi uygulamaları gerçekleştirilebilir.

Bilindiği gibi parametrelerin en iyi seviyelerinin belirlenmesi, algoritma performansını etkileyen en önemli unsurlardan biridir. Ele alınan GAP ve ÇTMHDP'nin her ikisi için de gerçekleştirilecek bir parametre optimizasyonu çalışması, daha iyi çözümler bulunmasını sağlayabilir. Ayrıca farklı çözüm dizisi gösterimleri ve uygunluk fonksiyonu kullanımının algoritma performansı üzerindeki etkisi incelenebilir.

Diğer taraftan her iki problem için de birbiriyle çelişen birden çok amaç aynı uygunluk fonksiyonu üzerinde değerlendirilerek farklı etkinlik ölçütleri eşzamanlı olarak en iyilebilir. Örneğin ÇTMHDP için hattın dengelenmesi, istasyon sayısının en küçüklenmesi, hat etkinliğinin en büyüklenmesi ve birbiriyle ilişkili işler arasındaki boş zamanın en küçüklenmesi gibi amaçlar aynı anda çözüm yaklaşımına dâhil edilebilir.

GAP'nde kullanılan komşuluk yapılarının algoritma performansı üzerindeki etkisinin incelendiği bölümde ise komşuluk yapıları ikili ve üçlü olarak değerlendirilebilir ve bu kombinasyonların performansa etkisi analiz edilebilir.

ÇTMHDP'nin çözümü için kullanılan AA ve YAK algoritmasında basit komşuluk yapıları kullanılmıştır. Deneysel çalışmalardan oldukça tatmin edici sonuçlar elde edilse de karmaşık komşuluk yapılarının algoritmalara dâhil edilmesiyle daha iyi çözümler elde edilebilecektir. Diğer taraftan çift taraflı montaj hatlarında aynı anda birden fazla benzer tipteki modelin karma olarak üretildiği ÇTKMMHDP de dikkate alınabilir. Bilimsel yazın araştırmasından da görüldüğü gibi ÇTKMMHDP üzerine yapılan çalışmalar oldukça az olup tek modelli hatlar üzerinde etkin bir performans gösteren AA ve YAK algoritmasının performansı karma modelli hatlar üzerinde de incelenebilir.

KAYNAKLAR

1. Baykasoglu, A., Goal Programming Using the Multiple Objective Tabu Search, Journal of Operational Research Society, 52, 1359-1369, 2001.
2. Von Frisch, K., The Dance Language and Orientation of Bees, Harvard University Press, Cambridge, Massachusetts, 1976.
3. Michelsen, A., Andersen, B.B., Storm, J., Kirchner, W.H., Lindauer, M., How Honeybees Perceive Communication Dances, Studied by Means of a Mechanical Model, Behavioral Ecology and Sociobiology, 30(3-4), 143-150, 1992.
4. Fisher, M.L., Jaikumar, R., Van Wassenhove, L.N., A Multiplier Adjustment Method for the Generalized Assignment Problem, Management Science, 32(9), 1095-1103, 1986.
5. Bartholdi, J.J., Balancing Two-Sided Assembly Lines: A Case Study, International Journal of Production Research, 31(10), 2447-2461, 1993.
6. Pham, D.T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., Zaidi, M., The Bees Algorithm - A Novel Tool for Complex Optimisation Problems, In Proceedings of Innovative Production Machines and Systems Virtual Conference, 454-461, 2006a.
7. Karaboga, D., An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
8. Karaboga, D., Basturk, B., A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, 39(3), 459-471, 2007a.
9. Bonabeau, E., Dorigo, M., Theraulaz, G., Swarm Intelligence: From Natural to Artificial Systems, New York, Oxford University Press, 1999.
10. Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., Trubian, M., Heuristics from Nature for Hard Combinatorial Optimization Problems, International Transactions in Operational Research, 3(1), 1-21, 1996.
11. Engelbrecht, A.P., Fundamentals of Computational Swarm Intelligence, John Wiley and Sons Ltd, England, 2005.

12. Kennedy, J., Eberhart, R., Particle Swarm Optimization, In Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, 1942-1948, 1995.

13. Dorigo, M., Optimization, Learning and Natural Algorithms, PhD Thesis, Politecnico di Milano, Italy, 1992.

14. Yonezawa, Y., Kikuchi, T., Ecological Algorithm for Optimal Ordering Used by Collective Honey Bee Behavior, In Proceedings of 7th International Symposium on Micro Machine and Human Science, 249-256, 1996.

15. Schmickl, T., Thenius, R., Crailsheim, K., Simulating Swarm Intelligence in Honey Bees: Foraging in Differently Fluctuating Environments, In Proceedings of Genetic and Evolutionary Computation Conference, Washington DC, USA, 273-274, 2005.

16. Lemmens, N., To Bee or Not to Bee: A Comparative Study in Swarm Intelligence, Master's Thesis, Institute for Knowledge and Agent Technology, Maastricht ICT Competence Centre, Maastricht University, Maastricht, Netherlands, 2006.

17. Adams, J., Rothman, E.D., Kerr, W.E., Paulino, Z.L., Estimation of the Number of Sex Alleles and Queen Matings from Diploid Male Frequencies in a Population of Apis Mellifera, Genetics, 86(3), 583–596, 1977.

18. Page, R.E., Kimsey, R.B., Laidlaw, H.H., Migration and Dispersal of Spermatozoa in Spermathecae of Queen Honey Bees: Apis Mellifera, Experientia, 40, 182–184, 1984.

19. Dietz, A., Evolution, In: Rinderer, T.E. (eds.), Bee Genetics and Breeding, Academic Press, Inc, 3-22, 1986.

20. Laidlaw, H.H., Page, R.E., Mating Designs, In: Rinderer, T.E., (eds.), Bee Genetics and Breeding, Academic Press, Inc, 323-341, 1986.

21. Rinderer, T.E., Collins, A.M., Behavioral Genetics, In: Rinderer, T.E., (eds.), Bee Genetics and Breeding, Academic Press, Inc, 155-176, 1986.

22. Sato, T., Hagiwara, M., Bee System: Finding Solution by a Concentrated Search, In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, 4(C), 3954-3959, 1997.

23. Yang, X.S., Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, Lecture Notes in Computer Science, 3562, 317-323, 2005.

24. Abbass, H.A., A Single Queen Single Worker Honey-Bees Approach to 3-SAT, In Proceedings of Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001a.

25. Abbass, H.A., A Monogenous MBO Approach to Satisfiability, In Proceedings of International Conference on Computational Intelligence for Modeling, Control and Automation, Las Vegas, USA, 2001b.

26. Abbass, H.A., MBO: Marriage in Honey Bees Optimization A Haplometrosis Polygynous Swarming Approach, In Proceedings of Congress of Evolutionary Computation, Seoul, Korea, 207-214, 2001c.

27. Teo, J., Abbass, H.A., An Annealing Approach to the Mating-Flight Trajectories in the Marriage in Honey Bees Optimization Algorithm, Technical Report CS04/01, School of Computer Science, University of New South Wales at ADFA, 2001.

28. Teo, J., Abbass, H.A., A True Annealing Approach to the Marriage in Honey-Bees Optimization Algorithm, International Journal of Computational Intelligence and Applications, 3(2), 199-211, 2003.

29. Chang, H.S., Converging Marriage in Honey-Bees Optimization and Application to Stochastic Dynamic Programming, Journal of Global Optimization, 35, 423-441, 2006.

30. Bozorg Haddad, O., Afshar, A., Mariano, M.A., Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization, Water Resources Management, 20, 661-680, 2006.

31. Afshar, A., Bozorg Haddad, O., Marino, M.A., Adams, B.J., Honey-Bee Mating Optimization (HBMO) Algorithm for Optimal Reservoir Operation, Journal of the Franklin Institute, 344(5), 452-462, 2007.

32. Arefi, A., Haghifam, M.R., Fathi, S.H., Nikham, T., Olamaei, J., A Novel Algorithm Based on Honey Bee Mating Optimization for Distribution Harmonic State Estimation Including Distributed Generators, In Proceedings of Power Tech Conference, Bucharest, Romania, 2009.

33. Marinakis, Y., Marinaki, M., A Hybrid Honey Bees Mating Optimization Algorithm for the Probabilistic Traveling Salesman Problem, In Proceedings of IEEE Congress on Evolutionay Computation, 1762-1769, 2009.

34. Bozorg Haddad, O., Afshar, A., MBO Algorithm, A New Heuristic Approach in Hydrosystems Design and Operation, In Proceedings of International Conference on Managing Rivers in the 21st Century, 499-504, 2004.

35. Fathian, M., Amiri, B., Maroosi, A., Application of Honey-Bee Mating Optimization Algorithm on Clustering, Applied Mathematics and Computation, 190(2), 1502-1513, 2007.

36. Horng, M.H., A Multilevel Image Thresholding Using the Honey Bee Mating Optimization, Applied Mathematics and Computation, 215, 3302-3310, 2009.

37. Jung, S.H., Queen-Bee Evolution for Genetic Algorithms, Electronic Letters, 39(6), 575-576, 2003.

38. Azeem, M.F., Saad, A.M., Modified Queen Bee Evolution Based Genetic Algorithm for Tuning of Scaling Factors of Fuzzy Knowledge Base Controller, In Proceedings of India Annual Conference, 299-303, 2004.

39. Qin, L.D., Jiang, Q.Y., Zou, Z.Y., Cao, Y.J., A Queen-Bee Evolution Based on Genetic Algorithm for Economic Power Dispatch, In Proceedings of 39th Internatinoal Conference of Universities Power Engineering, Bristol, UK, 453-456, 2004.

40. Karcı, A., Imitation of Bee Reproduction as a Crossover Operator in Genetic Algorithms, In Proceedings of 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zeland, LNAI, 3157, 1015-1016, 2004.

41. Lucic, P., Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing, Ph.D. Dissertation, Civil Engineering, Faculty of the Virginia Polytechnic Institute and State University, 2002.

42. Lucic, P., Teodorovic, D., Bee System: Modeling Combinatorial Optimization Transportation Engineering Problems by Swarm Intelligence, In Proceedings of Triennial Symposium on Transportation Analysis, Sao Miguel, Azores Islands, 441-445, 2001.

43. Lucic, P., Teodorovic, D., Transportation Modeling: An Artificial Life Approach, In Proceedings of the 14th International Conference on Tools with Artificial Intelligence, Washington DC, USA, 216-223, 2002.
44. Lucic, P., Teodorovic, D., Computing with Bees: Attacking Complex Transportation Engineering Problems, International Journal on Artificial Intelligence Tools, 12(3), 375-394, 2003a.
45. Lucic, P., Teodorovic, D., Vehicle Routing Problem with Uncertain Demand at Nodes: The Bee System and Fuzzy Logic Approach. In: Verdegay J.L. (eds.), Fuzzy Sets in Optimization, Springer-Verlag, Berlin Heidelbelg, 67-82, 2003b.
46. Teodorovic, D., Dell'Orco, M., Bee Colony Optimization - A Cooperative Learning Approach to Complex Transportation Problems, In: Jaszkiewicz, A. et al. (eds.), Advanced OR and AI Methods in Transportation, 51-60, 2005.
47. Wong, L.P., Low, M.Y.H., Chong, C.S., A Bee Colony Optimization Algorithm for Traveling Salesman Problem, In Proceedings of the 2nd Asia International Conference on Modelling & Simulation, 818-823, 2008.
48. Nakrani, S., Tovey, C., On Honey Bees and Dynamic Allocation in an Internet Server Colony, In Proceedings of 2nd International Workshop Mathematics and Algorithms of Social Insects, Atlanta, Georgia, USA, 2003.
49. Wedde, H.F., Farooq, M., Zhang, Y., BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior, Lecture Notes in Computer Science, 3172, 83-94, 2004.
50. Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.Y., A Bee Colony Optimization Algorithm to Job Shop Scheduling, In Proceedings of the 37th Conference on Winter Simulation, Monterey, California,1954-1961, 2006.
51. Koudil, M., Benatchba, K., Tarabet, A., Sahraoui, E.B., Using Artificial Bees to Solve Partitioning and Scheduling Problems in Codesign, Applied Mathematics and Computation, 186(2), 1710-1722, 2007.
52. Kumar, R., Sharma, D., Kumar, A., Economic Power Dispatch with Valve Point Effects Using Bee Optimization Algorithm, Journal of Electrical Engineering & Technology, 4(1), 19-27, 2009.

53. Chokpanyasuwan, C., Pothiya, S., Bhasaputra, P., Honey Bee Colony Optimization to Solve Economic Dispatch Problem with Generator Constraints, In Proceedings of 6th International Conference of Electrical Engineering / Electronics, Computer, Telecommunications and Information Technology, Pattaya, Thailand, 2009.

54. Bianco, G.M., Getting Inspired from Bees to Perform Large Scale Visual Precise Navigation, In Proceedings of International Conference on Intelligent Robots and Systems, Sendai, Japan, 619-624, 2004.

55. Drias, H., Sadeg, S., Yahi, S., Cooperative Bees Swarm for Solving the Maximum Weighted Satisfiability Problem, Lecture Notes in Computer Science, 3512, 318-325, 2005.

56. Benatchba, K., Admane, L., Koudil, M., Using Bees to Solve a Data Mining Problem Expressed as a Max-Sat One, In Proceedings of International Conference on Interplay between Natural and Artificial Computation, Canary Islands, Spain, 212-220, 2005.

57. Quijano, N., Passino, K.M., 2007, Honey Bee Social Foraging Algorithms for Resource Allocation Theory and Application, Submitted for journal publication, Engineering Applications of Artificial Intelligence, 2008.

58. Markovic, G.Z., Teodorovic, D., Acimovic-Raspopovic. V.S., Routing and Wavelength Assignment in All-Optical Networks Based on the Bee Colony Optimization, AI Communications, 20(4), 273-285, 2007.

59. Tereshko, V., Reaction–Diffusion Model of a Honeybee Colony's Foraging Behaviour, In: Schoenauer M. (eds.), Parallel Problem Solving from Nature VI, Lecture Notes in Computer Science, 1917, Springer–Verlag, Berlin, 807-816, 2000.

60. Seeley, T.D., The Wisdom of the Hive, Harvard University Press, Cambridge, 1995.

61. Pham, D.T., Castellani, M., The Bees Algorithm: Modelling Foraging Behaviour to Solve Continuous Optimization Problems, In Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 223(12), 2919-2938, 2009.

62. Basturk, B., Karaboga, D., An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization, In Proceedings of IEEE Swarm Intelligence Symposium, Indianapolis, Indiana, USA, 2006.

63. Karaboga, D., Basturk, B., On the Performance of Artificial Bee Colony (ABC) Algorithm, Applied Soft Computing, 8(1), 687-697, 2008.

64. Karaboga, D., Akay, B., Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization, In Proceedings of Innovative Production Machines and Systems Virtual Conference, Cardiff, UK, 2009b.

65. Karaboga, D., Akay, B., A Comparative Study of Artificial Bee Colony Algorithm, Applied Mathematics and Computation, 214, 108-132, 2009a.

66. Akay, B., Karaboga, D., Parameter Tuning for the Artificial Bee Colony Algorithm, In Proceedings of 1st International Conference on Computational Collective Intelligence - Semantic Web, Social Networks & Multiagent Systems, Wroclaw, Poland, 2009a.

67. Karaboga, D., Basturk, B., Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, LNCS: Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, Springer- Verlag, 4529, 789-798, 2007b.

68. Goldberg, D.E., Deb, K., A Comparison of Selection Schemes Used in Genetic Algorithms, Rawlins, G.J.E. (eds.), Foundations of Genetic Algorithms, 69-93, 1991.

69. Akay, B., Karaboga, D., Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm, In Proceedings of XI. Conferences on Advances in Artificial Intelligence by the Italian Association for Artificial Intelligence, Reggio Emilia, 2009b.

70. Quan, H., Shi, X., On the Analysis of Performance of the Improved Artificial-Bee-Colony Algorithm, In Proceedings of 4th International Conference on Natural Computation, 654-658, 2008.

71. Tsai, P.W., Pan, J.S., Liao, B.Y., Chu, S.C., Interactive Artificial Bee Colony (IABC) Optimization, In Proceedings of Intelligence and Security Informatics, Taiwan, 2008.

72. Kang, F., Li, J., Xu, Q., Structural Inverse Analysis by Hybrid Simplex Artificial Bee Colony Algorithms, Computers & Structures, 87(13-14), 861-870, 2009.

73. Nelder, J.A., Mead, R., A Simplex Method for Function Minimization, Computer Journal, 7, 308-313, 1965.

74. Bao, L., Zeng, J.C., Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm, In Proceedings of 9th International Conference on Hybrid Intelligent Systems, 2009.

75. Kuo, T., Huang, S.Y., Using Disruptive Selection to Maintain Diversity in Genetic Algorithms, Applied Intelligence, 7(3), 257-267, 1997.

76. Blickle, T., Thiele, L., A Mathematical Analysis of Tournament Selection, In Proceedings of the 6th International Conference on Genetic Algorithms, San Francisco, California, 9-16, 1995.

77. Mezura-Montes, E., Cetina-Domínguez, O., Exploring Promising Regions of the Search Space with the Scout Bee in the Artificial Bee Colony for Constrained Optimization, In Proceedings of the Artificial Neural Networks in Engineering Conference, 2009.

78. Hamida, S.B., Schoenauer, M., ASCHEA: New Results Using Adaptive Segregational Constraint Handling, In Proceedings of the Congress on Evolutionary Computation, Piscataway, New Jersey, 1, 884-889, 2002.

79. Duan, H.B., Xu, C.F., Xing, Z.H., A Hybrid Artificial Bee Colony Optimization and Quantum Evolutionary Algorithm for Continuous Optimization Problems, International Journal of Neural Systems, 20(1), 39-50, 2010.

80. Tsai, P.W., Pan, J.S., Liao, B.Y., Chu, S.C., Enhanced Artificial Bee Colony Optimization, International Journal of Innovative Computing, Information and Control, 5(12), 2009.

81. Karaboga, D., Ozturk, C., Neural Networks Training by Artificial Bee Colony Algorithm on Pattern Classification, Neural Network World, 19(3), 279-292, 2009.

82. Srinivasa Rao, R., Narasimham, S.V.L., Ramalingaraju, M., Optimization of Distribution Network Configuration for Loss Reduction Using Artificial Bee Colony Algorithm, International Journal of Electrical Power and Energy Systems Engineering, 1(2), 2008.

83. Bendeş, E., Özkan, C., Direk Lineer Trasformasyon Yönteminde Yapay Zeka Tekniklerinin Uygulanması, 2. Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Sempozyumu, Kayseri, Türkiye, 2008.

84. Karaboga, N., A New Design Method Based on Artificial Bee Colony Algorithm for Digital IIR Filters, Journal of the Franklin Institute, 346(4), 328-348, 2009.

85. Singh, A., An Artificial Bee Colony Algorithm for the Leaf-Constrained Minimum Spanning Tree Problem, Applied Soft Computing, 9(2), 625-631, 2009.

86. Rao, R.V., Pawar, P.J., Modelling and Optimization of Process Parameters of Wire Electrical Discharge Machining, In Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 223, 1431-1440, 2009.

87. Kurban, T., Beşdok, E., A Comparison of RBF Neural Network Training Algorithms for Inertial Sensor Based Terrain Classification, Sensors, 9, 6312-6329, 2009.

88. Kumar, S.K., Tiwari, M.K., Babiceanu, R.F., Minimisation of Supply Chain Cost with Embedded Risk Using Computational Intelligence Approaches, International Journal of Production Research, 48(13), 3717-3739, 2009.

89. Li, D.M., Cheng, B.Y., Artificial bee colony algorithm for scheduling a single batch processing machine with non-identical job sizes, Journal of Sichuan University (Natural Science Edition), 46(3), 2009.

90. Bahamish, H.A.A., Abdullah, R., Salam, R.A., Protein Tertiary Structure Prediction Using Artificial Bee Colony Algorithm, In Proceedings of 3rd Asia International Conference on Modelling & Simulation, 258-263, 2009.

91. Pacurib, J., Seno, G.M., Yusiong, J.P., Solving Sudoku Puzzles Using Improved Artificial Bee Colony Algorithm, In Proceedings of 4th International Conference on Innovative Computing, Information and Control, Kaohsiung, Taiwan, 2009.

92. Omkar, S.N., Senthilnath, J., Artificial Bee Colony for Classification of Acoustic Emission Signal, International Journal of Aerospace Innovations, 1(3), 129-143, 2009.

93. Karaboga, D., Akay, B., A Survey: Algorithms Simulating Bee Swarm Intelligence, Artificial Intelligence Review, 31(1), 68-85, 2009c.

94. Pham, D.T., Soroka, A.J., Ghanbarzadeh, A., Koc, E., Otri, S., Packianather, M., Optimising Neural Networks for Identification of Wood Defects Using the Bees Algorithm, In Proceedings of IEEE International Conference on Industrial Informatics, Singapore, 2006b.

95. Packianather, M.S., Drake, P.R., Identifying Defects on Plywood Using a Minimum Distance Classifier and A Neural Network. In: Pham D.T., Eldukhri E., Soroka A. (eds.), In Proceedings of 1st Virtual International Conference on Intelligent Production Machines and Systems, Elsevier, Oxford, 543-548, 2005.

96. Pham, D.T., Koç, E., Ghanbarzadeh, A., Otri, S., Optimisation of the Weights of Multi-Layered Perceptrons Using the Bees Algorithm, In Proceedings of 5th International Symposium on Intelligent Manufacturing Systems, Turkey, 2006c.

97. Pham, D.T., Ghanbarzadeh, A., Koç, E., Otri, S., Application of the Bees Algorithm to the Training of Radial Basis Function Networks for Control Chart Pattern Recognition, In Proceedings of 5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering, Ischia, Italy, 2006d.

98. Pham, D.T., Otri, S., Ghanbarzadeh, A., Koc, E., Application of the Bees Algorithm to the Training of Learning Vector Quantisation Networks for Control Chart Pattern Recognition, In Proceedings of Information and Communication Technologies, Syria, 1624-1629, 2006e.

99. Pham, D.T., Afify, A., Koç, E., Manufacturing Cell Formation Using the Bees Algorithm, In Proceedings of Innovative Production Machines and Systems Virtual Conference, Cardiff, UK, 2007a.

100. Pham, D.T., Koç, E., Lee, J.Y., Phrueksanant, J., Using the Bees Algorithm to Schedule Jobs for a Machine, In Proceedings of 8th International Conference on Laser Metrology, CMM and Machine Tool Performance, Euspen, UK, Cardiff, 430-439, 2007b.

101. Pham, D.T., Haj Darwish, A., Eldukhri, E.E., Otri, S., Using the Bees Algorithm to Tune a Fuzzy Logic Controller for a Robot Gymnast, In Proceedings of 3rd International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 546-551, 2007c.

102. Pham, D.T., Otri, S., Afify, A.A., Mahmuddin, M., Al-Jabbouli, H., Data Clustering Using the Bees Algorithm, In Proceedings of 40th CIRP Intelligent Manufacturing Systems Seminar, Liverpool, 2007d.

103. Pham, D.T., Ghanbarzadeh, A., Multi-Objective Optimisation Using the Bees Algorithm, In Proceedings of 3rd IPROMS International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2007.

104. Pham, D.T., Muhamad, Z., Mahmuddin, M., Ghanbarzadeh, A., Koç, E., Otri, S., Using the Bees Algorithm to Optimise a Support Vector Machine for Wood Defect Classification. In Proceedings of Innovative Production Machines and Systems Virtual Conference, Cardiff, UK, 2007e.

105. Pham, D.T., Mahmuddin, M., Otri, S., Al-Jabbouli, H., Application of the Bees Algorithm to the Selection Features for Manufacturing Data, In Proceedings of 3rd International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2007f.

106. Pham, D.T., Otri, S., Haj Darwish, A., Application of the Bees Algorithm to PCB Assembly Optimisation, In Proceedings of 3rd International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 511-516, 2007g.

107. Pham, D.T., Al-Jabbouli, H., Mahmuddin, M., Otri, S., Haj Darwish, A., Application of the Bees Algorithm to Fuzzy Clustering, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008a.

108. Pham, D.T., Ang, M.C., Ng, K.W., Otri, S., Haj Darwish, A., Generating Branded Product Concepts: Comparing the Bees Algorithm and an Evolutionary Algorithm, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008b.

109. Pham, D.T., Castellani, M., Fahmy, A.A., Learning the Inverse Kinematics of a Robot Manipulator Using the Bees Algorithm, In Proceedings of 17th IFAC World Congress, South Korea, 493-498, 2008c.

110. Pham, D.T., Castellani, M., Sholedol, M., Ghanbarzadeh, A., The Bees Algorithm and Mechanical Design Optimisation, In Proceedings of 5th International Conference on Informatics in Control, Automation and Robotics, 2008d.

111. Pham, D.T., Ghanbarzadeh, A., Otri, S., Koç, E., Optimal Design of Mechanical Components Using the Bees Algorithm, In Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2009a.

112. Pham, D.T., Lee, J.Y., Haj Darwish, A., Soroka, A.J., Multi-Objective Environmental/Economic Power Dispatch Using the Bees Algorithm with Pareto Optimality, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008e.

113. Lee, J.Y., Haj Darwish, A., Multi-Objective Environmental/Economic Dispatch Using the Bees Algorithm with Weighted Sum, In Proceedings of EU-Korea Conference on Science and Technology, Germany, 267-274, 2008.

114. Pham, D.T., Haj Darwish, A., Fuzzy Selection of Local Search Sites in the Bees Algorithm, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008.

115. Pham, D.T., Pham, Q.T., Ghanbarzadeh, A., Castellani, M., Dynamic Optimisation of Chemical Engineering Processes Using the Bees Algorithm, In Proceedings of 17th IFAC World Congress, South Korea, 6100-6105, 2008f.

116. Pham, D.T., Sholedolu, M., Using a Hybrid PSO-Bees Algorithm to Train Neural Networks for Wood Defect Classification, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008.

117. Pham, D.T., Negm, M.A., Otri, S., Using the Bees Algorithm to Solve a Stochastic Optimisation Problem, In Proceedings of 4th International Virtual Conference on Intelligent Production Machines and Systems, Whittles, Dunbeath, Scotland, 2008g.

118. Martello, S., Toth, P., An Algorithm for the Generalized Assignment Problems. In: Brans J.P. (eds.), Operational Research, North-Holland, 589-603, 1981.

119. Martello, S., Toth, P., Knapsack Problems: Algorithms and Computer Implementations, Wiley, New York, 1990.

120. Cattrysse, D., Set Partitioning Approaches to Combinatorial Optimization Problems, Ph.D. Dissertation, Katholieke Universiteit Leuven, Centrum Industrieel Beleid, Belgium, 1990.

121. Cattrysse, D., Salomon, M., Van Wassenhove, L.N., A Set Partitioning Heuristic for the Generalized Assignment Problem, European Journal of Operational Research, 72, 167-174, 1994.

122. Öncan, T., A Survey of the Generalized Assignment Problem and Its Applications, Information Systems and Operational Research, 45(3), 123-141, 2007.

123. Ross, G.T., Soland, P.M., A Branch and Bound Based Algorithm for the Generalized Assignment Problem, Mathematical Programming, 8, 91-103, 1975.

124. Savelsbergh, M., A Branch-and-Price Algorithm for the Generalized Assignment Problem, Operations Research, 45, 831-841, 1997.

125. Nauss, R.M., Solving the Generalized Assignment Problem: An Optimizing and Heuristic Approach, Informs Journal of Computing, 15(3), 249-266, 2003.

126. Osman, I.H., Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches, OR Spektrum, 17, 211-225, 1995.

127. Chu, P.C., Beasley, J.E., A Genetic Algorithm for the Generalized Assignment Problem, Computers and Operations Research, 24, 17-23, 1997.

128. Racer, M., Amini, M.M., A Robust Heuristic for the Generalized Assignment Problem, Annals of Operations Research, 50, 487-503, 1994.

129. Yagiura, M., Yamaguchi, T., Ibaraki, T., A Variable-Depth Search Algorithm with Branching Search for the Generalized Assignment Problem, Optimization Methods and Software, 10, 419-441, 1998.

130. Yagiura, M., Yamaguchi, T., Ibaraki, T., A Variable-Depth Search Algorithm for the Generalized Assignment Problem. In: Vob, S., Martello, S., Osman, I.H., Roucairol, C., (eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, 459-471, 1999.

131. Laguna, M., Kelly, J.P., Gonzalez-Velarde, J.L., Glover F., Tabu Search for the Multilevel Generalized Assignment Problem, European Journal of Operational Research, 82, 176-189, 1995.

132. Diaz, J.A., Fernandez, E., A Tabu Search Heuristic for the Generalized Assignment Problem, European Journal Operational Research, 132, 22-38, 2001.

133. Yagiura, M., Ibaraki, T., Glover, F., An Ejection Chain Approach for the Generalized Assignment Problem, Informs Journal of Computing, 16(2), 131-151, 2004.

134. Lourenço, H.R., Serra, D., Adaptive Search Heuristics for the Generalized Assignment Problem, Mathware and Soft Computing, 9, 209-234, 2002.

135. Alfandari, L., Plateau, A., Tolla, P., A Two-Phase Path Relinking Algorithm for the Generalized Assignment Problem, In Proceedings of the 4th Metaheuristic International Conference, Porto, Portugal, 175-179, 2001.

136. Alfandari, L., Plateau, A., Tolla, P., A Two-Phase Path Relinking Algorithm for the Generalized Assignment Problem, Technical Report No: 378, CEDRIC, CNAM, 2002.

137. Alfandari, L., Plateau, A., Tolla, P., A Path Relinking Algorithm for the Generalized Assignment Problem. In: Resende, M.G.C., Sousa, J.D. (eds.), Metaheuristics: Computer Decision-Making, Kluwer Academic Publishers, Boston, 1-17, 2004.

138. Yagiura, M., Ibaraki, T., Glover, F., An Effective Metaheuristic Algorithm for the Generalized Assignment Problem, In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Tucson, Arizona, USA, 2001.

139. Yagiura, M., Ibaraki, T., Glover, F., A Path Relinking Approach for the Generalized Assignment Problem, In Proceedings of International Symposium on Scheduling, Hamamatsu, Japan, 105-108, 2002.

140. Yagiura, M., Ibaraki, T., Glover, F., A Path Relinking Approach with Ejection Chains for the Generalized Assignment Problem, European Journal of Operational Research, 169, 548-569, 2006.

141. Randall, M., Heuristics for Ant Colony Optimisation Using the Generalised Assignment Problem, In Proceedings of IEEE Congress on Evolutionary Computation, Portland, Oregon, USA, 1916-1923, 2004.

142. Feltl, H., Raidl, G.R., An Improved Hybrid Genetic Algorithm for the Generalized Assignment Problem, In Proceedings of the Symposium on Applied Computing, Nicosia, Cyprus, 990-995, 2004.

143. Taşgetiren, M.F., Suganthan, P.N., Chua, T.J., Al-Hajri, A., Differential Evolution Algorithms for the Generalized Assignment Problem, In Proceedings of the 11th Congress on Evolutionary Computation, 2009.

144. Lorena, L.A.N., Narciso, M.G., Beasley, J.E., A Constructive Genetic Algorithm for the Generalized Assignment Problem, Submitted to Evolutionary Optimization, 2002.

145. Öztuna, D., Elhan, A.H., Gruplararası ve Grupiçi Karşılaştırma Yöntemleri, http://www.toraks.org.tr/mse-ppt-pdf/D_OZTUNA_H_ELHAN.pdf.

146. Wilcoxon, F., Individual Comparisons by Ranking Methods, Biometrics, 1, 80-83, 1945.

147. Karp, R.M., Reducibility Among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.), Complexity of Computer Computation, Plenum Press, New York, 85–103, 1972.

148. Kim, Y.K., Kim, Y.J., Kim, Y.H., Genetic Algorithms for Assembly Line Balancing with Various Objectives, Computers & Industrial Engineering, 30(3), 397–409, 1996.

149. Scholl, A.,Balancing and Sequencing of Assembly Lines, Heidelberg, Physica-Verlag, 1999.

150. Becker, C., Scholl, A., A Survey on Problems and Methods in Generalized Assembly Line Balancing, European Journal of Operational Research, 168, 694–715, 2006.

151. Baybars, I., A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem, Management Science, 32, 909–932, 1986a.

152. Salveson, M.E., The Assembly Line Balancing Problem, Journal of Industrial Engineering, 6, 18–25, 1955.

153. Jackson, J.R., A Computing Procedure for a Line Balancing Problem, Management Science, 2, 261–272, 1956.

154. Bowman, E.H., Assembly Line Balancing by Linear Programming, Operations Research, 8(3), 385–389, 1960.

155. Held, M., Karp, R.M., Shareshian, R., Assembly Line Balancing-Dynamic Programming with Precedence Constraints, Operations Research, 11, 442–459, 1963.

156. Dar-El, E.M., MALB-A Heuristic Technique for Balancing Large Single-Model Assembly Lines, AIIE Transactions, 5(4), 343–356, 1973.

157. Dar-El, E.M., Rubinovitch, Y., MUST-A Multiple Solutions Technique for Balancing Single Model Assembly Lines, Management Science, 25, 1105–1114, 1979.

158. Baybars, I., An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem, International Journal of Production Research, 24(1), 149–166, 1986b.

159. Suresh, G., Sahu, S., Stochastic Assembly Line Balancing Using Simulated Annealing, International Journal of Production Research, 32(8), 1801–1810, 1994.

160. Peterson, C., A Tabu Search Procedure for the Simple Assembly Line Balancing Problem, In Proceedings of the Decision Science Institute Conference, Washington DC, 1502-1504, 1993.

161. Lapierre, S.D., Ruiz, A., Soriano, P., Balancing Assembly Lines with Tabu Search, European Journal of Operational Research, 168, 826-837, 2006.

162. Falkenauer, E., Delchambre, A., A Genetic Algorithm for Bin Packing and Line Balancing, In Proceedings of IEEE International Conference on Robotics and Automation, Nice, France, 1189–1192, 1992.

163. Dimopoulos, C., Zalzala, A.M.S., Recent Developments in Evolutionary Computation for Manufacturing Optimisation: Problems, Solutions and Comparisons, IEEE Transactions on Evolutionary Computation, 4(2), 93–113, 2000.

164. Sabuncuoğlu, I., Erel, E., Tanyer, M., Assembly Line Balancing Using Genetic Algorithms, Journal of Intelligent Manufacturing, 11, 295-310, 2000.

165. Aytug, H., Khouja, M., Vergara, F.E., Use of Genetic Algorithms to Solve Production and Operations Management Problems: A Review, International Journal of Production Research, 41(17), 3955–4009, 2003.

166. Scholl, A., Becker, C., State-of-the-art Exact and Heuristic Solution Procedures for Simple Assembly Line Balancing, European Journal of Operational Research, 168, 666–693, 2006.

167. Ghosh, S., Gagnon, R.J., A Comprehensive Literature Review and Analysis of the Design, Balancing and Scheduling of Assembly Systems, International Journal of Production Research, 27(4), 637–670, 1989.

168. Erel, E., Sarin, S.C., A Survey of the Assembly Line Balancing Procedures, Production Planning and Control, 9, 414–434, 1998.

169. Boysen, N., Fliedner, M., Scholl, A., A Classification of Assembly Line Balancing Problems, European Journal of Operational Research, 183, 674-693, 2007.

170. Kim, Y.K., Kim, Y., Kim, Y.J., Two-Sided Assembly Line Balancing: A Genetic Algorithm Approach, Production Planning & Control, 11, 44-53, 2000.

171. Kim, Y.K., Kim, Y., Lee, T.O. (1999), Two-Sided Assembly Line Balancing Models, Technical Report, Chonnam National University, Kwangju, 1999.

172. Kim, Y.K., Kim, Y., Lee, T.O., Two-Sided Assembly Line Balancing Models, Working paper, Department of Industrial Engineering, Chonnam National University, Korea, 1998b.

173. Lee, T.O., Kim, Y., Kim, Y.K., Two-Sided Assembly Line Balancing to Maximize Work Relatedness and Slackness, Computers & Industrial Engineering, 40, 273-292, 2001.

174. Lapierre, S.D., Ruiz, A.B., Balancing Assembly Lines: An Industrial Case Study, Journal of Operational Research Society, 55(6), 589-597, 2004.

175. Simaria, A.S., Vilarinho, P.M., 2-ANTBAL: An Ant Colony Optimization Algorithm for Balancing Two-Sided Assembly Lines, In Proceedings of the 35th International Conference on Computers and Industrial Engineering, İstanbul, Türkiye, 2005.

176. Simaria, A.S., Vilarinho, P.M., 2-ANTBAL: An Ant Colony Optimization Algorithm for Balancing Two-Sided Assembly Lines, Computers & Industrial Engineering, 56(2), 489-506, 2009.

177. Hu, X., Wu, E., Jin, Y., A Station-Oriented Enumerative Algorithm for Two-Sided Assembly Line Balancing, European Journal of Operational Research, 186(1), 435-440, 2008.

178. Fleszar, K., Hindi, K.S., An Enumerative Heuristic and Reduction Methods for the Assembly Line Balancing Problem, European Journal of Operational Research, 145, 606–620, 2003.

179. Baykasoglu, A., Dereli, T., Two-Sided Assembly Line Balancing Using an Ant Colony Based Heuristics, International Journal of Advanced Manufacturing Technology, 36, 582-588, 2008.

180. Wu, E.F., Jin, Y., Bao, J.S., Hu, X.F., A Branch-and-Bound Algorithm for Two-Sided Assembly Line Balancing, International Journal of Advanced Manufacturing Technology, 39(9-10), 1009-1015, 2008.

181. Kim, Y.K., Song W.S., Kim J.H., A Mathematical Model and A Genetic Algorithm for Two-Sided Assembly Line Balancing, Computers & Operations Research, 36, 853-865, 2009.

182. Özcan, U., Toklu, B., A Tabu Search Algorithm for Two-Sided Assembly Line Balancing, International Journal of Advanced Manufacturing Technology, 43(7-8), 822, 2009a.

183. Özcan, U., Toklu, B., Multiple-Criteria Decision-Making in Two-Sided Assembly Line Balancing: A Goal Programming and a Fuzzy Goal Programming Models, Computers & Operations Research, 36, 1955-1965, 2009b.

184. Özcan, U., Toklu, B., Balancing of Mixed-Model Two-Sided Assembly Lines, Computers & Industrial Engineering, 57, 217-227, 2009c.

185. Leu, Y.Y., Matheson, L.A., Rees, L.P., Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria, Decision Sciences, 25, 581-606, 1994.

186. Baykasoğlu, A., Göçken, T., A Review and Classification of Fuzzy Mathematical Programs, Journal of Intelligent and Fuzzy Systems, 19(3), 205-229, 2008.

187. Zimmermann, H.J., Description and Optimization of Fuzzy Systems, International Journal of General Systems, 2, 209-215, 1976.

188. Bellman, R.E., Zadeh, L.A., Decision-Making in a Fuzzy Environment, Management Science, 17, 141-164, 1970.

189. Sinha, S., Fuzzy Mathematical Programming Applied to Multi-Level Programming Problems, Computers and Operations Research, 30, 1259-1268, 2003.

190. Chakraborty, M., Gupta, S., Fuzzy Mathematical Programming for Multi Objective Linear Fractional Programming Problem, Fuzzy Sets and Systems, 125, 335-342, 2002.

191. Chanas, S., The Use of Parametric Programming in Fuzzy Linear Programming, Fuzzy Sets and Systems, 11, 243-251, 1983.

192. Gen, M., Ida, K., Lee, J., Kim, J., Fuzzy Nonlinear Goal Programming Using Genetic Algorithm, Computers and Industrial Engineering, 33, 39-42, 1997.

193. Baykasoğlu, A., Göçken, T., A Tabu Search Approach to Fuzzy Goal Programs and An Application to Aggregate Production Planning, Engineering Optimization, 38(2), 155-177, 2006.

194. Narasimhan, R., Goal Programming in A Fuzzy Environment, Decision Sciences, 11, 325-336, 1980.

195. Hannan, E.L., On Fuzzy Goal Programming, Decision Sciences, 12, 522-531, 1981.

196. Lu, J., Wu, F., Zhang, G., On A Generalized Fuzzy Goal Optimization for Solving Fuzzy Multi-Objective Linear Programming Problems, Journal of Intelligent and Fuzzy Systems, 18(1), 83-97, 2007.

197. Tiwari, R.N., Dharmar, S., Rao, J.R., Fuzzy Goal Programming – An Additive Model, Fuzzy Sets and Systems, 24, 27-34, 1987.

198. Özbakır, L., Baykasoğlu, A., Tapkan, P., Bees Algorithm for Generalized Assignment Problem, Applied Mathematics and Computation, 215, 3782-3795, 2010.

199. Özbakır, L., Tapkan, P., Bees Algorithm for Zone Constrained Two-Sided Assembly Line Balancing Problem, Expert Systems with Applications, (İnceleme aşamasında).

200. Baykasoğlu, B., Özbakır, L., Tapkan, P., Balancing Fuzzy Multiple Objective Two-Sided Assembly Lines via Bees Algorithm, Journal of Intelligent and Fuzzy Systems, (Basım aşamasında).

201. Tapkan, P., Özbakır, L., Baykasoğlu, A., Arı Algoritması ve Genelleştirilmiş Atama Problemi: Farklı Komşuluk Yapılarının Karşılaşıtırılması, Endüstri Mühendisliği Dergisi, (Basım aşamasında).

202. Baykasoğlu, A., Özbakır, L., Tapkan, P., Artificial Bee Colony Algorithm and its Application to Generalized Assignment Problem, In: Chan, F.T.S., Tiwari, M.K. (eds.), Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, I-Tech Education and Publishing, Vienna, Austria, 113-144, 2007.

203. Tapkan, P., Özbakır, L., Baykasoğlu, A., Bees Algorithm for Two-Sided Assembly Line Balancing Problem, In Proceedings of 23. European Conference on Operational Research, Bonn, Germany, 2009.

204. Baykasoglu, A., Ozbakir, L., Tapkan, P., Özaslan, M., Genelleştirilmiş Atama Problemi için Arı Kolonisi Optimizasyonu, Yöneylem Araştırması / Endüstri Mühendisliği 27. Ulusal Kongresi, İzmir, 2007.

205. Tapkan, P., Özbakır, L., Baykasoğlu, A., Arı Algoritması ve Genelleştirilmiş Atama Problemi: Farklı Komşuluk Yapılarının Karşılaştırılması, Yöneylem Araştırması / Endüstri Mühendisliği 28. Ulusal Kongresi, İstanbul, 2008.

206. Tapkan, P., Özbakır, L., Baykasoğlu, A., Çift Taraflı Montaj Hattı Dengeleme Problemi için Arı Algoritması, Yöneylem Araştırması / Endüstri Mühendisliği 29. Ulusal Kongresi, Ankara, 2009.

207. Tapkan, P., Özbakır, L., Baykasoğlu, A., Arı Algoritması ile Bölgesel – Konumsal Kısıtlı Bulanık Çok-Kriterli Çift-Taraflı Montaj Hattı Dengelenmesi, Yöneylem Araştırması / Endüstri Mühendisliği 30. Ulusal Kongresi, İstanbul, 2010.

ÖZGEÇMİŞ

Pınar Zarif TAPKAN 1979 yılında Kayseri'de doğdu. İlk, orta ve lise öğrenimini Kayseri'de tamamladı. 1996 yılında Erciyes Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği bölümünü kazandı ve 2000 yılında mezun oldu. 2002 yılında Bilkent Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalında yüksek lisans eğitimini tamamladı. 2004 yılında Erciyes Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalında doktora eğitimine başladı. 2001 yılı Aralık ayında Erciyes Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü'ne araştırma görevlisi olarak atandı ve halen aynı bölümde görevine devam etmektedir.

İletişim Bilgileri:
Erciyes Üniversitesi, Mühendislik Fakültesi,
Endüstri Mühendisliği Bölümü,
38039 KAYSERİ
Tel: (0 352) 4374901 / 32477
e-mail: pinartan@erciyes.edu.tr.