SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SELÇUK
ÜNİVERSİTESİ

T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ALG ALGORİTMASI TABANLI DAĞITIK İŞ ATÖLYESİ ÇİZELGELEME PROBLEMLERİNİN OPTİMİZASYONU

Okan UZUNOĞLU

YÜKSEK LİSANS TEZİ

Elektrik Elektronik Mühendisliği Anabilim Dalı

Ağustos-2025
KONYA

TEZ KABUL VE ONAYI

Okan UZUNOĞLU tarafından hazırlanan “Yapay Alg Algoritması Tabanlı Dağıtık İş Atölyesi Çizelgeleme Problemlerinin Optimizasyonu” adlı tez çalışması 11/08/2025 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Elektrik Elektronik Mühendisliği Anabilim Dalı’nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri **İmza**

Başkan
Prof. Dr. Mehmet ÇUNKAŞ

Danışman
Doç. Dr. Mehmet Akif ŞAHMAN

Üye
Dr. Öğr. Üyesi Sedat KORKMAZ

Yukarıdaki sonucu onaylarım.

Prof. Dr. Hasan AYDOĞAN
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza

Okan UZUNOĞLU

Tarih: 11.08.2025

ÖZET

YÜKSEK LİSANS TEZİ

YAPAY ALG ALGORİTMASI TABANLI DAĞITIK İŞ ATÖLYESİ ÇİZELGELEME PROBLEMLERİNİN OPTİMİZASYONU

Okan UZUNOĞLU

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Mehmet Akif ŞAHMAN

2025, 65 Sayfa

Jüri
Doç. Dr. Mehmet Akif ŞAHMAN
Prof.Dr. Mehmet ÇUNKAŞ
Dr.Öğr.Üyesi Sedat KORKMAZ

DJSSP, Np-Zor olan geleneksel İş Atölyesi Çizelgeleme Probleminin (JSSP) bir uzantısıdır ve daha karmaşık bir optimizasyon problemidir. JSSP, aynı olmayan iş parçalarının farklı makinelerde değişken işlem sürelerine sahiptir ve belirli bir sıraya göre işlenmeleri gereken bir problemi ifade eder. Temel hedefi, iş parçalarını, makineleri ve zaman parametrelerini kullanarak toplam işlem süresini minimize etme esasına göre düzenlemektir. DJSSP ise birden fazla tesiste bulunan ve bu yerlerdeki makinelerin işlerle olan koordinasyonunu içeren daha karmaşık bir optimizasyon problemidir ve zaman çizelgesini de tesis, makine, iş, süre parametrelerini kullanarak bir hedefe ulaşmayı sağlar. NP-zor problemler olarak sınıflandırılan JSSP ve DJSSP problemleri kesin metotlar kullanılarak makul bir sürede çözülemezler. NP-zor problemleri çözmek için makul bir sürede kabul edilebilir çözümler sağlayan meta-sezgisel yöntemler kullanılır.

Bu tez çalışmasında, literatürde yer alan ve iyi bilinen 10 farklı metasezgisel algoritma (Parçacık Sürü Optimizasyonu Algoritması-PSO, Yapay Arı Kolonisi Algoritması-ABC, Gri Kurt Optimize Edici–GWO, Kızıl Tilki Optimize Edici– RFO, Jaya Algoritması-JAYA, Yapay Alg Optimizasyon Algoritması-AAA, Ağaç-Tohum Algoritması-TSA, Lévy Uçuş Dağıtımı Algoritması-LFD, Diferansiyel Arama Algoritması-DSA, Balina Optimizasyon Algoritması-WOA) DJSSP problemlerini çözmek için kullanılmıştır. Ayrıca sürekli arama uzayında çalışan metasezgisel algoritmaların DJSSP ayrık problemi için çalıştırılabilmesi için ayrıklaştırma amacıyla kullanılan 3 farklı kodlama şeması (Rastgele Anahtar Kodlama Şeması-RK, En Küçük Pozisyon Değeri Kodlama Şeması-SPV, Sıralanmış Değer Kodlama Şeması-ROV) ele alınmıştır. Her bir kodlama şeması 10 metasezgisel algoritma ile 48 adet DJSSP karşılaştırma problemi üzerinde uygulanmıştır. Elde edilen sonuçlara göre AAA algoritması diğer metasezgisel algoritmalardan daha kaliteli çözümler elde ettiği görülmüştür.

Anahtar Kelimeler: Dağıtık İş Atölyesi Çizelgeleme Problemi (DJSSP), İş Atölyesi Çizelgeleme Problemi (JSSP), Kodlama Şeması, Metasezgisel Algoritmalar, Optimizasyon, Üretim Atölyeleri

ABSTRACT

MS THESIS

OPTIMIZATION OF DISTRIBUTED JOB SHOP SCHEDULING PROBLEMS BASED ON ARTIFICIAL ALGAE ALGORITHM

Okan UZUNOGLU

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF MASTER OF SCIENCE
IN ELECTRICAL ELECTRONICS ENGINEERING

Advisor: Assoc. Prof. Dr. Mehmet Akif ŞAHMAN

2025, 65 Pages

Jury
Assoc. Prof. Dr. Mehmet Akif ŞAHMAN
Prof.Dr. Mehmet ÇUNKAŞ
Assist.Prof.Dr. Sedat KORKMAZ

DJSSP is an extension of the traditional Job Shop Scheduling Problem (JSSP), which is Np-Hard and is a more complex optimization problem. JSSP refers to a problem where non-identical workpieces have variable processing times on different machines and must be processed in a certain order. Its main objective is to arrange the workpieces on the basis of minimizing the total processing time using machines and time parameters. DJSSP, on the other hand, is a more complex optimization problem involving multiple facilities and the coordination of the machines in these locations with the jobs and the timetable to achieve an objective using the facility, machine, job and time parameters. JSSP and DJSSP problems are classified as NP-hard problems and cannot be solved in a reasonable time using exact methods. To solve NP-hard problems, meta-heuristics are used that provide acceptable solutions in a reasonable time.

In this thesis, 10 different well-known metaheuristic algorithms (Particle Swarm Optimization-PSO, Artificial Bee Colony-ABC, Grey Wolf Optimizer – GWO, Red Fox Optimizer – RFO, Jaya Algorithm-JAYA, Artificial Algae Algorithm-AAA, Tree-Seed Algorithm-TSA, Lévy Flight Distribution-LFD, Differential Search Algorithm - DSA, Whale Optimization Algorithm- WOA) from the literature are used to solve DJSSP problems. In addition, 3 different coding schemes (Random Key Encoding Scheme-RK, Smallest Position Value Encoding Scheme-SPV, Ranked-Over Value Encoding Scheme-ROV) used for discretization are discussed in order to run metaheuristic algorithms operating in the continuous search space for the DJSSP discrete problem. Each coding scheme is applied on 48 DJSSP benchmark problems with 10 metaheuristic algorithms. According to the results obtained, the AAA algorithm obtained better quality solutions than the other metaheuristic algorithms.

Keywords: Distributed Job Shop Scheduling Problem (DJSSP), Encoding Scheme, Job Shop Scheduling Problem (JSSP), Metaheuristic Algorithms, Manufacturing Job Shops, Optimization

ÖNSÖZ

Bu çalışmanın gerçekleşmesinde, bilgilerini benimle paylaşan ve her zaman bana doğru yolu gösteren, hep yanımda olan değerli danışmanım Doç. Dr. Mehmet Akif ŞAHMAN'a, bana olan destekleriyle kıymetli annem Aygül UZUNOĞLU'na, kıymetli babam Ayhan UZUNOĞLU'na ve sevgili eşim Damla UZUNOĞLU'na, her konuda beni destekleyen mesai arkadaşlarıma teşekkürlerimi borç bilirim.

Okan UZUNOĞLU
KONYA-2025

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

Kısaltmalar

AAA	: Yapay Alg Optimizasyon Algoritması-Artificial Algae Algorithm
ABC	: Yapay Arı Kolonisi Algoritması- Artificial Bee Colony Algorithm
CP	: Kısıtlama Programlama- Constraint Programming
CRO	: Kimyasal Reaksiyon Optimizasyon Algoritması- Chemical Reaction Optimization Algorithm
DE-SA	: Diferansiyel Evrim Simüle Edilmiş Tavlama- Differential Evolution Simulated Annealing
DFJSPC	: Vinç taşımacılığında Dağıtık Esnek İş Atölyesi Planlama Problemi- Distributed Flexible Job Shop Scheduling Problem With Crane Transportations
DFJSSP	: Dağıtık Esnek İş Atölyesi Çizelgeleme Problemi- Distributed Flexible Job Shop Scheduling Problem
DFJSSPT	: Transfer Edilmiş Dağıtılmış Esnek İş Atölyesi Çizelgeleme Problemi- Distributed Flexible Job Shop Scheduling Problem Transfered
DJSSP	: Dağıtık İş Atölyesi Çizelgeleme Problemi- Distributed Job Shop Scheduling Problem
DNN	: Derin Sinir Ağı- Deep Neural Network
DSA	: Diferansiyel Arama Algoritması- Differential Search Algorithm
DSHO	: Benekli Sırtlan Optimize Edicisinin- Discrete Version of The Spotted Hyena Optimizer
EDA	: Dağıtım algoritmasının tahmini- Estimation of Distribution Algorithm
EMA	: Verimli Memetik Algoritma- Efficient Memetic Algorithm
FMU	: Esnek Üretim Birimi- Flexible Manufacturing Unit
GA-GC	: Genetik Algoritma- Gantt Şeması- Genetic Algorithm- Gantt Chart
GATA	: Geliştirilmiş Ağaç Tohum Algoritması
GWO	: Gri Kurt Optimize Edici- Grey Wolf Optimizer
IGA	: Geliştirilmiş Genetik Algoritma- Improved Genetic Algorithm
JSPLIB	: JSP Kütüphanesi- JSP Library
JSSP	: İş Atölyesi Çizelgeleme Problemi- Job Shop Scheduling Problem
LFD	: Lévy Uçuş Dağıtımı- Lévy Flight Distribution

MGA	: Değiştirilmiş Genetik Algoritma- Modified Genetic Algorithm
MILP	: Karışık Tam Sayılı Doğrusal Programlama- Mixed Integer Linear Programming
PSO	: Parçacık Sürü Optimizasyon Algoritması- Particle Swarm Optimization Algorithm
QA	: Kuantum Tavlama- Quantum Annealing
QPU	: Kuantum İşleme Birimi- Quantum Processing Unit
QUBO	: Kuadratik Kısıtlanmamış İkili Optimizasyon- Quadratic Unconstrained Binary Optimization
RFO	: Kırmızı Tilki Optimize Edici- Red Fox Optimizer
RK	: Rastgele Anahtarlama- Random Key
ROV	: Sıralamada Üst Değer- Ranked-Over Value
SA	: Simüle Edilmiş Tavlama- Simulated Annealing
SLIM	: Kendi Kendini Etiketleme İyileştirme Yöntemi- Self-Labeling Improvement Method
SPV	: En Küçük Pozisyon Değeri- Smallest Position Value
TS	: Tabu Araması- Tabu Search
TSA	: Ağaç-Tohum Algoritması- Tree-Seed Algorithm
VNS	: Değişken Komşu Araması- Variable Neighborhood Search
WOA	: Balina Optimizasyon Algoritması- Whale Optimization Algorithm

1. GİRİŞ

Günümüzde teknolojinin gelişmesiyle ve küreselleşmenin de etkisiyle endüstri alanında yaşanılan gelişmeler hızlıca artmaktadır. Şirketlerin de bu gelişmeler karşısında paralel bir şekilde ilerlemesi ve geri kalmaması için planlamalarını da ona göre güncellemesi gerekmektedir. Bu sebeple şirketlerin üretim tesislerinde, makine ve iş parçalarının zamanında, düzgün sırayla ve verimli bir şekilde çalışmasının planlanması ve düzenlenmesi kritik öneme sahiptir. Bu planlama, üretim verimliliğini artırmak, iş süreçlerini minimuma indirmek ve maliyetleri azaltmak için gereklidir. İş Atölyesi Çizelgeleme Problemi (JSSP) ve Dağıtık İş Atölyesi Çizelgeleme Problemi (DJSSP), bu alanda karşılaşılan NP-Hard problemlerden biridir. JSSP'de her iş önceden belirlenen işlem sırasına göre hareket eder. Bu işlem sıralamasındaki her bir iş, işleneceği bir makineye ve sabit bir işleme süresine sahiptir. Bu işlemlerin sonunda üretilen her ürün, sabit makinelerde değişmeyen süreyle geçirdiği operasyon sonucu üretilir. Bu operasyonu minimum süre ile tamamlanması, aynı birim zaman içerisinde daha çok ürün çıkarılmasına ve genel olarak bakıldığında birim maliyetinin düşürülmesine olanak sağlar. Küreselleşmenin yaygın olduğu böyle bir dönemde, şirketlerin üretim modeli tekil bir tesis yerleşkesinden çoklu yerleşkelere dönüşmüştür. Dağıtılmış iş atölyesi planlama problemi, bu dağınık yerleşkelerdeki tesislerdeki operasyon sürecini planlamaya odaklanan NP-Hard problemlerden biridir. DJSSP'nin iki ana zorluğu vardır. Birincisi uygun yerleşkedeki bir tesisi seçmek, ikincisi ise üretim süresini en aza indirmek için en iyi operasyon dizisini belirlemektir.

Optimizasyon sürecini ele alan metasezgisel algoritmalar, daha çok doğadan ilham alınarak oluşturulan popülasyona dayalı algoritmalardır ve en iyiye ulaşmak için arama uzayını keşfeden metotlar kullanılır. Herhangi bir probleme özgü değillerdir ve birçok alanda uygulanabilir yapıdadırlar. Bu tez çalışmasında metasezgisel algoritmalar endüstri alanının önemli bir parçası olan JSSP ve DJSSP'de kullanılmıştır, üretim sürecinin kısaltılmasında ve düşük maliyetle sonuçlanmasında etkili olmuşlardır.

Bu tez çalışmasında, öncelikle üretim alanında önemli bir yere sahip olan DJSSP'ler tanıtılmıştır. DJSSP'lerin çözümünde sıklıkla kullanılan metasezgisel algoritmalar hakkında bilgi verilmiştir. Ayrıca DJSSP'lerin optimizasyon sonucunda daha verimli bir çalışma planı ile üretim tesislerinin etkinliğinin artırmak amacıyla uygulanan kodlama şemaları bu tez çalışmasında ele alınmıştır. Bu alanda yapılan önceki çalışmalar incelenmiştir ve çözüm kalitesinin arttırılması için farklı optimizasyon algoritmaları uygulanmıştır. Tez çalışmasında temel metasezgisel algoritmalar (PSO, ABC, GWO, RFO, JAYA, LFD, WOA, AAA, DSA,

TSA) ve farklı kodlama şemaları (RK, SPV, ROV) kullanılmıştır. Elde edilen deneysel sonuçlar karşılaştırılmış ve üzerine değerlendirmeler yapılmıştır.

Tez çalışmasının ikinci bölümünde literatür araştırması sunulmuştur. Üçüncü bölümde problemler detaylı bir şekilde anlatılmıştır. Dördüncü bölümde materyal olarak kullanılan karşılaştırma problemleri ve özellikleri sunulmuştur. Beşinci bölümde metot olarak kullanılan metasezgisel optimizasyon algoritmaları, tesis iş gücü dağıtımı ve kodlama şemalarından bahsedilmiştir. Altıncı bölümde yapılan deneysel sonuçlar ve karşılaştırmalar verilmiştir. Yedinci bölümde yapılan deneysel sonuçlar yorumlanmıştır ve gelecekte gerçekleştirilebilecek çalışmalara dair öneriler yapılmıştır.

2. KAYNAK ARAŞTIRMASI

Planlama problemi mevcut kaynakların görevlere uygun şekilde tahsis edilmesidir. JSSP, çizelgeleme probleminin en önemli ve karmaşık türlerinden biridir. Klasik JSSP'de görevleri temsil eden bir dizi iş ve kaynakları temsil eden bir dizi makine vardır(Chipperfield ve Fleming, 1997). JSSP'de ortak varsayım, her işin aynı tesiste işlenecek bir dizi makineye sahip olduğudur. Öte yandan tüm üretimin aynı tesiste yapılması gerçek hayattaki durumu yansıtmamaktadır. Gerçek hayatta ekonomik nedenlerden ve küreselleşmeden dolayı geleneksel tek tesisli üretimin çok tesisli üretime geçmesi, şirketleri diğerleriyle rekabet halinde olmaya zorlamaktadır. Bu nedenle araştırmacılar çoğunlukla JSSP'nin çok daha karmaşık bir türü olan ve gereksiz zaman tüketimini önleyerek maliyetleri azaltmak için işleri farklı tesislere atayan DJSSP üzerinde çalışmışlardır (Naderi ve Azab, 2015). JSSP'den farklı olarak DJSSP'de iki ana kararın alınması gerekir. İlk karar, işlerin uygun tesise atanması, ikinci karar ise ilgili tesislerdeki iş dizilerinin optimize edilmesidir (Naderi ve Azab, 2014). Yıllardır DJSSP birçok araştırmacı tarafından incelenmiştir ve literatür araştırması sonuçları aşağıda belirtilmiştir.

Yu Du ve arkadaşları (Du ve ark., 2021) , çalışmasında her fabrikada işlerin makineler arasında taşınması için bir vincin bulunduğu DFJSP'de vinç taşımacılığına yer vermiştir. Her vincin, iş pozisyonu, aynı işin önceki pozisyonu ve vinç pozisyonu arasındaki ilişkilere bağlı olarak dört farklı koşulu vardır. Bu çalışmada, DFJSP'yi vinçle taşıma (DFJSPC) ile çözmek için hibrit bir algoritma önermiştir. Bu çalışmada, DFJSPC'yi çözmek için dağıtım algoritması (EDA) ve değişken komşuluk aramasından (VNS) oluşan hibrit bir algoritma önermiştir. EDA ve VNS bileşenlerini değiştirerek ve DFJSPC'ye uygulanabilecek yeni işlemler ekleyerek hibrit EDA-VNS algoritmasını önermiştir. EDA bileşeni, bir yinelemenin hesaplama süresini azaltmak için bir olasılık değişkeni tarafından kontrol edilir ve parametreler, erken yakınsamayı önlemek için kendi kendine uyarlanacak şekilde ayarlanır. Popülasyon, keşif yeteneği nedeniyle EDA bileşeni tarafından geliştirilir ve kullanım yeteneği nedeniyle VNS uygulanır. VNS bileşeninde, optimize edilmiş çözümler elde etmek için küresel ve yerel stratejileri içeren beş komşuluk yapısı düzenlenmiştir. Önerilen algoritmanın performansını, Luo ve ark. önerdiği EMA'yı (Luo ve ark., 2020) , Li ve ark. önerdiği TS (Luo ve ark., 2020), Wu ve diğerleri tarafından önerilen DE-SA (Li ve ark., 2018) örnekleriyle karşılaştırılmıştır ve ayrıntılı deneysel analiz, önerilen EDA-VNS'nin güvenilirliğini ve sağlamlığını doğrulamıştır.

Wu ve arkadaşları (Wu ve ark., 2019), yaptığı çalışmada işlerin fabrikalar arasında transfer edilebildiği, transferlerle dağıtılmış esnek iş atölyesi çizelgeleme problemi (DFJSPT) önermiştir. Evrimsel algoritmaları yerel aramayla melezleştiren memetik algoritmalar, çok

amaçlı FJSP'lerin çözümünde umut verici bir performans sağlamıştır. İşlerin transferlerle birden fazla fabrikada işlenebildiği, dağıtılmış esnek iş atölyesi çizelgeleme probleminin (DFJSP), yani transferlerle dağıtılmış esnek iş atölyesi çizelgeleme probleminin (DFJSPT) genişletilmiş bir modelini önermiştir. DFJSPT'nin matematiksel bir modeli, üretim süresi, fabrikaların maksimum iş yükü ve toplam enerji tüketiminin eş zamanlı optimizasyon hedefleriyle oluşturulmuştur. Bu modeli çözmek için etkili memetik algoritma geliştirildi. EMA'nın etkinliği çeşitli deneylerle kanıtlanmıştır.

L. De Giovanni ve F. Pezzella (De Giovanni ve Pezzella, 2010), bu makalede sunulan Geliştirilmiş Genetik Algoritma (IGA), üç karar seviyesini aynı anda hesaba katabilmekte ve Esnek Üretim Birimi (FMU) sistemlerinin küresel üretim süresini en aza indirmeyi amaçlayan bir üretim planlaması sağlamıştır. IGA, Modifiye edilmiş Genetik Algoritma (MGA)'nın aynı basit kromozom kodlamasından başlar ve alternatif makine yolları arasında seçim yapabilen özel bir kod çözme şeması sayesinde bunu esnek duruma kadar genişletir. IGA ayrıca, gen değişimine dayalı bir mahalle araması yoluyla her neslin en iyi bireylerini yerel olarak geliştirmek için yeni bir mutasyon operatöründen yararlanır. IGA, dağıtılmış planlama sorunları için tasarlanan önceki algoritmayla karşılaştırılmış ve iyi bilinen planlama kriterlerinden türetilen geniş bir DFJS örnekleri kümesi üzerinde test edilmiş, tatmin edici sonuçlar sunmuş ve yeni kod çözme şemasının ve iyileştirme operatörünün etkinliğini kanıtlamıştır.

Meng ve arkadaşları (Meng ve ark., 2020), bu çalışmada, DFJSP'yi çözmek için dört farklı modelleme fikrine dayalı dört Karışık Tam Sayılı Doğrusal Programlama (MILP) modeli formüle etmiştir. DFJSP'nin NP-Hard özelliği nedeniyle, büyük boyutlu örnekleri daha etkili bir şekilde çözmek için bir Kısıtlama Programlama (CP) modeli önermiştir. Tüm MILP modellerini ve CP modelini mevcut algoritmalarla karşılaştırdılar ve deneysel sonuçlar, dizi bazlı modelin DFJSP çözümünde en etkili model olduğunu göstermektedir. Karşılaştırma sonuçlarından da CP modeli hem çözüm kalitesi hem de verimlilik açısından mevcut tüm algoritmalardan daha iyi performans gösterdiği anlaşılmaktadır.

Ziaee (Ziaee, 2014), DFJSP'yi çözmek için hızlı yapıcı bir buluşsal algoritma önerdi. İyi kalitede programları çok hızlı bir şekilde elde etmek için yapıcı bir prosedüre dayalı hızlı bir buluşsal algoritma geliştirmiştir. Algoritma, performansını değerlendirmek için literatürdeki kıyaslama örnekleri üzerinde test edilmiştir. Hesaplamalı sonuçlar, basitliğine rağmen önerilen buluşsal yöntemin hesaplama açısından verimli olduğunu ve pratik problemler için umut verici olduğunu göstermektedir.

Wu, Guo, Li ve Wang (Li ve ark., 2020), DFJS probleminin 3 boyutlu olduğunu bu boyutların, işten hücreye atama, operasyon sıralaması, operasyondan makineye atama olduğunu

söylemiştir. Tek boyuttan 3 boyutlu kod çözme yöntemi, yük dengeli olma eğiliminde olan iyi bir 3D çözüm elde etmek için tasarlanmıştır. Sayısal deneyler, De Giovanni ve Pezzella (De Giovanni ve Pezzella, 2010) tarafından geliştirilen IGA'nın DFJS problemlerinin çözümünde güncel en iyi performansı gösteren genetik algoritma olduğunu göstermektedir.

Gonçalvez (Gonçalves ve ark., 2005), bu makalede, iş yeri planlama problemi için bir hibrit genetik algoritma sunmuştur. Problemin kromozom gösterimi rastgele anahtarlara dayanmaktadır. Planlamalar, önceliklerin genetik algoritma tarafından tanımlandığı bir öncelik kuralı kullanılarak oluşturulur. Planlamalar, parametreli etkin planlamalar üreten bir prosedür kullanılarak oluşturulur. Bir planlama elde edildikten sonra çözümü iyileştirmek için yerel bir arama buluşsal yöntemi uygulanmıştır. Yaklaşım, literatürden alınan bir dizi standart örnek üzerinde test edilmiş ve diğer yaklaşımlarla karşılaştırılmıştır. Hesaplama sonuçları önerilen algoritmanın etkinliğini doğrular niteliktedir.

Diarmiud Grimes ve Emmanuel Hebrard (Grimes ve Hebrard, 2010), sundukları kısıtlama modellerinin iş yeri planlama probleminin iki çeşidini ele almak için nasıl kolayca genişletilebileceğini göstermiştir. Her iki durumda da yaklaşımlarının en son teknolojiyle rekabet edebilir olduğunu özellikle de her iki problem türünün bazı açık problemlerinde optimumu kanıtlamada uygulanabilir olduğunu gördüler. Standart iş yeri ve açık iş yeri planlama problemleri için arama verimliliğini tekdüze bir şekilde iyileştirdiği görülürken, kısıtlama ağırlıklandırma analizlerinin aslında bu problemlerin bazı çeşitleri için zararlı olabileceğini ortaya koymuşlardır.

James Kotary, Ferdinando Fioretto ve Pascal Van Hentenryck (Kotary ve ark., 2022) tarafından yazılan bu makaleye üretim zincirlerindeki artan stokastik ilham vermiştir. JSP'ye verimli ve doğru yaklaşımlar sunmak için derin öğrenme yaklaşımını araştırmışlar ve problem yapısını kullanmak için derin bir sinir ağı mimarisinin tasarımını, problem kısıtlamalarını yakalamak için Lagrangian ikiliğiyle entegrasyonunu ve çözümün uygulanabilirliğini garantilemek için bir işlem sonrası optimizasyonu önermişlerdir. JSP-DNN adı verilen ortaya çıkan yöntem, JSPLIB kıyaslama kütüphanesinden alınan JSP örnekleri üzerinde değerlendirilir. Hesaplama sonuçları, JSP-DNN'nin ihmal edilebilir hesaplama maliyetleriyle yüksek kalitede JSP yaklaşımları üretebileceğini göstermişlerdir.

Andrea Corsini, Angelo Porrello, Simone Calderara, Mauro Dell'Amico (Corsini ve ark., 2024), Bu çalışmada, kombinatoryal problemler için tasarlanmış kendi kendini denetleyen bir eğitim stratejisi önermişlerdir. Bu Kendi Kendini Etiketleme İyileştirme Yöntemini (SLIM), sinirsel kombinatoryal topluluk tarafından çok ilgi gören karmaşık bir kombinatoryal problem olan İş Atölyesi Planlaması üzerinde doğrulamışlardır. İyi bilinen işaretçi ağına dayalı üretken

bir model önermişler ve bunu SLIM ile eğitmişlerdir. Popüler kıyaslama noktaları üzerindeki deneyler, ortaya çıkan modellerin JSP için yapıcı sezgisel yöntemlerden ve son teknoloji öğrenme önerilerinden daha iyi performans göstermesiyle bu yaklaşımın potansiyelini göstermektedir. Son olarak, SLIM'in çeşitli parametrelere karşı sağlamlığını ve genelliğini Gezgin Satıcı Problemi'ne uygulayarak kanıtlamışlardır.

Lilia Toma, Markus Zajac, Uta Störl (Toma ve ark., 2024), Birçok modern üretim şirketi, hem coğrafi olarak dağılmış üretim emirlerini hem de çok tesisli üretim adımlarını ele alması gereken tek bir üretim tesisinden çok fabrikalı bir üretim ortamına evrilmiştir. Aynı işlemi gerçekleştirebilen farklı lokasyonlardaki bir dizi makinenin ve fabrikalar arasındaki nakliye sürelerinin mevcudiyeti, planlama sistemlerini klasik İş Yeri Planlama Probleminden (JSSP) Dağıtılmış Esnek İş Yeri Planlama Problemine (DFJSP) dönüştürmüştür. Sonuç olarak, üretim planlamasının karmaşıklığı önemli ölçüde artmıştır. Çalışmada, DFJSP'yi çözmek için Kuantum Tavlama (QA) kullandılar. Üretim emirlerinin üretim tesislerine atanmasına ek olarak, üretim adımlarının üretim tesislerine atanması da gerçekleşir. Bu gereklilik, bir yünlü tekstil üreticisinin gerçek bir kullanım durumuna dayanmaktadır. Bu yöntemin büyük problem örneklerine uygulanabilirliğini araştırmak için, 50 değişkenden 250 değişkene kadar değişen problemler, bir D-Wave kuantum tavlayıcı Kuantum İşleme Birimi'ne (QPU) yerleştirilebilecek en büyük problem formüle edilmiş ve çözülmüştür. Kuadratik Kısıtlanmamış İkili Optimizasyon (QUBO) modelinin Lagrange parametrelerinin ve QPU yapılandırma parametrelerinin belirlenmesine özel dikkat gösterilmiştir, çünkü bu faktörler çözüm kalitesini önemli ölçüde etkilemektedir. Elde edilen çözümler hem çözüm kalitesi hem de hesaplama süresi açısından Simüle Edilmiş Tavlama (SA) ile elde edilen çözümlerle karşılaştırılmıştır. Sonuçlar, QA'nın endüstriye özgü büyük problem örneklerini çözme potansiyeline sahip olduğunu göstermektedir.

Chaouch ve ark. (Chaouch ve ark., 2017) (Chaouch ve ark., 2020), 2017 ve 2020'de tek amaçlı DJSP'yi çözen optimizasyon tekniklerini araştırmıştır. Jia ve diğerleri (Jia ve ark., 2002), DJSP'nin çözümü için ilk çalışmayı 2002'de önerdi. DJSP'nin çözümü için değiştirilmiş bir genetik algoritma kullanıldı(Jia ve ark., 2003). Bu çalışmada iki aşamalı bir kodlama şeması önerilmiştir. İlk adımda tesisler; ikinci adımda işler ve operasyonlar dikkate alınır. Jia ve diğerleri, Gantt şemasını (GC) ve GA–GC olarak adlandırılan GA'yı melezleştirdi (Jia ve ark., 2007).

Marzouki, Driss ve Ghédira (Marzouki ve ark., 2018), Kimyasal reaksiyon optimizasyonu (CRO) algoritmasını, Wu ve arkadaşları (Li ve ark., 2020), yapay arı kolonisi algoritmasını (ABC), Huang ve Yao (Huang ve Yao, 2012) parçacık sürüsü optimizasyonu

(PSO) algoritması gibi diğer metasezgisel algoritmalarla DFJSP'yi çözmeye çalışmıştır. Cheng ve arkadaşları (Cheng ve ark., 1996), Bölüm I anketlerinde klasik İşyeri planlama problemlerini (JSP) çözmek için GA'ları kullanan makaleler hakkında ayrıntılı bir eğitim anketi sundular. Aynı ekip (Cheng ve ark., 1999) bölüm II'de, JSP'yi ele almak için hibrit GA kullanan makaleleri incelediler.

M. A. ŞAHMAN ve S. KORKMAZ(Şahman ve Korkmaz, 2022), bu çalışmasında ayrık optimizasyon problemlerini çözmek için Yapay Alg Algoritmasını önermiştir. JSSP probleminin çözümünde ilk defa üç kodlama şeması kullanılmıştır. Bu 3 kodlama şeması AAA algoritmasıyla entegre edilmiştir, karşılaştırmalı ve detaylı analiz sonucunda, üretim süresi değeri açısından en iyi sonuçların SPV kodlama şemasının AAA yöntemine entegre edilmesiyle elde edildiği görülmüştür.

M. A. ŞAHMAN(Şahman, 2022), yaptığı bu çalışmada büyük boyutlu JSP problemlerinin çözümünde denemeler yaparken Geliştirilmiş Ağaç Tohum Algoritmasını (GATA) yaklaşımının başarısını göstermiştir. Çalışmada kullanılan tüm metasezgisel algoritmalar eşit koşullar altında uygulanmıştır. Sonuçlar incelendiğinde ise GATA, hem diğer algoritmalara göre hem de TSA algoritmasına göre harcadığı zaman bakımından sonuca daha erken varmıştır.

M. A. ŞAHMAN (Şahman, 2021), bu çalışında Benekli Sırtlan Optimize Edicisinin (DSHO) ayrık bir versiyonunu DJSP'yi çözmek için kullanmıştır. Tesis-İş yüküne dayalı bir tesis sipariş mekanizması ile açgözlü sezgisel bir yaklaşım olan DSHO algoritması birleştirilmiştir. Dağıtık İş Atölyesi Çizelgeleme Problemi (DJSP) veri setinden 80 adet problem DSHO ile optimize edilmiştir. DSHO'nun etkinliğini değerlendirmek için, iyi bilinen JSP kıyaslama problemlerinden türetilen 480 adet örneğin sayısal olarak sonuçları, diğer dört farklı ayrık metasezgisel algoritma ile karşılaştırılarak DSHO'nun DJSP için literatüre etkin sonuçlar eklediğini göstermiştir.

M. A. ŞAHMAN ve A. O. Dündar (Şahman ve Dündar, 2024), tarafından yapılan bu çalışmada Parçacık Sürü Optimizasyonu (PSO), Yapay Arı Kolonisi (ABC) ve Boz Kurt Optimizasyonu (GWO) olmak üzere 3 adet meta-sezgisel algoritma kullanarak NP-Hard problem olan DJSP probleminin zorluğunu gidermeyi amaçlamışlardır. Bu çalışmada, operasyonların tesisler arasında dağıtılması yoluyla ürünün üretim süresini en aza indirmeyi hedefleyen iş yükü tabanlı bir tesis sıralama yöntemi uygulanmıştır. Planlama çözümleri, En Küçük Pozisyon Değeri (SPV) kodlama şemasıyla modellenmiştir. Deneysel aşamada, tesis sayısı 2 ile 4 arasında değişen 28 Atölye Planlama kıyaslama problemi ele alınmıştır. Elde edilen sonuçlar, GWO algoritmasının farklı problem senaryolarında optimum ya da optimuma

yakın çözümler üretme konusunda PSO ve ABC algoritmalarına kıyasla daha üstün bir performans sergilediğini ortaya koymaktadır.

Bu bölümde, iş atölyesi planlama probleminin çözümüne yönelik literatürde yer alan yaklaşımlar detaylı bir şekilde incelenmiştir. Bu bağlamda, sunulan literatürdeki çalışmalardan farklı olarak bu problemi çözmek için hem özgün kodlama şemaları hem de metasezgisel algoritmalar kullanılmıştır. Önerilen kodlama şemaları, problemi sürekli yapıdan ayrık yapıya dönüştürerek çözüm sürecinin etkinliğini artırmakta ve algoritmaların problem uzayında daha verimli gezinmesini sağlamaktadır.

3. İŞ ATÖLYESİ ÇİZELGELEME PROBLEMLERİ

3.1. İş Atölyesi Çizelgeleme Problemi (JSSP)

JSSP, yerleşke olarak sadece tekil üretim atölyelerindeki makineler ve iş parçalarının zaman çizelgesinin planlanmasıyla ilgili optimizasyon problemini ifade eder. Temel olarak, belirli iş parçalarının belirli makinelerde farklı işlem süresinde olan ve her iş parçasının belirli bir sıraya göre işlenmesi gerektiği karmaşık bir problemi ele alır. JSSP'nin amacı, belirli bir kriteri, örneğin belirli bir işin tamamlanma süresini minimize etmeyi karşılayacak şekilde iş parçalarını ve makineleri zaman içinde nasıl düzenleyeceğimizi belirlemektir. M. A. Şahman(Şahman, 2021) tarafından JSSP problemi formüllerle detaylı bir şekilde anlatılmıştır ve aşağıda sunulmuştur.

JSSP'nin boyutunu $n\ x\ m$ olarak düşünürsek, n adet iş vardır ve $\mathrm{J} = \{\mathrm{J}_1, \mathrm{J}_2, \mathrm{J}_3, \dots, \mathrm{J}_n\}$ şeklinde tanımlanabilir, m adet makine vardır ve $\mathrm{M} = \{\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3, \dots, \mathrm{M}_m\}$ şeklinde tanımlanabilir. Her işin tanımlanmış m adet işlemi vardır ve bu işlem $\mathrm{O}_{ij} = \{\mathrm{O}_{i1}, \mathrm{O}_{i2}, \mathrm{O}_{i3}, \dots, \mathrm{O}_{im}\}\ (\mathrm{i} = 1, 2, \dots, \mathrm{n};\ \mathrm{j} = 1, 2, \dots, \mathrm{m})$ şeklinde tanımlanır ve kesintisiz olarak sabit bir sürede gerçekleştirilir. Bu işlemler gerçekleştirilirken her makineye o anda sadece bir tane iş verilecek şekilde tanımlanır. Bir işin birden fazla işlemi olabilir, ardışık işlemler farklı makinelerde işlenebilir. Toplam işlem sayısı $O = 0, 1, 2, 3, \dots, mxn, mxn + 1$ şeklinde ifade edilebilir ve toplam işlem süresi $C_{ij} = \{C_{i1}, C_{i2}, C_{i3}, \dots, C_{im}\} (i = 1, 2, \dots, n;\ j = 1, 2, \dots, m)$ olarak tanımlanabilir. Toplam işlem sayısında tanımlanan 0 ve $mxn + 1$ noktaları herhangi bir süreç vermez, sadece başlangıç ve bitiş anlarını temsil eder. Tamamlanma süresi gibi farklı olarak tanımlanabilir.

JSSP'nin matematiksel modelini açıklayacak olursak:

Minimize $Cmax$	(3.1)
$Cs \le Ci - Ti, i = 0, 1, 2, \dots, m \times n + 1\ ;\ s \in pi$	(3.2)
$\sum_{\mathrm{j}\in\mathrm{A(t)}} Rij \le 1, m \in M; t \ge 0$	(3.3)
$Ci \ge 0, i = 0, 1, 2, \dots, m \times n + 1$	(3.4)

n	: İş sayısı
m	: Makine sayısı
Ci	: i. işlemin tamamlanma süresi, $(i = 0, 1, 2, \dots, m \times n + 1)$
Ti	: Verilen makinede i işleminin işlem süresi
Pi	: i. İşlemden önceki tüm işlemler
$A(t)$	: t anında işlenen işlem seti
Rij	: m makinesinde işlenmesini gerektiren j işlemi

JSSP'nin matematiksel modelinde, Denklem 3.1'de JSSP'nin toplam tamamlanma süresini minimuma indiren amaç fonksiyonunu ifade eder. Operasyonların birbirleriyle ilişkilerini temsil eden kısıtlamalar Denklem 3.2'de ifade edilmiştir. Bir işlemin aynı anda sadece bir makinede gerçekleşebileceğini gösteren kısıtlama Denklem 3.3'te, tamamlanma süresinin negatif olmasını engelleyen kısıtlama Denklem 3.4'te belirtilmiştir.

3.2. Dağıtık İş Atölyesi Çizelgeleme Problemi (DJSSP)

Küreselleşmeyle birlikte imalathane tesisleri dünyanın dört bir yanına dağılmıştır. DJSSP, dağıtılan bu iş atölyelerinin planlama problemi olarak ele alınır ve birden fazla yerleşim yerine sahip olan bu dağıtılmış tesislerdeki optimizasyon sırasını çözmeye çalışır. Böylelikle ekstra bir parametre olarak tesisler de eklendiği için yerleşim yerlerindeki makinelerin koordinasyonunu gerektiren daha karmaşık bir optimizasyon problemi çözülmeye çalışılır. Bu problemin çözümü, iş parçalarının farklı atölyelerdeki farklı makinelerde işlenmesini içerir ve bu makinelerin koordinasyonunu ve iş birliğini gerektirir. DJSSP'nin amacı, bu koordinasyon ve iş birliğini birbiriyle ilişkilendirirken zaman çizelgesini oluşturarak işlem süresini minimize ederek dolaylı olarak tamamlanma süresini minimize etmeye çalışır. DJSSP, farklı tesisler kullanılması yönüyle JSSP'den daha karmaşıktır. M. A. Şahman (Şahman, 2021) tarafından DJSSP problemi formüllerle detaylı bir şekilde anlatılmıştır ve aşağıda sunulmuştur.

Dağıtılmış atölye çizelgeleme probleminin matematiksel modelini tanımlayacak olursak n adet işe, m adet makineye ve f adet tesise sahiptir. Matematiksel modelin asıl amacı, tüm tesislerde maksimum yapım süresi değerini en aza indirmektir. Literatürde bazı çalışmalar maksimum ve toplam gecikmeyi amaç fonksiyonu olarak kullanmaktadır. DJSSP'de iki ana zorluk var. Birincisi işleri uygun tesislere atamak, ikincisi ise yapım süresini en aza indirerek işlerin makineler üzerindeki işlemlerini sıralamaktır. İşlerin işlemleri, belirli makinelerde sabit sürelere sahip, önceden belirlenmiş değerlerdir. Bir işin bir tesise verilmesi durumunda operasyonlar diğer tesislere bölünemez.

DJSSP'nin matematiksel modelini açıklayacak olursak:

Minimize $Cmax$ (3.5)

$\sum_{r=1}^{f} Yj,r = 1, \ \forall j$ (3.6)

$Cj,i \geq pj,i, \ \ \forall j,i$ (3.7)

$Cj,i \geq Cj,l + pj,i, \forall j,i,l \neq i | aj,i,l = 1$ (3.8)

$Cj,i \geq Ck,i + pj,i - M\,(3 - Xk,j,i - Yj,r - Yk,r)\,, \forall_{r,i,k<n,j>k}$ (3.9)

$Ck,i \geq Cj,i + pk,i - M\,(2 + Xk,j,i - Yj,r - Yk,r), \forall_{r,i,k<n,j>k}$ (3.10)

$$Cmax \geq Cj,i\,, \quad \forall j,i \tag{3.11}$$

$$Cj,i \geq 0, \forall j,i \tag{3.12}$$

$$Yj,r \in \{0,1\}, \quad \forall j,r \tag{3.13}$$

$$Xk,j,i \in \{0,1\}, \forall j < n, k > j, i \tag{3.14}$$

$$Cmax \geq 0 \tag{3.15}$$

---

n : İş sayısı , $k = 1,2,\ldots,N$

m : Makine sayısı , $l = 1,2,\ldots,M$

f : Tesis sayısı, $r = 1,2,\ldots,F$

pj,i : j işinin i makinesindeki işlem süresi

aj,i,l : İkili değişken, eğer j işi yapılırken l makinesinden hemen sonra i makinesi kullanılırsa 1 değerini alır, aksi takdirde 0 değerini alır.

M : Yeterince büyük pozitif sayı

Karar değişkenleri şu şekilde verilmektedir:

Xk,j,l : İkili değişken, i makinesinde j işi, k işinden sonra işleniyorsa 1 değerini alır, aksi takdirde 0 değerini alır.

Yj,r : İkili değişken, j işi r fabrikasında işleniyorsa 1 değerini alır, aksi takdirde 0 değerini alır.

Cj,i : Sürekli değişken, j işinin i makinesindeki işleminin tamamlanma süresi

$Cmax$: Tesislerin maksimum yapım süresi değeri

DJSSP'nin matematiksel modelinde, Denklem 3.5 ve Denklem 3.11 yapım süresini tanımlayan amaç fonksiyonudur. Denklem 3.6'daki kısıtlama her işin atandığı fabrikayı temsil eder. Denklem 3.7'deki kısıtlama bir işlemin tamamlanma süresinin işlem süresinden daha uzun olmasını sağlar. Denklem 3.8'deki kısıtlama bir işin aynı anda yalnızca bir makinede işlenebilmesini sağlar. Denklem 3.9 ve Denklem 3.10'da bir makinede aynı anda en fazla bir işi işlenebileceği kısıtlama bulunmaktadır. Denklem 3.11 yapım süresini hesaplayan kısıtlamadır. Denklem 3.12, 3.13, 3.14'te karar değişkenlerini tanımlamak için kullanılmaktadır.

4. MATERYAL

Bu tez çalışmasında, deneyler için 48 karşılaştırma problemi kullanıldı ve karşılaştırma problemlerinin özellikleri Çizelge 4.1'de verilmiştir. Ölçütler dört sınıftan oluşur. Bu sınıflar Fisher ve Thompson (ft06-ft20) (Crowston ve ark., 1963), Adams ve diğerleri (abz5-abz9) (Adams ve ark., 1988), Lawrence (la01-la30) (Lawrence, 1984), Applegate ve Cook (orb01-orb10) (Applegate ve Cook, 1991) tarafından alıntılanmıştır. Kullanılan problemlerin boyutu 72 ile 600 arasındadır. Bu problemleri dağıtık olarak 2 ile 4 tesis arasında çözümlenmiştir.

Çizelge 4.1 Karşılaştırma Problemleri ve Özellikleri

Problem Sayısı	İsim	İş x Makine	Boyut	Problem Sayısı	İsim	İş x Makine	Boyut
1	Ft06	6x12	72	25	La17	10x20	200
2	Ft10	10x20	200	26	La18	10x20	200
3	Ft20	20x10	200	27	La19	10x20	200
4	Abz5	10x20	200	28	La20	10x20	200
5	Abz6	10x20	200	29	La21	15x20	300
6	Abz7	20x30	600	30	La22	15x20	300
7	Abz8	20x30	600	31	La23	15x20	300
8	Abz9	20x30	600	32	La24	15x20	300
9	La01	10x10	100	33	La25	15x20	300
10	La02	10x10	100	34	La26	20x20	400
11	La03	10x10	100	35	La27	20x20	400
12	La04	10x10	100	36	La28	20x20	400
13	La05	10x10	100	37	La29	20x20	400
14	La06	15x10	150	38	La30	20x20	400
15	La07	15x10	150	39	Orb01	10x20	200
16	La08	15x10	150	40	Orb02	10x20	200
17	La09	15x10	150	41	Orb03	10x20	200
18	La10	15x10	150	42	Orb04	10x20	200
19	La11	20x10	200	43	Orb05	10x20	200
20	La12	20x10	200	44	Orb06	10x20	200
21	La13	20x10	200	45	Orb07	10x20	200
22	La14	20x10	200	46	Orb08	10x20	200
23	La15	20x10	200	47	Orb09	10x20	200
24	La16	10x20	200	48	Orb10	10x20	200

5. YÖNTEM

NP-Hard bir optimizasyon problemi olan DJSSP'ler farklı optimizasyon yaklaşımları kullanılarak çözümlenebilmektedir. Bu tez çalışmasında metasezgisel yaklaşımlardan faydalanılmıştır. Bu metasezgisel algoritmalar sürekli optimizasyon problemlerin çözümünde kullanılacak şekilde önerilmişlerdir. Fakat DJSSP'ler ayrık optimizasyon problemleri olmalarından dolayı, sürekli çalışan bu algoritmalara ait çözümler ayrıklaştırılması gerekmektedir. Bu ayrıklaştırma işlemi aşağıda detayını verdiğimiz kodlama şemaları ile sağlanmaktadır. Tez çalışmasında, JSSP'lerden farklı olarak dağıtık yapıdaki DJSSP'ler üzerinde çalışıldığı için işlerin tesislere verimli bir şekilde dağıtılması gerekmektedir, bunun için de Tesis İş Yükü Dağıtımı yaklaşımından faydalanılmıştır. Alt başlıklarda tez çalışmasında kullanılan her bir metasezgisel yaklaşım, tesis seçimi için kullanılan Tesis İş Yükü Dağıtımı ve ayrıklaştırma sürecinde kullanılan kodlama şemaları hakkında detaylı bilgi verilmiştir.

5.1. Metasezgisel Algoritmalar

Metasezgisel algoritmalar, NP-Hard problemleri gibi büyük ölçekli optimizasyon problemlerinin çözümü için en iyiye yakın sonuçlar veren algoritmalardır. Metasezgisel optimizasyon algoritmaları, doğadan esinlenerek doğadaki popülasyon örneklerini algoritmanın popülasyonu olarak algılayan dinamik bir yapıdadır. Doğadan esinlenme tavrını yürütürken popülasyondaki sürü, genetik, sosyal, fizik, müzik gibi farklı değişkenler altında incelemektedir. Bu çalışmamızda on adet metasezgisel algoritma kullanılmıştır ve bu algoritmalar aşağıda açıklanmıştır.

5.1.1. Parçacık Sürü Optimizasyonu Algoritması (Particle Swarm Optimization-PSO)

Parçacık Sürü Optimizasyonu (PSO), Dr. James Kennedy ve Dr. Russell Eberhart (Kennedy ve Eberhart, 1995) tarafından 1995 yılında geliştirilmiştir. Sürü halinde hareket eden balık ve kuşları bir popülasyon olarak görüp, inceleyip, sosyal davranışları gözlemlenerek, en temelinde ise sürü zekasına odaklanan optimizasyon algoritmasıdır. Sürü halindeki bu hayvanların yiyecek arayışı, kendilerini güvenli bir ortamda bulundurmak için yaptıkları rastgele hareketlere odaklanır. Her bir hayvanı parçacık olarak gören bu algoritma, her bir parçacık için kendisinden bir önceki parçacığın tecrübesinden faydalanarak kendisini sürüdeki en iyi pozisyona taşımaya çalışır. Algoritmanın temelinde ise her bir parçacığın sürüdeki en iyi pozisyona sahip olan parçacığa doğru yaklaştırılması sağlanır. Rastgele tanımlanan yaklaşma

hızı ile en iyi parçacığa yaklaştırılma sağlanırken bir önceki konumdan daha iyi konuma gelinceye kadar en nihayetinde hedefe ulaşana kadar devam eder.

PSO Algoritmasının matematiksel modelini açıklayacak olursak:

xi, vi	(5.1)
$vi(t+1) = w \cdot vi + c1 \cdot r1 \cdot (pi - xi(t)) + c2 \cdot r2 \cdot (g - xi(t))$	(5.2)
$xi(t+1) = xi(t) + vi(t+1)$	(5.3)
En İyileri Güncelleme	
Parçacık için: Eğer $f(xi(t+1)) < f(pi)$ ise $pi = xi(t+1)$	(5.4)
Sürü için: Eğer $f(xi(t+1)) < f(g)$ ise $g = xi(t+1)$	(5.5)

$x_i(t)$: i parçacığının t zamanındaki konumu

$v_i(t)$: i parçacığının t zamanındaki hızı

p_i : i parçacığının şimdiye kadarki en iyi bulduğu konum(pbest)

g : Sürüdeki tüm parçacıklar arasındaki en iyi konum(gbest)

w : Atalet katsayısı

c_1, c_2 : Öğrenme faktörleri, kişisel ve küresel en iyi konumların etkisini belirler

r_1, r_2 : Rastgele üretilen sayılar

PSO Algoritmasının matematiksel modelinde öncelikle Denklem 5.1'de parçacıkların konum ve hızlarının rastgele başlatılması için değişkenler tanımlandı. Parçacıkların daha iyi bir konuma geçmesi için gerekli olan hız güncellemesi Denklem 5.2'de verilmiştir. Denklem 5.3'te güncellenen hız vektörü ve eski konum kullanılarak yeni konum elde edilir. Denklem 5.4' te parçacık için en iyi güncelleme denetlenirken Denklem 5.5'te ise sürü için en iyi güncelleme yapılır.

5.1.2. Yapay Arı Kolonisi Algoritması (Artificial Bee Colony-ABC)

2005 yılında Karaboga (Karaboga, 2005) tarafından tanıtılan bu algoritma, bal arılarının yemek arama davranışını taklit eden bir metasezgisel optimizasyon algoritmasıdır. ABC algoritması popülasyonu yapay arı olan ve potansiyel çözümleri arayan bir gruptur, bu arayışta bulunmaları optimizasyon problemini en iyi çözüme yaklaştırır. ABC algoritması, içerisinde tür olarak işçi, gözcü ve izci arıların bulunduğu bir bal arısı kolonisinin yapısını taklit eder. Bu 3 tür arıyı açıklayacak olursak:

- İşçi Arılar (Employed Bees)

İşçi arılar, bir yandan potansiyel çözüm adaylarını ararken bir yandan da mevcut çözümleri temsil eder. İşçi arılarda her bir birey çözümü ifade eden bir bal peteği seçer. İşçi arılar

seçilen peteğin konumundayken bu çözümü rasgele değiştirerek yeni bir çözüm elde etmeye çalışır. Elde edilen yeni çözüm eski çözümden daha iyi bir performans gösterirse, eski çözümü unutarak yeni çözümü akılda tutar.

- Gözcü Arılar (Onlooker Bees):

Gözcü arılar, işçi arılar tarafından bulunan yeni çözümleri inceler ve en iyi çözümleri seçerler. Bulunan çözümlerin kalitesine göre farklı çözümlere odaklanan gözcü arılar çözümlerin değerlerini değerlendirir. En iyi çözümlere odaklanan gözcü arılar, bu çözümleri hatırlar ve diğer işçi arılara iletmek için dans ederler.

- İzci Arılar (Scout Bees):

İzci arılar, belirli bir süre boyunca daha iyi bir çözüm bulamayan işçi arıları temsil eder. Rastgele yeni çözümler üreten bu izci arılar, potansiyel olarak daha iyi bir çözüm bulmaya çalışırlar. Eğer bir izci arı daha iyi bir çözüm bulursa, bu çözümü diğer arılarla paylaşır.

ABC algoritmasının matematiksel modelini açıklayacak olursak:

$$vi = xi + \phi ij \cdot (xi - xk) \quad (5.6)$$

$$(f(vi) > f(xi)) \text{ ise } f(xi) = f(vi) \quad (5.7)$$

$$p_i = \frac{f_{(x_i)}}{\sum_{J=1}^{N} f(x_J)} \quad (5.8)$$

$$vi = xi + \phi ij \cdot (xi - xk) \quad (5.9)$$

xi: Problem uzayındaki çözümü temsil eder.

$f(xi)$: Çözümün uygunluğunu ve kalitesini ölçer.

vi: Yeni çözümler

ϕij: Rastgele bir sayı (-1,1)

xk: Rastgele seçilmiş başka bir çözüm

pi: x_i çözümünün seçilme olasılığı

ABC algoritmasının matematiksel modelinde öncelikle x_i çözümü rastgele başlatılarak, Denklem 5.6'da işçi fazında her işçi arı kendi mevcut çözümü iyileştirmeye çalışır. İyileştirilen çözümün uygunluğu ise Denklem 5.7 de doğrulanır ve yeni duruma göre değerler atanır. Her çözümün seçilme olasılığı Denklem 5.8' de gösterildiği gibi hesaplanır. Gözlemci arı fazında, işçi arılar tarafından iyileştirilen çözümler arasından seçtikleri çözümleri Denklem 5.9 ile daha da iyileştirirler ve yeni çözümler elde ederler. Keşifçi arı fazında ise eğer çözüm belirli bir deneme sayısında iyileştirilemezse, o çözüm keşifçi arılar tarafından rastgele bir çözümle değiştirilir.

5.1.3. Gri Kurt Optimize Edici (Grey Wolf Optimizer – GWO)

Gri Kurt Optimize Edici, 2014 yılında S. Mirjalili ve ark (Mirjalili ve ark., 2014) tarafından geliştirilmiş bir metasezgisel optimizasyon algoritmasıdır. GWO, popülasyonu gri kurt sürüsünün olduğu ve liderlik yapısını, av avlama stratejilerini taklit eden bir algoritmadır. Bu algoritma, optimize edilmesi gereken bir problemin en iyi çözüme ulaşmak için gri kurt sürüsünün doğal davranışını taklit ederek çalışır. GWO algoritması şu adımları takip eder:

Başlangıç: Rastgele dağıtılan gri kurt sürüsü başlangıç pozisyonlarına yerleştirilirler. Rastgele dağıtılan bu başlangıç pozisyonlar, problem alanındaki potansiyel çözümleri ifade eder.

Gri Kurt Hiyerarşisi: Sürü içindeki her gri kurt kendi aralarında lider (alfa), beta ve delta olarak adlandırılan üç farklı pozisyona sahiptir. Lider diğerleri arasında en iyi çözümü ifade ederken, beta ve delta diğer iyi çözümleri ifade eder.

Av Avlama Davranışı: Alfa, beta ve delta kurtlarının her biri, av avlama stratejilerini kullanarak yeni potansiyel çözümler üretirler. Bu stratejiler, liderin konumunu daha iyiye güncellemek için en iyi, beta ve delta kurtlarının konumlarına göre bir takım hesaplamayı içerir.

Pozisyon Güncelleme: Yeni üretilen pozisyonlar, mevcut liderlik yapısı ve av avlama stratejileri kullanılarak tekrardan değerlendirilir. Eğer yeni konumlar mevcut liderlik yapısından daha iyi konumdaysa, lider (alfa), beta ve delta pozisyonları güncellenir.

İterasyon: Optimizasyon için belirlenen bir iterasyon sınırına veya herhangi bir kritere takılıncaya kadar adımlar tekrarlanır. Bu süreçte sürü üyelerinin pozisyonlarını iyileştirerek en iyi çözüme yaklaşmalarını sağlar.

GWO algoritmasının matematiksel modelini açıklayacak olursak:

$$Xi \tag{5.10}$$

$$D\alpha = |C1 \cdot X\alpha - X| \tag{5.11}$$

$$D\beta = |C2 \cdot X\beta - X| \tag{5.12}$$

$$D\delta = |C3 \cdot X\delta - X| \tag{5.13}$$

$$X1 = X\alpha - A1 \cdot D\alpha \tag{5.14}$$

$$X2 = X\beta - A2 \cdot D\beta \tag{5.15}$$

$$X3 = X\delta - A3 \cdot D\delta \tag{5.16}$$

$$X(t+1) = \frac{X1+X2+X3}{3} \tag{5.17}$$

$$A = 2a \cdot r1 - a \tag{5.18}$$

$$C = 2 \cdot r2 \tag{5.19}$$

$$a(t) = 2\left(1 - \frac{t}{T}\right) \quad (5.20)$$

Kurtlar: Çözüm uzayındaki bireyler

Alfa (α): En iyi çözümü ifade eder.

Beta (β) ve Delta (δ): İkinci ve üçüncü en iyi çözümü ifade eder.

Omega (ω): Diğer çözümler

xi: Kurtların konumlarını ifade eder.

A ve C: Arama ajanlarını güncellemek için kullanılan katsayılar.

a: Doğrusal olarak azalan bir vektördür.

r1, r2: Rastgele vektörlerdir. (0,1)

t: Mevcut iterasyon sayısı

T: Toplam iterasyon sayısı

GWO algoritmasının matematiksel modelinde öncelikle Denklem 5.10'da kurtların konumları rastgele dağıtılır. Kurtlar arasında hiyerarşi olarak Alfa(α), Beta(β), Delta(δ) ve Omega(ω) kurtları belirlenir. Bu kurtlar, sürünün hiyerarşik liderleridir ve en iyi çözümleri temsil eder. Avlanma sürecinde kurtlar, Denklem 5.11 ve Denklem 5.17 arasında avın (optimum çözümün) yerini belirlemek ve ona daha yakın olabilmek için Alfa, Beta ve Delta tarafından yönetilirler. Her bir kurt, liderlerin etrafında avına daha da yakınlaşarak yakalamaya çalışırlar. Denklem 5.18 ve Denklem 5.19'da A ve C, her durumda kurtları güncellemek için kullanılan katsayılardır. Atalet katsayısı olan a, Denklem 5.20' de olduğu gibi doğrusal bir grafik izlemektedir. Belirli bir iterasyon sayısına veya belirli bir durma kriterine ulaşıldığı zamana kadar bu adımlar tekrarlanır.

5.1.4. Kızıl Tilki Optimize Edici (Red Fox Optimizer – RFO)

Sürekli optimizasyon problemlerinin çözümü için Hardi Mohammed ve Tarik Rashid (Mohammed ve Rashid, 2023) tarafından önerilen FOX optimizasyon algoritması, kızıl tilkilerin avlanma yaklaşımlarını modelleyerek ifade etmektedir. FOX optimizasyon algoritması özellikle kızıl tilkilerin zorlu kış koşullarında ses dalgalarını takip ederek avının yerini nasıl tespit ettiğine ve avın yerini tespit ettikten sonra hangi taraftan saldıracağına nasıl karar verdiğine odaklanıyor. Başlangıçta kızıl tilki, avını keşfetmek için arama alanında rastgele hareket eder. Bu rastgele yürüyüş, FOX algoritmasında keşifsel davranış sağlamak için ilham alınır ve kullanır. Kızıl tilki avının ultrasonunu duyarak avına yönelebilir. Eğer ortam avını görmeye uygun olmadığında ise çevredeki sesleri dinleyip takip etmeye başlar. Eğer ses duyarsa, gelen sese göre mesafeyi belirleyerek avına yaklaşır. Sesin geldiği yeri tam olarak

belirledikten sonra hangi yönden atlayıp avına saldıracağına karar verme süreci başlar. Kızıl tilki avının üzerine kuzeydoğudan atlarsa %82 başarı oranına sahiptir. Bu nedenle kuzeydoğuya atlamak algoritmanın başarısında çok önemli bir faktördür. Kısaca özetlemek gerekirse:

- Kızıl tilki rastgele av aramaya çalışır.
- Kızıl tilki, avından gelen ultra sesi duyarak avını bulur. Daha sonra avın yakınına varmak zaman alır.
- Kızıl tilki, avının sesini ve saat farkını dinleyerek avıyla arasındaki mesafeyi belirler
- Mesafeyi belirledikten sonra kızıl tilki, avı yakalamak için gereken atlamayı tahmin eder.
- Yürüyüş minimum süreye ve en iyi pozisyona göre gerçekleştirilir.

Kızıl tilki optimizasyonuyla avını yakalamak için minimum süreye sahip bu konumu bulunması amaçlanmıştır.

RFO algoritmasını açıklayacak olursak:

$$Dist\_S\_Tit = Sp\_S \cdot Time\_S\_Tit \quad (5.21)$$

$$Dist\_Fox\_Preyit = Dist\_S\_Tit \cdot 0{,}5 \quad (5.22)$$

$$Jumpit = 0{,}5 \cdot 9{,}81 \cdot t \quad (5.23)$$

$$X(it+1) = Dist\_Fox\_Preyit \cdot Jumpit \cdot C1 \quad (5.24)$$

$$X(it+1) = Dist\_Fox\_Preyit \cdot Jumpit \cdot C2 \quad (5.25)$$

$$\mathrm{tt} = \frac{\mathrm{sum(Time\_S\_Tit(i,;))}}{\mathrm{dimension}},\ MinT = Min(tt) \quad (5.26)$$

$$\mathrm{a} = 2 \cdot \left(\mathrm{it} - \left(\frac{1}{\mathrm{Max_{it}}}\right)\right) \quad (5.27)$$

$$X(it+1) = BestXit \cdot rand(1, dimension) \cdot MinT \cdot a \quad (5.28)$$

$Dist\_S\_T_{it}$: Yeni bir konum bulmak için sesin katettiği mesafe

$Time\_S\_T_{it}$: Sesin seyahat süresi [0,1]

$Dist\_Fox\_Prey_{it}$: Kızıl tilkinin avına olan mesafesi

Sp_s: Ses hızı

$BestPosition_{it}$: Kızıl Tilkinin en iyi konumu

$Jump_{it}$: Atlama yüksekliği

C_1: 0,18

C_2: 0,82

X_{it}: Tilkinin Konumu

a: Doğrusal olarak azalan bir vektör

p: Rastgele değişken
MinT: Minimum zaman değişkeni
tt: Minimum zaman ortalaması
Max_{it}: Maksimum yineleme

Rastgele değişken p'nin değeri [0,1] aralığındadır. Bu nedenle p rastgele sayısı 0,18'den büyükse kızıl tilkinin yeni konumunun bulunması gerekir. Yeni bir konum bulmak için sesin kat ettiği mesafe Dist_S_Tit, kızıl tilkinin avına olan mesafesi Dist_Fox_Preyit ve atlama değeri Jumpit hesaplanmalıdır. Denklem 5.21'de sömürü aşamasında ses hızıyla sesin havada geçirdiği süre çarpılarak sesin katettiği mesafe bulunur. Kızıl tilki tarafından gönderilen ses dalgası avına çarpıp geri döndüğünden aslında yol iki kez kullanılmış olur, av ile avcı arasındaki mesafe Denklem 5.22'de hesaplandığı gibi mesafenin yarısı anlamına gelmektedir. Tilki ile av arasındaki mesafe bulunduktan sonra avını yakalamak için atlamak zorunda kalır, Denklem 5.23 ile atlama yüksekliğini hesaplaması gerekir. Denklem 5.23'te 9,81 yer çekimi ivmesidir ve t atlama esnasında geçirilen süre olduğu için hem yukarı hem aşağı bir hareket yapılmaktadır bu yüzden 0,5 ile çarpılmıştır. Denklem 5.24 ve Denklem 5.25'te kızıl tilkiye yeni bir yer bulmak için kullanılır ve p koşulunun kullanılasıyla bunlardan sadece bir tanesi seçilir, p değeri 0,18 den büyükse C_1, küçük eşitse C_2 kullanılır. Burada rastgele üretilen p değeri ile kızıl tilkinin kuzeydoğu yönüne atlayıp atlamadığı anlaşılarak avını öldürme şansı ortaya çıkar. Keşif olarak rastgele yürüyüş yapan kızıl tilki, o ana kadar bulunduğu en iyi konuma göre rastgele arama yapar. Bu aşamada atlama tekniğine sahip değildir çünkü öncelikle arama uzayında avı keşfetme gereksinimi duymaktadır. Kızıl tilkinin en iyi konuma rastgele yürüyebilmesi için Denklem 5.26 ve Denklem 5.27'de gösterildiği gibi minimum zaman değişkeni (MinT) ve a değişkeni kullanılır. Minimum zaman ortalamasını bulmak için sesin havada geçirdiği sürelerin toplamının probleminin boyutuna bölünmesiyle elde edilir. Kızıl tilki, Denklem 5.28'deki $X_{(it+1)}$ arama uzayında yeni bir konum ararken keşif tekniğini gösteren bu denklemde rastgele bir sayı olan bir değişkenle arama ve kullanım aşamalarını dengelemek için kullanılır ve en iyi çözüm $BestX_{it}$'in keşif aşamasında büyük etkisi vardır.

5.1.5. Jaya Algoritması (JAYA)

Jaya algoritması, 2015 yılında R. Venkata Rao arkadaşları (Rao ve Patel, 2012) tarafından önerilen, parametresiz bir metasezgisel optimizasyon algoritmasıdır. "Jaya" kelimesi Sanskritçe'de "başarı" veya "kazanmak" anlamına gelir. Jaya algoritması, her iterasyonda popülasyondaki bireyleri daha iyi çözümlere doğru yönlendirirken, aynı zamanda kötü

çözümlerden uzaklaştırmayı amaçlar ve bu yaklaşım, evrimsel algoritmalardaki çaprazlama, mutasyon veya hız güncelleme gibi ek mekanizmalara ihtiyaç duymadan çalışmasını sağlar. Parametre ayarlaması gerektirmemesi, algoritmanın öne çıkan avantajıdır. Yeni çözüm daha iyi bir uygunluk değerine sahipse, mevcut çözümle yer değiştirir.

JAYA Algoritmasının matematiksel modelini açıklayacak olursak:

$$x_{i,j}^{new} = x_{i,j} + r_1(x_j^{best} - | x_{i,j} | - r2(x_j^{worst} - | x_{i,j} |) \quad (43) \qquad (5.29)$$

$x_{i,j}$: Bireyin j. değişkeni

x_j^{best} : Mevcut iterasyondaki en iyi bireyin j. bileşeni

x_j^{worst} : En kötü bireyin j. bileşeni

$r_1, r_2 \in [0,1]$: Rastgele üretilen sayı değerleri

Jaya algoritması, sadeliğin gücünü gösteren etkili bir optimizasyon tekniğidir. Parametresiz yapısı sayesinde kullanıcı dostu bir uygulama sunar ve birçok farklı problemde etkili çözümler üretir. Her ne kadar bazı durumlarda çeşitlilik eksikliği nedeniyle sınırlamaları olsa da genel performansı itibariyle günümüz metasezgisel algoritmalarının önemli bir üyesi hâline gelmiştir.

5.1.6. Yapay Alg Optimizasyon Algoritması (Artificial Algae Algorithm-AAA)

Yapay Alg Optimizasyon Algoritması (AAA), 2015 yılında Sait Ali Uymaz, Gülay Tezel ve Esra Yel (Uymaz ve ark., 2015) tarafından geliştirilmiştir. Bu algoritma, doğadan esinlenen metasezgisel optimizasyon yöntemleri arasında yer almaktadır. AAA, mikro alglerin yaşam döngüsünden ve davranışlarından esinlenerek geliştirilmiştir. Algoritma, mikro alglerin fotosentez, yüzme hareketleri ve adaptasyon yeteneklerini model alarak, bu biyolojik süreçleri optimizasyon problemlerine uyarlamaktadır.

AAA Algoritmasının matematiksel modelini açıklayacak olursak:

- Helisel Hareket (Helical Movement)

Algoritmanın başlangıcında, çözüm uzayında rasgele yerleştirilen bir dizi "alga", helisel hareketlerle daha iyi çözümlere yönlendirilir. Mikro alglerin besin kaynaklarına veya ışığa yönelmesi gibi hareketleri simüle eder. Bir alganın 1'inci pozisyonu, helisel hareketi şu şekilde güncellenir:

$$X_i(t+1) = X_i(t) + \alpha \cdot \sin(b \cdot t) \cdot d \qquad (5.30)$$

$X_i(t)$: i'nci alganın konumu t zamanında

α, b : Helisel hareketin genlik ve frekans parametreleri

d : İleri yönlü hareket vektörü

t : Zaman adımı

- Evrimsel Süreç (Reproductive Process)

Mikro alglerin çoğalma ve bölünme süreci, popülasyondaki çeşitliliği artırmak için kullanılır. Evrimsel süreç, popülasyondaki algaların kendilerini ve çevrelerine uyum sağlayarak daha iyi çözümlere yaklaşmalarını sağlar. Evrimsel sürecin bir örneği şu şekilde modellenebilir. Bir popülasyondaki alganın güncellenen pozisyonu şu şekilde belirlenir:

$$X_i(t+1) = X_i(t) + \lambda \cdot (X_{best}(t) - X_i(t)) \quad (5.31)$$

$X_i(t)$: i'nci alganın mevcut konumu

$X_{best}(t)$: Popülasyonun en iyi çözümünün konumu

λ: Evrimsel parametre (genellikle 0-1 aralığındadır.)

Bu denklemler, popülasyondaki algaların daha iyi çözümleri "genetik" olarak miras kalmasını sağlar.

- Adaptasyon Süreci (Adaptation Process)

Mikro algler çevresel değişimlere uyum sağlarlar. Benzer şekilde, yapay alg algoritması çözüm uzayındaki yerel optimizasyonu ve çözümün yakınsamasını sağlamak için adaptasyon sürecini kullanır. Adaptasyon genellikle, çevreye uyum sağlamak amacıyla çözümün iteratif olarak iyileştirilmesi ile yapılır. Bir alganın çözüm uzayındaki pozisyonu çevresel faktörlere bağlı olarak şu şekilde güncellenebilir:

$$X_i(t+1) = X_i(t) + \mu \cdot (X_{best}(t) - X_i(t)) \quad (5.32)$$

μ: Adaptasyon katsayısı (genellikle 0-1 aralığındadır.)

$X_{best}(t)$: En iyi bulunan çözüm

Bu süreç, çözümün daha hızlı yakınsamaya ulaşmasına yardımcı olur.

- Genel Güncelleme Modeli

AAA algoritmasının genel güncelleme kuralı, yukarıda belirtilen helisel hareket, evrimsel süreç ve adaptasyon süreçlerinin birleşimiyle elde edilir. Çözüm uzayındaki bir alganın genel güncellenmiş pozisyonu şu şekilde ifade edilebilir:

$$X_i(t+1) = X_i(t) + \alpha \cdot \sin(b \cdot t) \cdot d + \lambda \cdot (X_{best}(t) - X_i(t) + \mu \cdot (X_{best}(t) - X_i(t) \quad (5.33)$$

5.1.7. Ağaç-Tohum Algoritması (Tree-Seed Algorithm-TSA)

Ağaç-Tohum Algoritması (TSA), 2015 yılında Dr. Mustafa Servet Kıran (Kiran, 2015) tarafından geliştirilmiştir. Bu algoritma, sürekli optimizasyon problemlerini çözmek amacıyla doğadan esinlenen bir metasezgisel yöntem olarak önerilmiştir. TSA, doğadaki ağaçların tohum üretimi ve bu tohumların yeni ağaçlara dönüşme sürecinden esinlenmiştir. Bu biyolojik süreç, algoritmanın çözüm uzayında yeni çözümler üretme ve mevcut çözümleri iyileştirme mekanizmasına temel oluşturur. TSA Algoritmasının matematiksel modelini açıklayacak olursak:

- Başlangıç Popülasyonu

Algoritma, rastgele oluşturulan bir ağaç popülasyonu ile başlar. Her ağaç, potansiyel bir çözümü temsil eder.

- Tohum Üretimi

Her ağaç, belirli sayıda tohum üretir. Tohum sayısı genellikle popülasyonun %10 ila %25'i arasında belirlenir. Her tohum, aşağıdaki iki denklemden biri kullanılarak oluşturulur.

En İyi Çözüme Yaklaşma: $S_{i,j} = T_{i,j} + r \cdot (B_j - T_{i,j})$ (5.34)

Rastgele Arama: $S_{i,j} = T_{i,j} + r \cdot (T_{r,j} - T_{i,j})$ (5.35)

$S_{i,j}$: i'nci ağacın j'nci boyuttaki tohumu

$T_{i,j}$: i'nci ağacın j'nci boyuttaki değeri

B_j: En iyi çözümün j'nci boyuttaki değeri

$T_{r,j}$: Rastgele seçilen bir ağacın j'nci boyuttaki değeri

r: [-1,1] aralığında rastgele bir sayı

Hangi denklemin kullanılacağı, Arama Eğilimi (Search Tendency - ST) parametresine bağlıdır. ST, [0,1] aralığında bir değerdir ve genellikle 0.5 olarak seçilir.

- Seçim ve Güncelleme

Her ağacın ürettiği tohumlar değerlendirilir ve en iyi tohum, ana ağacın yerine geçer. Bu süreç, popülasyonun genel kalitesini artırmayı hedefler.

- Döngü ve Sonlandırma

Yukarıdaki adımlar, belirlenen iterasyon sayısına ulaşılana kadar tekrarlanır. Sonuçta, en iyi çözüm raporlanır.

5.1.8. Lévy Uçuş Dağıtımı Algoritması (Lévy Flight Distribution-LFD)

Lévy Flight Distribution (LFD) algoritması, Prof. Dr. Essam H. Houssein ve çalışma arkadaşları (Houssein ve ark., 2020) tarafından geliştirilmiş ve 2020 yılında yayımlanmıştır. Bu algoritma, mühendislik optimizasyon problemlerini çözmek amacıyla yeni bir metasezgisel yöntem olarak önerilmiştir. LFD algoritması, doğada bazı hayvanların av arama davranışlarını modelleyen Lévy uçuşu (Lévy flight) olgusundan esinlenmiştir. Lévy uçuşu, kısa mesafeli aramaların yanı sıra ara ara yapılan uzun mesafeli sıçramaları içeren bir rastgele yürüyüş modelidir. Bu davranış, çözüm uzayında hem yerel hem de küresel arama yeteneklerini dengelemek için kullanılır. LFD Algoritmasının matematiksel modelini açıklayacak olursak:

Lévy Uçuşu Mekanizması

$$X_i^{t+1} = X_i^t + \alpha \cdot Levy(\lambda) \quad (5.36)$$

$$Levy{\sim}u = t^{-\lambda}, (1 < \lambda \leq 3) \quad (5.37)$$

X_i^{t+1}: i'nci çözüm adayının t zamanındaki konumu

α: Adım boyutunu kontrol eden skaler

$Levy(\lambda)$: Levy dağılımından örneklenen rastgele bir vektör

Bu dağılım, nadiren de olsa çok büyük adımların atılmasına olanak tanır, bu da algoritmanın yerel minimumlara takılmadan küresel optimuma ulaşma şansını artırır.

5.1.9. Diferansiyel Arama Algoritması (Differential Search Algorithm - DSA)

Diferansiyel Arama Algoritması (DSA), 2012 yılında Prof. Dr. Pinar Civicioglu (Civicioglu, 2012) tarafından geliştirilmiştir. Bu algoritma, biyolojik türlerin göç davranışlarından esinlenmiştir. Algoritma, popülasyonun yeni yaşam alanları arayışında rastgele hareketler yaparak daha iyi çözümler bulma sürecini modellemektedir. Belirli bir yılda gerçekleşen mevsimsel değişiklikler, çeşitli türlerin (eusocial, subsocial veya presocial) hayatta kalmak için güvendiği kaynakların çoğunu (meralar, su kütleleri, vb.) etkiler. Sonuç olarak, doğadaki çok sayıda tür (örneğin kuşlar, kelebekler, ateş böcekleri, bal arıları ve balinalar) periyodik olarak göç etmeye zorlanır. DSA, özellikle mühendislik tasarımı ve yapısal optimizasyon gibi alanlarda uygulanmıştır. DSA Algoritmasının matematiksel modelini açıklayacak olursak:

$$X_i = [x_{i,j}] \quad (5.38)$$

$$Superorganism_g = X_i \quad (5.39)$$

$$x_i = rand \cdot \left(up_j - low_j\right) + low_j \quad (5.40)$$

$$donor = [x_{Random\_Shuffling(i)}] \quad (5.41)$$

$$Scale = randg[2 \cdot rand1] \cdot (rand2 - rand3) \quad (5.42)$$

$$StopoverSite = Superorganism + Scalex \cdot (donor - Superorganism) \quad (5.43)$$

Denklem 5.38'de yapay organizmalar, Denklem 5.39'ta süper organizmalar tanıtılmıştır. Süper organizmalar Denklem 5.40 ile üst ve alt değerlere göre rastgele oluşturulur. Rastgele oluşturulan yapay organizmalar, yeni bir durak keşfetmek ve başarılı bir göçü tamamlamak için donör hedefine göç eder. Genellikle, DSA üzerindeki çalışmalarda, donör Denklem 5.41 kullanılarak elde edilir. Süper organizmaların göçü sırasında, pozisyon boyutundaki değişiklik ölçek faktörü aracılığıyla yönetilir. Ölçek değeri gama rastgele sayı üreteci (randg) kullanılarak elde edilir ve tekdüze rastgele sayı üreteci (rand) tarafından kontrol edilir. Denklem 5.42, gösterilen ölçek değerini elde etmek için kullanılır. Süper organizmaların elde ettiği ölçek ve donör değerleri, Denklem 5.43 kullanılarak yapılan duraklama alanının hesaplanmasında büyük bir etkiye sahiptir. DSA'nın sonraki aşamalarında, yeni elde edilen duraklama alanı önceki süper organizmalarla karşılaştırılır ve süreç daha iyi çözüm kullanılarak devam eder.

5.1.10. Balina Optimizasyon Algoritması (Whale Optimization Algorithm- WOA)

Balina Optimizasyon Algoritması (WOA), Seyedali Mirjalili (Mirjalili ve Lewis, 2016) tarafından 2016 yılında geliştirilmiş bir doğadan esinlenen metasezgisel optimizasyon algoritmasıdır. WOA, doğadaki kambur balinaların (humpback whales) baloncuk ağı tekniği olarak bilinen avlanma davranışından esinlenerek geliştirilmiştir. Bu davranışta balinalar, avlarını spiral şeklinde yukarı doğru çıkararak çevreler ve yavaşça daralan bir halka içinde avlarını yakalarlar. WOA, bu davranışı matematiksel olarak modelleyerek optimizasyon problemlerine uygular. WOA Algoritmasının matematiksel modelini açıklayacak olursak:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_{p(t)}} - \overrightarrow{X_{(t)}}\right| \quad (5.44)$$

$$\overrightarrow{X_{(t+1)}} = \overrightarrow{X_{p(t)}} - \vec{A} \cdot \vec{D} \quad (5.45)$$

$$\vec{A} = 2\vec{a} \cdot \overrightarrow{r_1} - \vec{a} \quad (5.46)$$

$$\vec{C} = 2 \cdot \overrightarrow{r_2} \quad (5.47)$$

$$\overrightarrow{X_{(t+1)}} = \overrightarrow{D'} \cdot e^{bt} \cdot \cos(2\pi t) + \overrightarrow{X^*_{(t)}} \quad (5.48)$$

$$\vec{D} = \left|\overrightarrow{X^*_{(t)}} - \overrightarrow{X_{(t)}}\right| \quad (5.49)$$

$$\overrightarrow{X_{(t+1)}} = \begin{cases} \overrightarrow{X^*_{(t)}} - \vec{A} \cdot \vec{D}, & p < 0.5 \\ \overrightarrow{D'} \cdot e^{bt} \cdot \cos(2\pi t) + \overrightarrow{X^*_{(t)}}, & p \geq 0.5 \end{cases} \quad (5.50)$$

t: Yineleme sayısı

$\vec{A}$,$\vec{C}$: Katsayı vektörleri

$\vec{a}$: Yinelemeler boyunca 2 den 0 a doğrusal düşen vektör

r_1, r_2: [0,1] aralığıda rastgele vektör

$\overrightarrow{X_p}$: Avın pozisyon vektörü

$\vec{X}$: Balinanın pozisyon vektörü

$\overrightarrow{D'}$: i'inci balinanın ava olan uzaklığı (şimdiye kadar elde edilen en iyi çözüm)

b: Logaritmik sarmalın şeklini tanımlayan bir sabit ve [-1,1] aralığındaki rastgele bir sayıdır.

Balina türü olarak kambur balinalardan esinlenen bu algoritmada, balinalar avının yerini tanıyabilir ve ona yaklaşarak avını çevreleyebilir. En iyi konuma yaklaşma esnasında tasarımın arama alanındaki konumu önceden bilinmediğinden, WOA algoritması mevcut en iyi aday çözümün hedefi olan av olduğunu veya en iyiye yakın olduğunu farz eder. Arama aracıları arasından en iyisi kendi aramasını tanımlandıktan sonra, diğer arama araçları da mevcut konumlarını en iyi arama aracısına doğru yenilemeye çalışacaktır. Bu yenileme davranışı Denklem 5.44 ve Denklem 5.45 ile temsil edilir. Bu denklemlerde t geçerli yinelemeyi gösterir. Katsayı vektörleri Denklem 5.46 ve Denklem 5.47 ile belirlenir. Kambur balinaların balon ağlarına karşı davranışları Denklem 5.48 ve Denklem 5.49 ile matematiksel olarak modellenmiştir. Kambur balinaların avın etrafında küçülen bir daire içinde ve aynı anda spiral şekilli bir yol boyunca yüzer. Bu eş zamanlı davranışı modellemek için, optimizasyon sırasında balinaların konumunu güncellemek için küçülen çevreleme mekanizması veya spiral model arasında seçim yapma olasılığının %50 olduğunu düşünülür ve Denklem 5.50'de matematiksel model olarak gösterilmiştir.

5.2. Tesis İş Yükü Dağıtımı

DJSSP ile JSSP arasında tesis sayısındaki farklılık, bu iki problem arasındaki temel en büyük farktır. DJSSP'de işlerin uygun bir tesise atanması problemin daha iyi bir çözümler elde etmesini sağlamaktadır. Böylelikle uygun bir tesise atanması kararı büyük önem arz etmektedir. Atama kararı verilirken her işin, işlendiği makinedeki işlem süresi göz önüne alınır. İşlemler makinelere özgür bir şekilde rastgele atandığında aynı makineye birden fazla uzun işlem süresine sahip iş atanmış olabilir, bu durum üretim süresini önemli ölçüde artırır. Her makine

için makinelerin işlem süresi, tesise atanan işler için kullanıldığında aynı makinedeki işlemler göz ardı edilebilir. Bu durum da iş ataması için uygun bir yol olmayacaktır. Bu durumlar göz önüne alındığında, tez çalışmasında, her bir makinenin iş yükü hem aynı makinelerin işlem süresini hem de işler için makine sıralamasını dikkate almak amacıyla bir tesis iş atama kuralı kullanılmıştır. İş tesis atamasında kullanılan her bir makinenin sahip olduğu iş yükü, bu tez çalışmasında iş yükü kuralı olarak tanımlanmıştır. İş yükü kuralı Denklem 5.51 ile iş yükünü hesaplamak için her bir makinenin iş yükü ayrı ayrı değerlendirilir. Çalışmasında iş yükünden bahseden M. A. ŞAHMAN(Şahman, 2021), aşağıdaki formül ile makinelere atanacak olan operasyonların daha verimli hale getirilmesini sağlamıştır.

$$\text{İş Yükü (j, m)} = \left(\sum_{k \in PM_{j,m}} PT_{j,k} \right) + PT_{j,m} \ , m \in M, j \in J \qquad (5.51)$$

m: Makine indeksi

M: Makinelerin kümesi

j: İş indeksi

J: İşlerin kümesi

k: Mevcut makinenin indeksi

PM: Önceki makinelerin indeksi

PT: Önceki makinelerin kümesi

Çizelge 5.1 Verilen Örneğin İşlem Süreleri ve Rotası

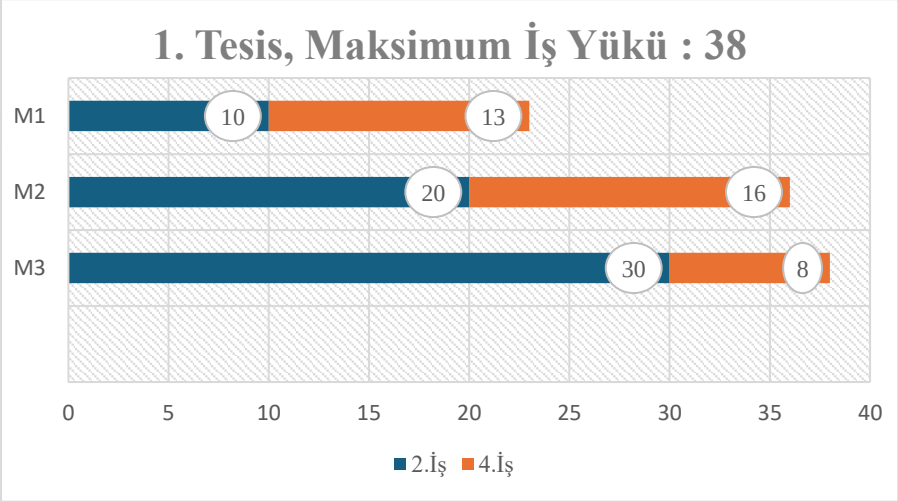
İş	**Makine**			**İşlem rotası**
	1	2	3	
1	6	7	3	(3,2,1)
2	10	10	10	(1,2,3)
3	8	9	1	(2,1,3)
4	5	3	8	(3,1,2)
5	5	4	3	(1,3,2)

Çizelge 5.2 Verilen Örneğin İş Yükü

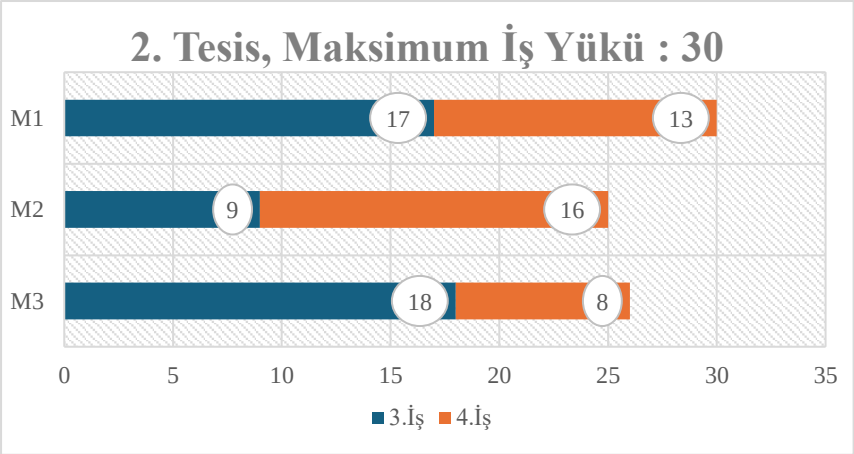
İş	**Makine**			**Toplam Zaman**	**Sıralama**
	1	**2**	**3**		
1	16	10	3	29	4
2	10	20	30	60	1
3	17	9	18	44	2
4	13	16	8	37	3
5	5	12	8	25	5

DJSP'deki iş yükü kuralını tanımlamak ve daha iyi açıklamak için, f = 2, n = 5 ve m = 3 (f = tesis sayısı, n = iş sayısı ve m = makine sayısı) olan sayısal örnek verilmiştir. Verilen örneğin işlem süreleri ve rotaları Çizelge 5.1'de verilmiştir. İşlerin iş yükü Çizelge 5.2'de verilmiştir.

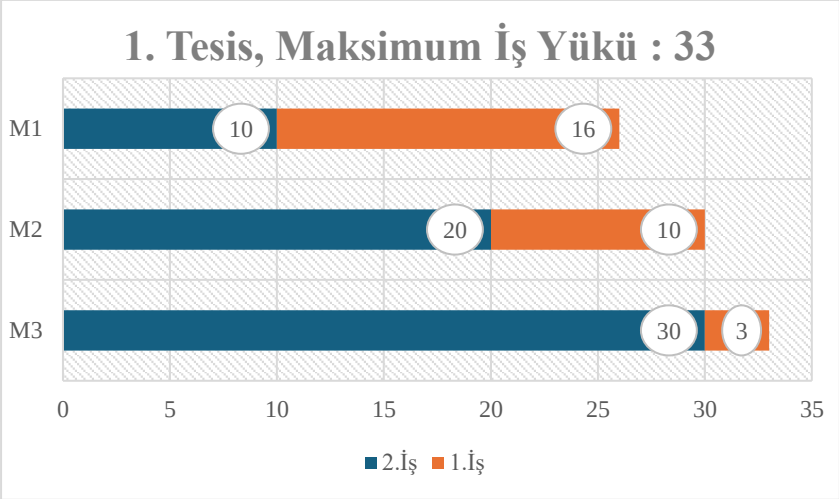
İşlerin toplam iş yükleri Denklem 5.51 kullanılarak hesaplanır, işler azalan düzende {2, 3, 4, 1, 5} şeklinde sıralanır ve Çizelge 5.2'te listelenir. İlk adımda, en yüksek toplam iş yüküne sahip iki iş (2 ve 3) sırasıyla tesis 1 ve 2'ye atanır. Böylece 1. tesiste 1, 2 ve 3 numaralı makinelerin iş yükleri sırasıyla 10, 20 ve 30 olur. Aynı şekilde 2. tesiste 1, 2 ve 3 numaralı makinelerin iş yükleri sırasıyla 17, 9 ve 18 olur. Ayrıca 1. tesisin ve 2. tesisin maksimum iş yükleri sırasıyla 30 ve 18 olur. Üçüncü en büyük sıralama değerine sahip 4. iş tesise atanmak üzere seçilir. 4. iş, tesislerin makine iş yükleriyle tek tek değerlendirilecek ve maksimum iş yükünün minimumuna sahip tesise atanacaktır. 4. işin 1. tesise atandığını varsayalım, bu durumda Şekil 5.1'de görüldüğü gibi 1, 2 ve 3 numaralı makinelerin iş yükleri sırasıyla 23, 36 ve 38 olacaktır.



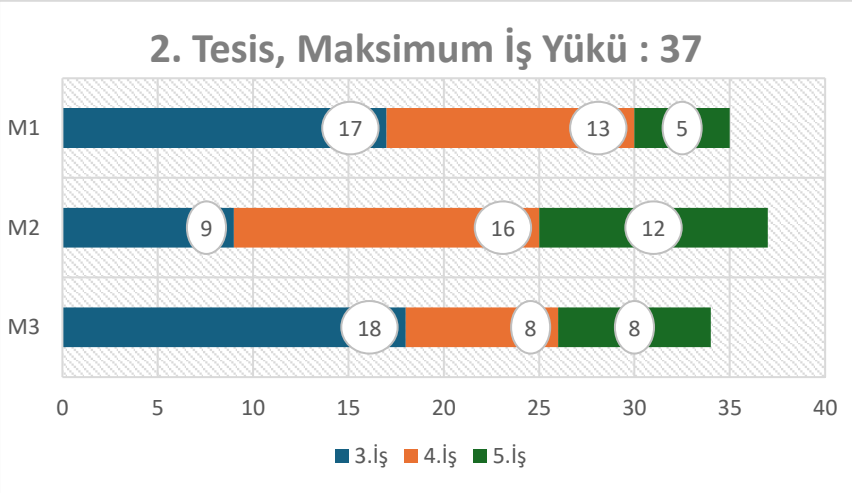
Şekil 5.1 1.Tesise 4. İşin atanması durumu



Şekil 5.2 2.Tesise 4. İşin atanması durumu



Şekil 5.3 1.Tesise 1. İşin atanması durumu



Şekil 5.4 2.Tesise 5. İşin atanması durumu

Benzer şekilde, 4. iş 2. tesise atanırsa Şekil 5.2'de görüldüğü gibi 1, 2 ve 3 numaralı makinelerin iş yükleri sırasıyla 30, 25 ve 26 olacaktır. 1. tesisin ve 2. tesisin maksimum iş yükü sırasıyla 38 ve 30 olur, böylece 4. iş, tesisteki maksimum iş yükünün en azına sahip olan 2.

tesise atanır. Bu süreç son iş, iş yükü kuralına göre uygun tesise atanana kadar devam eder. Son olarak Şekil 5.3 ve Şekil 5.4'te görüldüğü gibi, 1 ve 2 numaralı işler 1. tesise ve 3, 4, 5 numaralı işler 2. tesise atanır.

Bu tez çalışmasında, iş-tesis ataması bir vektör olarak kodlanmıştır. Örneğin, tesis-iş ataması {1, 1, 2, 2, 2} olarak verilirse, 1 ve 2 numaralı işlerin 1. tesise, 3, 4 ve 5 numaralı işlerin ise 2. tesise atanacağını gösterilmiştir.

5.3. Kodlama Şemaları

Kodlama şemaları, çözülmesi hedeflenen bir problemin, özellikle de optimizasyon ve çizelgeleme gibi ayrık yapılı problemlerin, algoritmalar tarafından işlenebilir bir formata dönüştürülmesini sağlayan sistematik yapılardır. Bu şemalar, problemin temel bileşenlerini (örneğin işler, makineler, işlemler veya kaynaklar) belirli bir veri yapısı içerisinde temsil eder. Böylece metasezgisel algoritmalar gibi çözümleyiciler, problemin farklı çözüm adaylarını oluşturabilir, değerlendirebilir ve karşılaştırabilir hale gelir.

Her problem türüne özgü farklı kodlama şemaları geliştirilmiştir; örneğin iş atölyesi çizelgeleme problemlerinde işlem sıralarını gösteren permütasyon tabanlı şemalar sıkça kullanılırken, genetik algoritmalarda kromozom yapısına uygun ikili ya da gerçek sayılı kodlamalar tercih edilebilir. Kodlama şemasının doğru seçilmesi, hem algoritmanın çözüm uzayını etkili bir şekilde keşfetmesini sağlar hem de yerel optimumlara sıkışma gibi problemleri azaltır.

Gerçek değer serilerinin JSSP'nin işlem sırasına dönüştürülmesi üç aşamada gerçekleştirilir. İlk aşamada reel değerler artan şekilde sıralanır. İkinci aşamada reel değerlerin sıralaması tamsayı serileri şeklinde artan şekilde ifade edilir. İlk iki aşamadaki işlem adımları her kodlama şeması için aynıdır ancak JSSP'nin işlem sırasının bulunduğu son aşama, Random Key (RK), Smallest Position Value (SPV) ve Ranked-Over Value (ROV) kodlama şemaları yaklaşımı için farklıdır. Bu yaklaşımlar aşağıdaki alt başlıklarda verilmiştir.

5.3.1. Random Key (RK) Kodlama Şeması

Bean (Bean, 1994) tarafından önerilen ve JSSP'de sürekli arama alanını ayrık arama alanına dönüştürmek için kullanılabilecek rastgele anahtar kodlama şemasıdır.

Çizelge 5.3 RK Kodlama Şeması İşlem Sıralaması

Gerçek Değerler	0,35	0,92	0,88	0,55	0,73	0,2
Tamsayı Değerleri	2	6	5	3	4	1
İş Endeksleri	1	1	2	2	1	2
İşlem Sırası	$O_{1,1}$	$O_{1,2}$	$O_{2,1}$	$O_{2,2}$	$O_{1,3}$	$O_{2,3}$

$$\vec{O} = (\phi k \bmod n) + 1, \quad k = 1, 2, \ldots, n \times m \tag{5.52}$$

Burada, $\vec{O}$ işlem sırası, ϕk tam sayı serisinin k'inci değeri ve n, JSSP'nin iş numarasıdır. Çizelge 5.3'te gösterildiği gibi 2 işimiz ve her işin 3 işlemi var olduğunu düşünelim, $n \times m$ gerçek değerleri sırasıyla $\{0.35, 0.92, 0.88, 0.55, 0.73, 0.2\}$ olsun. Tamsayı serisini elde etmek için gerçek değerler artan düzende sıralanır; yani 0,2 en küçük değerdir, 1'e kadar sıralanır, Tamsayı serisi {2,6,5,3,4,1} olarak belirlenir. Tamsayı serisindeki ilk değer 2'dir ve Denklem 5.52'e göre iş indeksi $(2 \bmod 2) + 1 = 1$ şeklinde hesaplanır. Benzer şekilde hesaplanarak birinci, ikinci ve beşinci tamsayı değerleri ($O_{1,1}$, $O_{1,2}$, $O_{1,3}$) işlemleri olarak belirlenir. Tam sayı serisinin geri kalanı ikinci işin işlemleri ($O_{2,1}$, $O_{2,2}$, $O_{2,3}$) olarak belirlenir. RK ile operasyon sırası doğru bir şekilde böyle sıralandırılır.

5.3.2. Smallest Position Value (SPV) Kodlama Şeması

Taşgetiren ve ark. (Tasgetiren ve ark., 2006) en küçük konum değeri kodlama şemasını (SPV) önermiştir. SPV kodlama şemasının ilk iki aşaması RK ile aynıdır fakat üçüncü aşamasında JSSP'nin işlem sırası Denklem 5.53 ile hesaplanmaktadır.

Çizelge 5.4 SPV Kodlama Şeması İşlem Sıralaması

Gerçek Değerler	0,35	0,92	0,88	0,55	0,73	0,2
Tamsayı Değerler	2	6	5	3	4	1
İş Endeksleri	1	2	2	1	2	1
İşlem Sırası	$O_{1,1}$	$O_{2,1}$	$O_{2,2}$	$O_{1,2}$	$O_{2,3}$	$O_{1,3}$

$$\vec{O} = floor\left(\frac{\phi k - 1}{m}\right) + 1, k = 1, 2, \ldots, n \times m \tag{5.53}$$

Burada $O^{\rightarrow}$ işlem sırası, ϕk tamsayı serisinin k'inci değeri ve m JSSP'nin makine numarasıdır. Kodlama şemasının iki aşaması RK ile aynıdır. Tamsayı serileri {2,6,5,3,4,1} olarak belirlenir. Denklem 5.53, Çizelge 5.4'te verilen iş indekslerini elde etmek için kullanılmaktadır. Hesaplamalara göre birinci, dördüncü ve altıncı tam sayı değerleri ilk işin ($O_{1,1}$, $O_{1,2}$, $O_{1,3}$) işlemleri olarak belirlenir ve benzer şekilde, geri kalan tamsayı değerleri ikinci işin ($O_{2,1}$, $O_{2,2}$, $O_{2,3}$) işlemleri olarak belirlenir.

5.3.3 Ranked-Over Value (ROV) Kodlama Şeması

Farklı bir kodlama şeması olarak, Liu ve diğerleri (Liu ve ark., 2020) tarafından sıralanmış bir değer kodlama şeması (ROV) önerilmektedir Bu yöntem, diğer kodlama şemalarından farklı olarak işlem sırasını belirlemek için bir iş indeksi modeli kullanır. Kodlama şemasının iki aşaması RK ve SPV ile aynıdır. Üçüncü aşamada bu kodlama şeması, başlatma adımında belirlenen ve metasezgisel algoritmaların süreçlerine göre değişmeyen, yani değişmeyen iş indeksi modelinden yararlanılır. Başlatma adımında iş indeksleri desenine, artan sıralı tamsayı konumlarına {1,2,3,4,5,6} göre {1,2,2,1,2,1} olarak karar verildiğini varsayalım. Metasezgisel algoritma sürecinde yapay etmen değerinin konumu güncellendiğinde, bu gerçek değerli konumlar sıralanır ve tamsayı serileri kurulur. Daha sonra kurulan tamsayı serileri iş indeksleri örüntüsüne göre eşleştirilir yani Çizelge 5.5'te verilen {2,6,5,3,4,1} tamsayı serileri {2,1,2,2,1,1} işlem sırası ile eşleştirilir. Bu sayede her farklı tamsayı serisi uygun şekilde kodlanabilmektedir.

Çizelge 5.5 ROV Kodlama Şeması İşlem Sıralaması

Tamsayı Konumları	1	2	3	4	5	6
İş Endeksleri Modeli	1	2	2	1	2	1
İşlem Sırası Modeli	$O_{1,1}$	$O_{2,1}$	$O_{2,2}$	$O_{1,2}$	$O_{2,3}$	$O_{1,3}$
Gerçek Değerler	0,35	0,92	0,88	0,55	0,73	0,2
Tamsayı Değerleri	2	6	5	3	4	1
İş Endeksleri	2	1	2	2	1	1
İşlem Sırası	$O_{2,1}$	$O_{1,1}$	$O_{2,2}$	$O_{2,3}$	$O_{1,2}$	$O_{1,3}$

6. DENEYSEL ÇALIŞMALAR

Karmaşık optimizasyon problemi olarak ele alınan DJSSP, çözümü için diğer algoritmalara kıyasla daha az zaman harcadığı için metasezgisel algoritmalar kullanılmaktadır. Bu çalışmada da DJSSP'yi çözmek için metasezgisel algoritmalar önerilmiştir. Önerilen algoritmalar 3 farklı kodlama şeması kullanılarak sürekli bir yapıdan ayrık bir yapıya dönüştürülmüştür. Deneyler, 16 GB RAM ve Intel® Alder Lake Core™ i7 CPU' ya sahip 3,50 GHz dizüstü sistemi üzerinde Matlab programı kullanılarak gerçekleştirilmiştir. Algoritmalara uygulanacak olan ortak parametreler $pop = 40, maxrun = 30, max\_fes = D * 1000$ olarak ayarlanmıştır. Burada popülasyonu değişkeni 'pop', maksimum çalıştırma değişkeni 'maxrun', fonksiyon değerlendirme (function evaluations) değişkeni 'max_fes' olarak belirtilmiştir.

JSSP için n iş ve m makineyi temsil etmektedir ve her bir metasezgisel algoritmanın yapay etmenlerinin boyut uzunluğu $n \times m$'dir. Yapay ajanların konumları gerçek sayılarla temsil edilmektedir. Bu $n \times m$ gerçek sayılar, kodlama şeması yaklaşımları kullanılarak JSSP'nin permütasyon dizisini taklit etmek için tam sayı serisine dönüştürülür. Dönüştürülen tamsayı serileri işlem dizileri olarak temsil edilir.

Çizelge 6.1 Algoritmaların Parametreleri

Parametreler	Algoritmalar									
	PSO	TSA	ABC	LFD	JAYA	RFO	DSA	GWO	WOA	AAA
Popülasyon(N)	40	40	40	40	40	40	40	40	40	40
Boyut(D)	n x m	n x m	n x m	n x m	n x m	n x m	n x m	n x m	n x m	n x m
dmin, dmax	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5	-5, 5
max_fes	D*1000	D*1000	D*1000	D*1000	D*1000	D*1000	D*1000	D*1000	D*1000	D*1000
c1	2	-	-	-	-	0,18	-	-	-	-
c2	2	-	-	-	-	0,82	-	-	-	-
maxLimit	-	-	40	-	-	-	-	-	-	-
r_1	(0,1)	[-1,1]	-	-	-	-	-	(0,1)	-	-
r_2	(0,1)	-	-	-	-	-	-	(0,1)	-	-
$\vec{a}$	-	-	-	-	-	-	-	[0,2]	[0,2]	-
L	-	N*0,1	-	-	-	-	-	-	-	-
U	-	N*0,25	-	-	-	-	-	-	-	-
ST	-	[0,1]	-	-	-		-	-	-	-
CSV	-	-	-	0.5	-	-	-	-	-	-
K	-	-	-	-	-	-	-	-	-	2
Le	-	-	-	-	-	-	-	-	-	0,3
Ap	-	-	-	-	-	-	-	-	-	0,5

Veri setinde bulunan karşılaştırma problemlerine 3 farklı kodlama şeması ve 10 farklı optimizasyon algoritması uygulanmıştır. Çizelge 6.2, 6.3, 6.4'te uygulanan optimizasyon algoritmalarının sonuçları belirtilmiştir ve tamamlama süresi diğerlerine göre en minimum olan değerler **bold** renkte belirtilmiştir.

Çizelge 6.2 RK Kodlama Şeması Uygulama Sonuçları

	PSO			TSA			ABC			LFD			JAYA			RFO			DSA			GWO			WOA			AAA		
	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
'ft06'	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	48	47	47	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**
'ft10'	771	682	**658**	804	**668**	**658**	771	**668**	**658**	797	683	658	821	687	**658**	819	681	**658**	771	**668**	**658**	771	**668**	**658**	770	**668**	**658**	**746**	**668**	**658**
'ft20'	845	702	540	838	694	541	816	678	531	824	678	534	860	679	555	825	691	539	815	672	526	781	646	515	795	653	525	**723**	**632**	**485**
'abz5'	**1009**	**957**	**880**	1069	**957**	**880**	1050	**957**	**880**	1017	957	880	1049	**957**	**880**	1077	**957**	**880**	**1009**	**957**	**880**	1049	**957**	**880**	**1009**	**957**	**880**	**1009**	**957**	**880**
'abz6'	843	**751**	**774**	848	**751**	**774**	848	**751**	**774**	848	751	774	848	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**	**842**	**751**	**774**
'abz7'	589	541	477	672	581	526	650	562	517	607	554	490	655	573	517	649	579	531	567	530	476	668	574	510	563	512	472	**544**	**498**	**458**
'abz8'	604	525	470	703	585	514	684	575	516	632	539	487	687	567	506	711	596	528	591	519	465	691	592	518	587	521	465	**562**	**502**	**459**
'abz9'	629	530	491	688	587	531	684	576	526	659	551	494	686	580	533	708	597	543	604	512	487	700	591	535	607	519	487	**575**	**493**	**483**
'la01'	462	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	459	447	413	462	**447**	**413**	462	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**
'la02'	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	477	405	394	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**
'la04'	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	430	398	369	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**
'la05'	420	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**	420	380	380	420	**380**	**380**	420	**380**	**380**	420	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**
'la06'	588	454	**413**	581	455	**413**	581	455	**413**	578	456	413	587	455	**413**	563	467	**413**	570	**453**	**413**	574	472	**413**	573	**453**	**413**	**551**	**453**	**413**
'la07'	557	462	**397**	566	464	403	565	458	**397**	542	459	397	569	458	**397**	561	469	**397**	557	458	**397**	560	**457**	**397**	552	458	**397**	**521**	**457**	**397**
'la08'	567	**437**	**400**	576	441	**400**	567	**437**	**400**	571	437	400	576	**437**	**400**	575	462	**400**	559	**437**	**400**	537	441	**400**	548	**437**	**400**	**528**	**437**	**400**
'la09'	**553**	**496**	**439**	572	**496**	**439**	559	**496**	**439**	562	496	439	569	**496**	443	572	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**
'la10'	540	**443**	**453**	564	**443**	**453**	556	**443**	**453**	545	443	453	573	**443**	**453**	567	**443**	**453**	**521**	**443**	**453**	557	**443**	**453**	**521**	**443**	**453**	**521**	**443**	**453**
'la11'	718	544	478	744	576	499	730	549	476	741	552	478	748	561	494	748	569	493	711	552	473	731	**524**	480	715	549	462	**676**	**524**	**457**
'la12'	628	489	416	638	496	431	**623**	469	416	623	491	416	625	500	416	653	500	433	**623**	474	**408**	**623**	489	**408**	**623**	477	**408**	**623**	**458**	**408**
'la13'	674	555	481	667	561	477	678	538	471	676	553	479	714	574	485	708	562	486	**657**	537	469	674	550	473	**657**	**522**	464	**657**	**522**	**458**
'la14'	693	**512**	**465**	723	537	482	693	535	476	691	537	465	730	529	482	724	545	475	**691**	537	**465**	726	**512**	480	**691**	517	**465**	**691**	**512**	**465**
'la15'	760	567	533	780	593	538	762	568	518	759	561	530	791	567	523	793	601	541	745	556	504	755	558	507	740	545	507	**701**	**525**	**490**
'la16'	764	**717**	**717**	761	**717**	**717**	761	**717**	**717**	761	717	717	775	**717**	**717**	763	**717**	**717**	761	**717**	**717**	758	**717**	**717**	758	**717**	**717**	**754**	**717**	**717**
'la17'	663	**646**	**646**	663	**646**	**646**	673	**646**	**646**	658	646	646	690	**646**	**646**	691	**646**	**646**	663	**646**	**646**	669	**646**	**646**	658	**646**	**646**	**657**	**646**	**646**
'la18'	**718**	**663**	**663**	719	**663**	**663**	719	**663**	**663**	719	663	663	719	**663**	**663**	723	**663**	**663**	**718**	**663**	**663**	719	**663**	**663**	**718**	**663**	**663**	**718**	**663**	**663**
'la19'	731	**644**	**634**	723	**644**	**634**	741	**644**	641	730	644	634	747	**644**	**634**	737	**644**	**634**	732	**644**	**634**	**713**	**644**	**634**	718	**644**	**634**	**713**	**644**	**634**
'la20'	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	761	756	756	765	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**
'la21'	881	770	**719**	974	802	**719**	948	801	**719**	940	782	719	951	801	723	978	811	**719**	881	775	**719**	965	803	**719**	864	770	**719**	**829**	**749**	**719**
'la22'	781	665	**619**	858	697	645	823	659	**619**	808	678	619	840	676	631	853	686	637	761	655	**619**	815	685	639	754	656	**619**	**732**	**634**	**619**
'la23'	830	737	**714**	917	793	**714**	867	764	**714**	865	747	716	906	775	**714**	921	792	**714**	830	**716**	**714**	915	763	716	827	**716**	716	**783**	**716**	**714**
'la24'	844	738	**704**	892	751	**704**	868	738	**704**	881	749	704	889	754	**704**	899	773	**704**	847	738	**704**	920	761	**704**	808	738	**704**	**783**	**725**	**704**
'la25'	817	728	**723**	889	**723**	**723**	839	**723**	**723**	842	723	723	844	736	**723**	879	737	**723**	816	727	**723**	844	737	**723**	811	**723**	**723**	**780**	**723**	**723**
'la26'	1028	788	768	1127	888	822	1071	847	807	1090	804	795	1117	875	809	1125	901	821	995	784	767	1124	884	803	997	780	768	**920**	**767**	**750**
'la27'	986	897	754	1097	945	809	1040	899	766	1059	920	790	1114	952	805	1117	951	813	964	844	754	1083	958	801	973	833	752	**915**	**788**	**712**
'la28'	1041	820	799	1105	904	829	1087	865	808	1069	859	816	1128	881	818	1129	919	821	990	815	794	1081	902	818	990	798	**776**	**925**	**781**	**776**
'la29'	905	786	729	1029	865	764	1015	812	738	966	814	745	1010	856	748	1058	824	774	905	775	730	1014	834	764	913	775	728	**859**	**759**	**723**
'la30'	1043	853	**772**	1109	925	838	1081	885	815	1068	877	821	1120	911	834	1136	933	848	1010	840	**772**	1095	929	832	992	839	**772**	**937**	**786**	**772**
'orb01'	862	761	**722**	912	754	**722**	875	754	**722**	875	754	722	901	765	**722**	917	760	**722**	856	755	**722**	888	762	**722**	831	**745**	**722**	**829**	**745**	**722**
'orb02'	706	**672**	**663**	740	**672**	**663**	708	**672**	**663**	706	672	663	709	**672**	**663**	732	**672**	**663**	706	**672**	**663**	**700**	**672**	**663**	706	**672**	**663**	**700**	**672**	**663**
'orb03'	870	770	673	903	778	673	861	743	**661**	852	723	662	873	764	668	911	766	677	838	734	**661**	875	743	**661**	842	729	**661**	**781**	**705**	**661**
'orb04'	800	**787**	**789**	832	**787**	**789**	798	**787**	**789**	819	787	789	820	**787**	**789**	820	**787**	**789**	800	**787**	**789**	808	**787**	**789**	793	**787**	**789**	**789**	**787**	**789**
'orb05'	756	660	**610**	736	653	**610**	734	644	**610**	756	639	610	769	658	**610**	764	639	**610**	715	650	**610**	**696**	637	**610**	709	637	**610**	**696**	**633**	**610**
'orb06'	843	732	**715**	866	757	**715**	842	736	**715**	843	737	715	873	759	**715**	873	751	**715**	835	**715**	**715**	837	720	**715**	822	**715**	**715**	**793**	**715**	**715**
'orb07'	**326**	**304**	**277**	338	**304**	**277**	336	**304**	**277**	335	304	277	336	**304**	**277**	345	**304**	280	**326**	**304**	**277**	**326**	**304**	**277**	**326**	**304**	**277**	**326**	**304**	**277**
'orb08'	797	640	**593**	804	664	**593**	771	633	**593**	780	654	593	804	653	594	802	665	**593**	771	633	**593**	738	660	**593**	**727**	628	**593**	**727**	**622**	**593**
'orb09'	755	**695**	**659**	783	**695**	**659**	744	**695**	**659**	775	695	659	786	**695**	**659**	783	**695**	**659**	752	**695**	**659**	770	**695**	**659**	752	**695**	**659**	**739**	**695**	**659**
'orb10'	786	**722**	**686**	**823**	**722**	**686**	851	**722**	**686**	863	724	**686**	821	**722**	**686**	843	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**

Çizelge 6.3 ROV Kodlama Şeması Uygulama Sonuçları

	PSO			TSA			ABC			LFD			JAYA			RFO			DSA			GWO			WOA			AAA		
	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
'ft06'	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**
'ft10'	780	681	**658**	820	685	**658**	796	**668**	**658**	795	672	**658**	810	672	**658**	821	681	**658**	769	**668**	**658**	770	**668**	**658**	769	**668**	**658**	**746**	**668**	**658**
'ft20'	841	689	522	841	693	548	816	666	527	821	662	532	816	685	539	848	692	531	788	662	519	791	662	534	791	645	508	**728**	**632**	**481**
'abz5'	1042	**957**	**880**	1063	**957**	**880**	1056	**957**	**880**	1067	**957**	**880**	1064	**957**	**880**	1022	**957**	**880**	**1009**	**957**	**880**	1055	**957**	**880**	1021	**957**	**880**	**1009**	**957**	**880**
'abz6'	846	**751**	**774**	848	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**	**842**	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**	846	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**
'abz7'	680	580	524	666	582	526	655	554	512	658	580	513	671	580	525	673	573	527	628	554	502	669	588	518	625	554	487	**538**	**493**	**458**
'abz8'	689	575	524	702	592	533	676	570	500	675	567	516	704	590	518	696	581	520	665	565	501	701	588	508	626	548	482	**566**	**500**	**459**
'abz9'	695	581	531	699	588	538	674	572	522	683	574	521	703	589	536	691	575	534	655	560	512	702	590	530	643	552	501	**576**	**495**	**483**
'la01'	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	462	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**
'la02'	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**
'la04'	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**
'la05'	420	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**
'la06'	592	473	**413**	600	473	**413**	578	**453**	**413**	584	455	**413**	581	455	**413**	592	**453**	**413**	572	**453**	**413**	587	**453**	**413**	572	**453**	**413**	**551**	**453**	**413**
'la07'	560	467	**397**	560	467	404	561	**457**	**397**	560	458	**397**	560	461	**397**	568	467	404	556	**457**	**397**	527	**457**	**397**	555	**457**	**397**	**521**	**457**	**397**
'la08'	564	446	**400**	583	445	**400**	543	441	**400**	563	441	**400**	584	441	**400**	585	446	**400**	545	440	**400**	551	**437**	**400**	554	**437**	**400**	**532**	**437**	**400**
'la09'	**553**	**496**	**439**	566	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**	573	**496**	**439**	568	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**	**553**	**496**	**439**
'la10'	546	**443**	**453**	568	**443**	**453**	551	**443**	**453**	545	**443**	**453**	561	**443**	**453**	563	**443**	**453**	546	**443**	**453**	538	**443**	**453**	540	**443**	**453**	**521**	**443**	**453**
'la11'	739	549	488	740	557	495	720	555	470	723	554	488	744	557	479	742	579	490	719	554	468	709	534	486	677	543	462	**674**	**524**	**457**
'la12'	638	486	**408**	626	488	424	624	481	411	**623**	479	409	636	498	416	639	499	429	**623**	477	**408**	**623**	478	**408**	**623**	472	**408**	**623**	**458**	**408**
'la13'	714	566	471	710	559	477	679	547	470	669	550	481	703	563	484	703	548	488	672	543	473	676	551	475	**657**	**522**	470	**657**	**522**	**458**
'la14'	717	544	**465**	719	542	473	**691**	517	**465**	692	546	**465**	693	537	481	718	549	482	**691**	528	471	**691**	546	**465**	**691**	517	**465**	**691**	**512**	**465**
'la15'	751	577	530	784	589	539	773	561	503	774	587	517	785	592	517	790	591	530	753	574	506	764	554	506	727	546	**490**	**691**	**522**	**490**
'la16'	761	**717**	**717**	761	**717**	**717**	758	**717**	**717**	761	**717**	**717**	775	**717**	**717**	761	**717**	**717**	758	**717**	**717**	758	**717**	**717**	761	**717**	**717**	**754**	**717**	**717**
'la17'	672	**646**	**646**	669	**646**	**646**	**657**	**646**	**646**	672	**646**	**646**	658	**646**	**646**	690	**646**	**646**	**657**	**646**	**646**	663	**646**	**646**	**657**	**646**	**646**	**657**	**646**	**646**
'la18'	719	**663**	**663**	738	**663**	**663**	719	**663**	**663**	723	**663**	**663**	730	**663**	**663**	720	**663**	**663**	**718**	**663**	**663**	719	**663**	**663**	**718**	**663**	**663**	**718**	**663**	**663**
'la19'	728	**644**	**634**	741	**644**	**634**	714	**644**	**634**	731	**644**	**634**	738	647	641	746	**644**	**634**	729	**644**	**634**	731	**644**	**634**	**713**	**644**	**634**	**713**	**644**	**634**
'la20'	**761**	**756**	**756**	762	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**
'la21'	955	808	**719**	971	798	**719**	929	782	**719**	954	797	**719**	973	808	**719**	961	784	**719**	933	776	**719**	946	807	**719**	870	**749**	**719**	**823**	**749**	**719**
'la22'	833	697	649	835	705	645	797	663	628	813	692	644	830	686	644	841	695	649	812	667	**619**	828	705	646	768	675	**619**	**729**	**634**	**619**
'la23'	899	795	716	890	771	**714**	885	770	**714**	902	765	**714**	910	798	716	894	789	**714**	850	755	**714**	865	734	**714**	842	741	**714**	**782**	**716**	**714**
'la24'	902	748	**704**	875	762	**704**	880	738	**704**	888	760	**704**	902	746	**704**	906	763	**704**	866	738	**704**	888	749	**704**	850	738	**704**	**781**	**725**	**704**
'la25'	878	728	**723**	891	730	**723**	856	**723**	**723**	850	728	**723**	879	734	**723**	864	736	**723**	842	**723**	**723**	830	735	**723**	817	**723**	**723**	**777**	**723**	**723**
'la26'	1112	880	825	1131	872	833	1088	864	810	1100	860	812	1107	863	827	1125	869	835	1052	843	786	1132	860	828	1029	837	757	**929**	**755**	**750**
'la27'	1129	959	797	1125	925	813	1087	923	768	1055	927	782	1121	948	806	1080	918	795	1030	855	762	1102	931	805	1003	856	745	**916**	**783**	**712**
'la28'	1095	889	826	1109	897	818	1086	874	812	1063	871	812	1108	902	804	1102	895	818	1033	859	800	1105	886	818	1024	848	**776**	**901**	**781**	**776**
'la29'	1011	840	772	1038	871	760	985	833	763	1009	812	760	1029	861	778	1035	855	772	973	821	749	1016	855	760	942	807	734	**853**	**759**	**723**
'la30'	1132	917	817	1095	911	822	1076	893	833	1080	925	825	1116	939	855	1139	930	841	1024	888	815	1050	921	819	1020	865	783	**942**	**786**	**772**
'orb01'	897	763	**722**	887	761	**722**	894	754	**722**	879	754	**722**	888	766	**722**	891	761	**722**	876	748	**722**	863	754	**722**	838	**745**	**722**	**829**	**745**	**722**
'orb02'	740	**672**	**663**	713	**672**	**663**	708	**672**	**663**	735	**672**	**663**	738	**672**	**663**	735	**672**	**663**	703	**672**	**663**	713	**672**	**663**	**700**	**672**	**663**	**700**	**672**	**663**
'orb03'	865	766	**661**	896	746	662	884	727	**661**	876	749	**661**	885	773	666	894	748	662	850	734	**661**	845	743	**661**	814	733	**661**	**786**	**705**	**661**
'orb04'	796	**787**	**789**	807	**787**	**789**	794	**787**	**789**	820	**787**	**789**	824	**787**	**789**	845	**787**	**789**	794	**787**	**789**	804	**787**	**789**	792	**787**	**789**	**789**	**787**	**789**
'orb05'	755	637	**610**	775	639	**610**	736	644	**610**	747	660	**610**	762	664	**610**	754	660	**610**	729	639	**610**	741	639	**610**	732	635	**610**	**696**	**633**	**610**
'orb06'	848	742	**715**	877	751	**715**	845	743	**715**	845	744	**715**	865	733	**715**	876	751	**715**	842	730	**715**	838	**715**	**715**	818	719	**715**	**793**	**715**	**715**
'orb07'	340	**304**	**277**	333	**304**	**277**	337	**304**	**277**	340	**304**	**277**	348	**304**	**277**	338	**304**	**277**	333	**304**	**277**	342	**304**	**277**	327	**304**	**277**	**326**	**304**	**277**
'orb08'	792	657	594	779	665	**593**	768	641	**593**	769	638	**593**	794	665	**593**	795	660	**593**	768	633	**593**	749	649	**593**	749	633	**593**	**726**	**622**	**593**
'orb09'	760	**695**	**659**	750	**695**	**659**	755	**695**	**659**	766	**695**	**659**	773	**695**	**659**	766	**695**	**659**	752	**695**	**659**	748	**695**	**659**	753	**695**	**659**	**739**	**695**	**659**
'orb10'	809	**722**	**686**	791	**722**	**686**	**777**	**722**	**686**	833	**722**	**686**	834	**722**	**686**	828	**722**	**686**	**777**	**722**	**686**	780	**722**	**686**	**777**	**722**	**686**	777	**722**	**686**

Çizelge 6.4 SPV Kodlama Şeması Uygulama Sonuçları

	PSO			TSA			ABC			LFD			JAYA			RFO			DSA			GWO			WOA			AAA		
	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4	2	3	4
'ft06'	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**	**48**	**47**	**47**
'ft10'	850	708	**658**	812	**668**	**658**	800	698	**658**	769	**668**	**658**	852	697	665	831	697	**658**	766	**668**	**658**	771	**668**	**658**	771	**668**	**658**	**746**	**668**	**658**
'ft20'	901	690	584	838	685	542	812	678	550	793	653	522	861	701	566	865	711	569	775	649	523	765	669	519	776	646	521	**714**	**632**	**485**
'abz5'	1142	1043	**880**	1049	**957**	**880**	1068	**957**	**880**	**1009**	**957**	**880**	1103	**957**	**880**	1093	**957**	**880**	**1009**	**957**	**880**	1016	**957**	**880**	1019	**957**	**880**	**1009**	**957**	**880**
'abz6'	877	787	**774**	**842**	**751**	**774**	852	**751**	**774**	848	**751**	**774**	865	**751**	**774**	853	**751**	**774**	**842**	**751**	**774**	848	**751**	**774**	848	**751**	**774**	**842**	**751**	**774**
'abz7'	775	662	597	651	577	507	652	569	522	641	562	505	697	587	542	686	604	544	613	535	483	613	546	506	600	537	486	**540**	**498**	**458**
'abz8'	791	669	589	683	587	518	691	587	532	642	574	504	734	611	544	732	623	539	647	533	491	644	549	496	639	545	483	**565**	**503**	**459**
'abz9'	787	652	585	685	591	532	686	582	543	673	563	506	719	603	552	722	606	550	633	549	500	648	554	505	647	536	502	**572**	**493**	**483**
'la01'	479	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	462	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**	**459**	**447**	**413**
'la02'	509	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**	**477**	**405**	**394**
'la04'	445	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**	**430**	**398**	**369**
'la05'	432	**380**	**380**	**419**	**380**	**380**	420	**380**	**380**	420	**380**	**380**	420	**380**	**380**	420	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**	**419**	**380**	**380**
'la06'	645	488	419	589	469	**413**	581	455	**413**	572	466	**413**	570	475	**413**	611	480	**413**	559	**453**	**413**	551	**453**	**413**	578	**453**	**413**	**550**	**453**	**413**
'la07'	626	507	433	565	459	**397**	560	462	403	550	**457**	**397**	593	467	404	596	479	408	535	**457**	**397**	548	459	**397**	553	**457**	**397**	**521**	**457**	**397**
'la08'	639	491	438	585	441	**400**	584	462	**400**	554	446	**400**	612	462	415	610	455	**400**	553	**437**	**400**	555	**437**	**400**	551	**437**	**400**	**524**	**437**	**400**
'la09'	639	521	471	558	**496**	**439**	**553**	**496**	441	**553**	**496**	**439**	594	500	**439**	591	500	**439**	**553**	**496**	**439**	**553**	**496**	441	**553**	**496**	**439**	**553**	**496**	**439**
'la10'	620	478	**453**	555	**443**	**453**	554	**443**	**453**	539	**443**	**453**	575	445	**453**	569	**443**	**453**	540	**443**	**453**	534	**443**	**453**	**521**	**443**	**453**	**521**	**443**	**453**
'la11'	795	663	517	745	577	482	736	553	470	711	545	470	783	587	510	772	594	493	711	532	462	719	545	462	697	**524**	462	**667**	**524**	**457**
'la12'	684	570	474	630	495	420	**623**	495	429	**623**	478	**408**	668	519	461	655	517	429	**623**	477	**408**	**623**	468	416	**623**	468	**408**	**623**	**458**	**408**
'la13'	812	633	515	697	557	481	702	560	488	669	533	474	732	581	505	725	597	506	669	531	459	666	**522**	**458**	659	**522**	470	**657**	**522**	**458**
'la14'	796	595	532	712	542	475	707	542	480	**691**	537	**465**	743	568	489	756	566	499	**691**	520	**465**	**691**	536	475	**691**	534	**465**	**691**	**512**	**465**
'la15'	866	671	593	791	583	533	773	595	545	754	580	517	827	618	555	809	609	557	753	558	507	753	564	492	737	542	506	**692**	**519**	**490**
'la16'	851	718	**717**	761	**717**	**717**	771	**717**	**717**	761	**717**	**717**	771	**717**	**717**	771	**717**	**717**	758	**717**	**717**	761	**717**	**717**	**754**	**717**	**717**	**754**	**717**	**717**
'la17'	746	**646**	**646**	670	**646**	**646**	690	**646**	**646**	**657**	**646**	**646**	687	**646**	**646**	704	**646**	**646**	**657**	**646**	**646**	**657**	**646**	**646**	**657**	**646**	**646**	**657**	**646**	**646**
'la18'	824	684	**663**	719	**663**	**663**	741	**663**	**663**	**718**	**663**	**663**	751	**663**	**663**	741	**663**	**663**	**718**	**663**	**663**	719	**663**	**663**	719	**663**	**663**	**718**	**663**	**663**
'la19'	798	680	642	728	**644**	**634**	731	646	641	725	**644**	**634**	758	670	**634**	764	658	642	**713**	**644**	**634**	718	**644**	642	718	**644**	**634**	**713**	**644**	**634**
'la20'	810	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	762	**756**	**756**	771	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**	**761**	**756**	**756**
'la21'	1088	854	783	950	799	**719**	972	809	723	928	775	**719**	1005	845	756	978	803	**719**	872	761	**719**	910	791	**719**	874	761	**719**	**835**	**749**	**719**
'la22'	989	755	704	843	713	**619**	842	710	653	780	677	**619**	871	709	680	868	722	653	777	660	**619**	769	653	**619**	775	648	**619**	**724**	**634**	**619**
'la23'	1055	883	793	906	787	717	912	799	721	853	753	**714**	977	787	725	958	808	733	824	735	**714**	824	755	**714**	854	740	**714**	**780**	**716**	**714**
'la24'	1004	853	753	892	765	**704**	890	770	**704**	829	738	**704**	912	794	708	953	787	710	829	**725**	**704**	829	738	**704**	845	738	**704**	**792**	**725**	**704**
'la25'	1002	807	794	875	725	**723**	890	742	**723**	821	**723**	**723**	890	757	**723**	913	736	**723**	813	**723**	**723**	813	**723**	**723**	819	**723**	**723**	**777**	**723**	**723**
'la26'	1228	1000	868	1106	887	818	1102	884	800	1055	822	794	1172	921	866	1189	927	848	1031	827	782	1057	822	768	1023	822	772	**918**	**764**	**750**
'la27'	1269	1074	906	1099	920	806	1070	938	814	1063	888	762	1149	976	828	1167	969	850	1010	860	764	1024	840	749	1044	859	747	**903**	**783**	**712**
'la28'	1245	1007	940	1120	899	823	1106	852	818	1039	847	794	1178	944	845	1166	930	847	1010	820	780	1040	847	792	995	820	786	**922**	**781**	**776**
'la29'	1195	963	886	1043	849	760	1020	840	769	985	812	748	1055	887	809	1078	881	803	944	794	729	967	803	734	942	788	729	**843**	**759**	**723**
'la30'	1303	1061	962	1114	927	824	1108	942	833	1060	883	790	1149	958	864	1124	972	866	1038	858	782	1050	878	815	1010	862	780	**932**	**799**	**772**
'orb01'	983	820	**722**	900	762	**722**	900	770	**722**	872	**745**	**722**	919	778	**722**	929	781	**722**	859	**745**	**722**	866	760	**722**	862	760	**722**	**829**	**745**	**722**
'orb02'	821	690	664	731	**672**	**663**	740	**672**	**663**	719	**672**	**663**	755	**672**	**663**	740	**672**	**663**	**700**	**672**	**663**	706	**672**	**663**	**700**	**672**	**663**	**700**	**672**	**663**
'orb03'	956	857	743	886	756	668	877	755	677	867	728	**661**	907	779	706	925	803	663	834	724	**661**	848	758	**661**	840	738	**661**	**781**	**705**	**661**
'orb04'	867	**787**	**789**	812	**787**	**789**	845	**787**	**789**	**789**	**787**	**789**	859	**787**	**789**	846	**787**	**789**	**789**	**787**	**789**	820	**787**	**789**	790	**787**	**789**	**789**	**787**	**789**
'orb05'	811	703	616	743	650	**610**	749	668	**610**	714	636	**610**	781	665	**610**	747	661	**610**	713	**633**	**610**	717	**633**	**610**	713	**633**	**610**	**696**	**633**	**610**
'orb06'	987	783	766	859	751	**715**	883	769	**715**	848	738	**715**	898	763	**715**	864	751	**715**	819	**715**	**715**	798	751	**715**	831	730	**715**	**791**	**715**	**715**
'orb07'	365	332	298	344	**304**	**277**	348	**304**	**277**	333	**304**	**277**	356	306	280	351	308	280	328	**304**	**277**	333	**304**	**277**	333	**304**	**277**	**326**	**304**	**277**
'orb08'	837	680	610	792	659	**593**	790	658	**593**	758	633	**593**	795	670	**593**	830	646	595	735	633	**593**	727	648	**593**	765	636	**593**	**726**	**622**	**593**
'orb09'	836	725	**659**	773	**695**	**659**	790	**695**	**659**	752	**695**	**659**	791	**695**	**659**	781	**695**	**659**	744	**695**	**659**	752	**695**	**659**	**739**	**695**	**659**	**739**	**695**	**659**
'orb10'	912	766	**686**	831	**722**	**686**	808	**722**	**686**	**777**	**722**	**686**	843	727	**686**	821	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**	**777**	**722**	**686**

Çizelge 6.5 2 Tesis Sayısına Göre Friedman Test Sonuçları

	RK										ROV										SPV									
	PSO	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**	**PSO**	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**	**PSO**	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**
'ft06'	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	5,65	**5,48**	**5,48**	**5,48**	**5,48**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'ft10'	3,97	8,08	5,08	6,17	8,13	8,38	3,43	6,97	3,77	**1,02**	7,33	7,13	5,13	6,15	7,03	7,82	3,13	6,83	3,37	**1,07**	9,8	5,92	5,72	5,28	9,12	6,72	2,55	3,53	5,22	**1,15**
'ft20'	8,15	6,27	3,78	5,35	8,53	8,82	4,75	5,1	3,22	**1,03**	8,02	7,23	3,8	6,4	7,2	7,45	3,07	7,07	3,77	**1**	9,68	7,27	5,6	5,6	9,1	7,18	3,07	2,27	4,03	**1,2**
'abz5'	2,8	8,15	6,42	6,38	6,75	9,33	2,48	7,73	3	**1,95**	7,73	6,63	4,67	5,82	7,45	7	2,78	6,55	5,17	**1,2**	9,9	5,97	5,25	5,27	8,93	6,78	3,35	2,88	5,52	**1,15**
'abz6'	3,57	8,03	5,77	6,62	6,42	7,72	3,08	7,88	3,33	**2,58**	7,73	6,4	5,1	6,23	6,55	5,93	3,15	7,1	4,73	**2,07**	9,97	5,07	5,82	5,68	9,03	5,87	2,93	3,22	5,92	**1,5**
'abz7'	3,83	8,57	6,9	5,42	7,47	7,33	2,65	9,23	2,53	**1,07**	8,8	6,8	4,2	6,5	7,42	7,72	2,57	7,05	2,95	**1**	9,97	7,08	5,3	5,28	8,95	7,4	2,95	2,67	4,4	**1**
'abz8'	3,97	9,03	7,18	5,08	6,88	8,2	2,73	8,52	2,33	**1,07**	8,45	7,25	4,52	6,07	7,52	7,47	2,65	7,13	2,95	**1**	10	6,97	5,37	5,42	9	7,42	2,97	3	3,87	**1**
'abz9'	3,87	8,15	6,02	6,1	6,9	9,42	2,57	8,35	2,63	**1**	8,65	7,62	4,35	5,95	7,08	7,37	2,68	7,32	2,98	**1**	10	7,2	5,52	5,18	8,82	7,25	2,85	3,02	4,17	**1**
'la01'	4,83	7,2	4,5	5,62	7,37	9,15	4,52	6,03	4,27	**1,52**	6,87	7,05	4,77	6,28	6,48	7,65	3,55	5,68	5	**1,67**	8,3	5,15	6,08	5,48	8,28	5,93	2,38	4,7	6,85	**1,83**
'la02'	5,43	7,67	4,52	5,53	6,98	7,23	3,93	6,08	4,77	**2,85**	7,45	6,8	4,18	4,95	8,27	6,97	3,8	4,88	4,82	**2,88**	9,52	5,2	5,17	5,02	8,27	6,27	3,47	3,38	5,9	**2,82**
'la04'	5,8	5,25	5,17	5,93	5,13	8	4,5	6,12	5,2	**3,9**	7,47	5,68	4,8	5,83	6,77	5,28	**3,98**	5,25	5,95	**3,98**	9,27	4,52	5,4	5,33	7,25	4,72	**3,78**	4,42	6,53	**3,78**
'la05'	5,53	6,82	5,88	5,67	5,52	7,63	3,9	5,55	5,95	**2,55**	7,17	5,62	5,33	5,78	7,3	5,98	4,17	5,18	6,17	**2,3**	9,3	5,12	5,23	5,97	7,52	5,25	3,37	4,93	6,32	**2**
'la06'	5,23	7,5	5,12	5,8	7,42	8,7	3,6	6,4	3,92	**1,32**	7,87	6,97	4,27	5,92	7,4	7,7	3,47	5,8	4,53	**1,08**	9,98	6,63	5,72	5,67	7,5	7,22	2,58	4,2	4,45	**1,05**
'la07'	3,53	8,07	5,68	6,82	8,17	8,48	2,63	6,47	3,88	**1,27**	8	6,98	5,1	6,48	7,45	7,28	3,35	4,95	4,22	**1,18**	9,7	6,32	5,42	5,47	9,08	6,97	3,27	3,08	4,63	**1,07**
'la08'	4,98	7,6	4,92	6,8	8,13	8,43	3,17	5,93	3,9	**1,13**	7,43	7,45	4,25	6,2	7,17	7,38	3,98	5,83	4,15	**1,15**	9,87	6,62	5,32	5,25	9,08	6,48	2,88	3,33	5,13	**1,03**
'la09'	3,92	8,07	5,8	6,93	8,3	8,13	2,4	7,13	2,37	**1,95**	7,73	6,98	4,45	6,22	7,33	7,63	2,88	6,55	3,95	**1,27**	9,97	6,82	5,55	5,32	8,93	6,93	2,85	2,82	4,13	**1,68**
'la10'	4,05	8,05	5,75	6,6	8,13	7,93	2,97	7,63	2,38	**1,5**	7,1	7,75	4,3	5,58	7,88	7,67	3,42	5,58	4,68	**1,03**	9,87	5,92	4,73	6,52	8,63	6,57	3,48	3,33	4,85	**1,1**
'la11'	3,45	7,92	5,22	6,98	7,95	9,08	3,65	7,3	2,45	**1**	7,8	6,85	4,02	6,28	7,9	8,4	3,18	6,37	3,17	**1,03**	9,97	6,62	5,25	5,45	9,02	6,6	3,67	3,2	4,2	**1,03**
'la12'	5,28	7,65	4,98	7,2	8,22	8,73	2,53	6,4	2,23	**1,77**	8,15	7,33	4,15	5,88	7,45	8	2,92	6,4	3,25	**1,47**	10	7,23	4,37	4,87	9	6,88	2,73	3,45	4,35	**2,12**
'la13'	4,12	8	5,47	6,82	8,55	7,32	3,17	7,9	2,43	**1,23**	8,32	7,6	4,25	5,92	7,35	7,18	2,72	6,93	3,68	**1,05**	9,97	6,77	4,95	5,1	8,97	7,23	3,4	3,08	4,42	**1,12**
'la14'	4,18	8,17	4,95	6,8	8,62	7,95	3,4	7,5	2,07	**1,37**	7,75	7,6	4	5,98	7,57	7,48	3,13	6,55	3,8	**1,13**	9,9	6,87	4,68	5,32	9,03	6,87	3,08	3,45	4,32	**1,48**
'la15'	4,8	7,78	5,05	6,43	7,82	8,45	3,22	7,33	3,12	**1**	7,68	7,17	4,4	6,77	7,03	7	3,23	7,35	3,33	**1,03**	9,87	6,8	4,87	5,17	9,07	6,83	3,27	3,52	4,6	**1,02**
'la16'	5,6	6,38	4,7	6,87	8,27	7,75	3,72	6,93	3,27	**1,52**	7,45	7,18	5,43	6,18	7,95	7,48	3,37	4,62	4,12	**1,22**	9,82	6,43	5,95	5,33	8,92	6,68	3,02	3,62	4,02	**1,22**
'la17'	4,73	7,72	5,5	5,58	7,5	9,1	3,93	7,15	2,73	**1,05**	8,28	6,58	4,42	6,15	7,72	7,32	3,8	6,58	3	**1,15**	9,87	6,17	6,38	5,08	8,65	6,75	3,12	2,4	5,25	**1,33**
'la18'	2,53	7,7	6,38	6,67	6,75	9,5	3	7,7	2,82	**1,95**	7,68	7,52	4,12	6,8	7,33	6,8	3,18	6,47	3,9	**1,2**	9,87	6,43	5,58	4,73	8,88	6,27	2,85	3,1	6,1	**1,18**
'la19'	5,43	7,05	5,28	6,83	7,8	8,35	3,87	5,93	3,42	**1,03**	7,48	6,4	4,93	6,37	7,68	7,13	3,6	6,22	4,12	**1,07**	9,93	5,98	6,48	5,03	8,55	6,35	2,78	2,95	5,78	**1,15**
'la20'	5,7	6,95	4,77	5,57	8,4	8,33	3,72	6,38	2,9	**2,28**	7,63	7,75	3,97	6,85	7,02	6,55	3,33	6,25	3,63	**2,02**	9,63	5,7	6,12	4,78	8,95	6,3	**2,55**	3,28	5,13	**2,55**
'la21'	3,93	8,47	6,05	6,32	7,2	9,35	2,45	7,43	2,73	**1,07**	7,83	7,53	4,33	5,82	8,25	7	2,73	6,92	3,58	**1**	9,97	6,9	5,67	5,6	9	7,13	2,97	2,77	3,93	**1,07**
'la22'	3,98	8,93	6,28	7,15	7,37	7,02	2,45	8,13	2,6	**1,08**	8,3	7,32	4,03	6,42	7,95	7,27	3,05	6,5	3,17	**1**	9,95	6,58	5,52	5,45	9,05	7,13	3,2	2,53	4,55	**1,03**
'la23'	3,72	8,5	6,9	6,37	7,52	7,1	2,65	8,47	2,72	**1,07**	8,83	6,92	4,33	6,27	7,8	7,07	3,23	6,95	2,6	**1**	10	6,88	5,15	4,9	9	7,18	3	3,07	4,82	**1**
'la24'	3,77	8,2	5,43	6,85	7,6	8,43	3,27	8,15	2,27	**1,03**	7,8	7,1	4,42	6,15	8,1	7,67	3,03	6,87	2,87	**1**	10	6,9	5,6	4,93	8,77	7,32	3,32	2,63	4,53	**1**
'la25'	3,72	8,72	6,03	6,58	7,92	7,45	2,87	8,17	2,45	**1,1**	7,98	7,23	4,33	6,97	7,55	7,15	2,68	6,7	3,4	**1**	10	6,7	5,9	4,72	8,82	7,35	3,2	2,92	4,4	**1**
'la26'	3,83	7,82	5,58	6,8	8,12	8,7	2,77	7,85	2,53	**1**	9,27	7,48	4,13	5,97	7,07	7,25	2,7	7,3	2,83	**1**	10	7,02	5,37	5,42	8,98	7,38	3,2	2,45	4,17	**1,02**
'la27'	4	7,9	6,13	5,98	7,63	9,2	2,33	8,02	2,67	**1,13**	8,22	7,65	4,6	5,7	7,63	7,43	3,05	7,17	2,55	**1**	10	6,97	4,73	5,18	8,9	7,25	3,15	2,95	4,87	**1**
'la28'	4,33	7,82	5,52	6,57	8,65	9,13	2,45	6,95	2,58	**1**	8,38	7,17	4,23	5,55	7,42	7,55	2,83	7,5	3,37	**1**	10	7,07	5,33	5,35	8,9	7,38	2,87	3,13	3,93	**1,03**
'la29'	3,5	8,9	6,38	5,97	6,9	8,62	2,58	8,2	2,95	**1**	8,13	6,48	4,35	6,7	7,87	7,52	2,85	7,1	3	**1**	10	7,02	5,02	5,85	8,83	7,3	2,95	2,67	4,37	**1**
'la30'	4,03	8,15	5,78	6,15	7,93	8,98	2,78	7,93	2,25	**1**	8,03	7,38	4,38	5,62	7,82	7,73	2,68	7,25	3,1	**1**	10	7	5,52	5,1	8,93	7,17	3,47	2,23	4,58	**1**
'orb01'	3,68	8,5	6,32	5,92	6,78	9,23	2,55	7,13	3,72	**1,17**	7,48	6,55	4,8	6,13	7,57	7,43	3,7	6,2	4,07	**1,07**	9,82	6,6	6	5,27	8,8	7,08	3,05	2,4	4,93	**1,05**
'orb02'	3,13	8,97	6,77	6,1	6,62	8,63	2,75	7,2	3,45	**1,38**	7,93	6,88	3,98	6,77	7,97	7,72	3	6,03	3,63	**1,08**	9,97	5,93	5,87	5,7	8,83	5,97	2,62	2,82	6,25	**1,05**
'orb03'	4,97	7,85	4,83	6,37	7,32	9,17	2,55	7,7	3,08	**1,17**	7,77	7,35	5,07	5,7	8	7,65	2,98	5,93	3,52	**1,03**	9,88	6,32	6,08	5,3	8,92	7,1	2,93	2,95	4,52	**1**
'orb04'	4,25	7,6	5,97	7,1	7,07	8,4	2,58	7,6	3,28	**1,15**	7,83	6,73	4,95	5,92	7,33	7,15	3,38	6,6	3,97	**1,13**	9,63	6,57	5,7	5,22	9,07	6,33	3,05	2,92	5,37	**1,15**
'orb05'	6,48	7,45	4,83	7,08	8,33	7,02	3,32	6,12	3,28	**1,08**	8,35	7,78	4,68	6,07	7,75	6,6	3,27	5,77	3,73	**1**	9,57	5,75	6,45	5,25	9,22	5,97	3,13	2,82	5,7	**1,15**
'orb06'	4,75	7	5,62	6,58	8,08	8,78	2,92	7	3,27	**1**	8,15	7,25	4,33	6,08	7,82	7,08	3,18	5,8	4,3	**1**	9,92	6,02	6,13	5,45	8,87	5,73	3,15	2,6	6,03	**1,1**
'orb07'	3,28	8,37	6,05	7,25	8,23	7,88	1,95	6,92	3,22	**1,85**	7,13	6,33	5,07	6,02	7,57	7,8	3,4	6,72	3,93	**1,03**	9,83	5,87	6,53	5,18	9	6,33	2,95	3	5,27	**1,03**
'orb08'	6,83	7,65	4,18	6,43	7,87	7,8	3,3	6,47	3,4	**1,07**	7,78	7,08	4,48	6,02	8,12	7,12	3,28	6,28	3,77	**1,07**	8,97	6,72	5,8	5,15	8	7,5	3,4	2,25	6,07	**1,15**
'orb09'	4,07	6,98	6,1	6,75	7,83	8,78	3,03	7,27	3,15	**1,03**	7,77	6,5	4,67	6,92	7,02	6,98	3,45	6,6	4,05	**1,05**	9,9	6,32	6,07	5,47	8,68	6,95	2,97	2,63	4,98	**1,03**
'orb10'	4,55	8,18	5,87	6,58	7,57	8,53	3,2	6,27	3,12	**1,13**	8,37	6,55	4,2	6,48	7,62	7,03	3,8	5,62	4,28	**1,05**	9,97	6,47	6,22	5,68	8,28	6,15	2,83	2,37	5,68	**1,35**

Çizelge 6.6 3 Tesis Sayısına Göre Friedman Test Sonuçları

	RK										ROV										SPV									
	PSO	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**	**PSO**	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**	**PSO**	**TSA**	**ABC**	**LFD**	**JAYA**	**RFO**	**DSA**	**GWO**	**WOA**	**AAA**
'ft06'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'ft10'	6,02	6,72	4,6	7,22	8,93	7,28	3,63	6,82	2,12	**1,67**	7,53	7,45	4,92	5,75	7,62	7,43	3,38	5,7	4	**1,22**	9,87	5,43	6,35	5,53	8,73	6,02	2,82	2,97	5,55	**1,73**
'ft20'	8,82	6,15	3,93	5,9	8,62	6,4	3,63	5,48	5	**1,07**	8,22	7,02	4,43	6,47	6,82	7,53	3,43	6,4	3,6	**1,08**	9,8	6,67	5,47	5,63	8,6	7,07	2,52	4,1	4,05	**1,1**
'abz5'	4,72	5,8	4,9	5,3	4,87	8,98	4,72	6,28	4,72	4,72	7,55	5,48	4,5	5,83	7,1	5,67	**4,25**	5,67	4,7	**4,25**	9,9	4,77	5,57	4,63	8,82	4,78	**3,95**	**3,95**	4,68	**3,95**
'abz6'	5,62	6,08	4,65	5,92	5,35	7,82	4,4	6,47	4,42	4,28	7,18	6,13	5,18	5,2	7,2	6,28	4,23	5,15	4,45	**3,98**	9,6	4,72	5,87	4,08	8,83	4,55	**3,75**	**4,12**	5,73	**3,75**
'abz7'	3,88	8,25	6,23	5,67	7,02	9,28	2,63	8,53	2,5	**1**	8,25	7,57	4,22	6,77	7,47	7,07	2,67	7,2	2,8	**1**	10	7,02	5,37	5,3	8,9	7,13	3,02	2,88	4,38	**1**
'abz8'	3,97	8,68	6,47	5,18	6,65	9,43	2,72	8,45	2,45	**1**	7,98	7,55	4,88	6,75	7,1	7,05	2,63	6,97	3,08	**1**	10	6,7	5,27	5,73	8,95	7,6	2,92	2,82	4	**1,02**
'abz9'	3,87	8,25	6,52	5,23	7,03	9,48	2,67	8,42	2,53	**1**	8,47	7,27	4,5	5,93	7,7	6,55	2,95	7,92	2,72	**1**	10	7,1	5,05	5,37	8,87	6,97	2,88	3,82	3,95	**1**
'la01'	9	**4,55**	**4,55**	**4,55**	5,87	5,13	5,05	**4,55**	7,2	**4,55**	6,87	5	5,17	5	5,98	5	5	5,33	6,65	5	5,17	5,17	5,17	5,17	6,32	5,17	5,17	5,5	7,02	5,17
'la02'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'la04'	6,57	5,12	5,12	5,12	5,77	6,22	5,12	5,12	5,75	5,12	6,43	5,27	5,27	5,27	5,92	5,27	5,27	5,27	5,78	5,27	7,47	**4,8**	4,93	**4,8**	7,12	**4,8**	**4,8**	**4,8**	6,68	**4,8**
'la05'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'la06'	6,73	6,37	4,82	5,73	5,53	6,42	4,15	6,8	7,02	**1,43**	7,48	7,02	4,55	6,25	7,12	6,62	3,78	6,67	4,35	**1,17**	9,82	6,2	5,9	6,18	8,12	5,58	3,28	3,75	4,88	**1,28**
'la07'	4,7	7,02	5,45	5,87	7,67	9,3	3,2	6,6	4,1	**1,1**	7,27	7,23	4,3	6,3	7,37	6,85	3,58	6,55	4,22	**1,33**	9,77	6,38	5,3	5,7	8,78	6,95	3	3,62	4,13	**1,37**
'la08'	3,6	7,05	6,15	6,62	5,85	9,38	3,17	7,07	4,65	**1,47**	7,72	7,05	5,57	5,9	6,9	6,23	3,67	5,33	5,3	**1,33**	9,93	5,67	5,23	5,85	8,78	6,32	2,93	3,45	5,55	**1,28**
'la09'	3,62	6,48	5,15	7,35	7,38	8,58	3,28	6,27	3,72	**3,17**	7	7,13	4,42	6,35	6,85	5,98	2,88	6,5	5,23	**2,65**	9,9	4,47	4,97	5,45	8,9	5,8	3	3,9	5,95	**2,67**
'la10'	4,17	7,72	5,05	5,47	6,62	7,85	4,17	5,63	4,17	4,17	7,15	7,15	4	5,53	6,93	6,87	3,77	5,12	4,85	**3,63**	9,97	5,07	4,45	5,35	8,68	5,82	**3,45**	**3,95**	4,82	**3,45**
'la11'	3,97	8,23	5,58	6,8	7,77	7,03	4,22	7,57	2,77	**1,07**	7,75	6,92	4,53	5,47	7,55	7,72	3,52	6,7	3,72	**1,13**	10	6,53	5,02	5,23	8,82	7,2	3,05	3,13	4,85	**1,17**
'la12'	4,92	7,75	5,15	6,37	8,4	8,07	3	7,3	2,98	**1,07**	8,02	7,08	3,97	5,85	7,6	8	3,17	6,52	3,8	**1**	9,93	6,93	5,15	4,77	9,03	6,73	3,22	3,68	4,55	**1**
'la13'	6,5	7,22	4,85	7,07	8,72	7,87	3,48	5,88	2,33	**1,08**	7,97	7,53	4,2	6,55	7,33	6,93	2,95	6,83	3,63	**1,07**	10	6,57	5,5	4,95	8,95	6,32	3,23	3,62	4,82	**1,05**
'la14'	3,5	8,02	5,7	6,52	7,68	8,52	3,23	7,43	3,17	**1,23**	7,82	6,98	4,5	6,47	7,52	7,65	2,82	6,23	3,92	**1,1**	9,95	6,95	5,27	4,72	9,03	6,75	3	3,87	4,43	**1,03**
'la15'	4,48	8,2	6,18	5,78	7,08	9,48	3,12	7,37	2,3	**1**	7,27	7,5	4,7	6,7	7,33	7,82	3,35	6,13	3,13	**1,07**	9,9	6,6	4,85	5,45	9,02	6,95	3,1	3,3	4,82	**1,02**
'la16'	6,92	4,97	4,97	4,97	6,38	6,93	4,97	4,97	4,97	4,97	5,93	5,43	5,43	5,43	5,6	5,43	5,43	5,43	5,43	5,43	9,22	**4,82**	**4,82**	**4,82**	7,08	**4,82**	**4,82**	**4,82**	4,98	**4,82**
'la17'	5,4	5,4	5,57	5,4	5,57	5,4	5,4	6,07	5,4	5,4	6,97	5,12	5,12	5,12	6,6	5,28	5,12	5,45	5,12	5,12	7,6	**5,07**	5,7	**5,07**	6,07	**5,07**	**5,07**	**5,07**	5,23	**5,07**
'la18'	5,48	5,48	5,48	5,48	5,48	5,65	5,48	5,48	5,48	5,48	5,95	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	9,55	**4,73**	**4,73**	**4,73**	7,58	**4,73**	**4,73**	**4,73**	**4,73**	**4,73**
'la19'	7,87	5,62	4,85	5,52	8,73	6,72	4,95	5,33	3,55	**1,87**	7,68	6,53	4,43	5,83	8,13	6,07	3,43	5,78	5,13	**1,97**	9,85	5,05	6,65	5,05	8,93	4,92	2,72	3,13	6,2	**2,5**
'la20'	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	5,65	**5,48**	**5,48**	**5,48**	**5,48**	5,65	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**
'la21'	3,68	7,9	6,1	6,5	7,18	8,95	3,33	7,6	2,73	**1,02**	8,23	6,7	4,62	6,28	7,37	6,67	3,12	7,15	3,83	**1,03**	9,97	6,45	5,35	5,97	9,03	6,37	3,02	3,02	4,78	**1,05**
'la22'	3,18	8,53	6,15	6,05	6,58	9,13	2,85	8,2	3,15	**1,17**	8,05	6,77	4,43	6,08	7,92	7,28	2,98	6,63	3,85	**1**	9,97	6,65	5,95	4,72	8,77	6,73	2,95	3,32	4,88	**1,07**
'la23'	3,57	9,08	6,6	6,13	7,4	7,48	2,28	8,12	2,87	**1,47**	8,22	6,78	4,8	6,05	7,73	7,55	2,9	6,23	3,73	**1**	10	7,02	5,73	5,45	8,77	6,8	2,9	2,82	4,47	**1,05**
'la24'	3,52	8,18	6,02	6,35	7,05	9,3	2,93	7,55	3,02	**1,08**	8,2	6,5	5,05	5,9	6,9	7,63	2,92	6,62	4,25	**1,03**	9,98	6,55	6,07	4,67	9,02	6,6	2,7	2,82	5,5	**1,1**
'la25'	3,58	7,63	5,45	6,37	7,87	9,05	3,17	7,67	3,12	**1,1**	7,93	7,25	4,65	6,38	7,23	6,95	2,72	6,72	3,98	**1,18**	9,87	6,5	4,97	5,73	9,03	6,45	3,2	2,52	5,28	**1,45**
'la26'	3,77	8,57	6,37	5,08	7,77	8,3	2,15	8,67	3,12	**1,22**	7,92	7,67	4,87	5,9	8,05	7,13	2,87	6,87	2,73	**1**	10	6,82	5,6	5,17	8,95	7,25	3,1	3,17	3,95	**1**
'la27'	4,37	7,65	5,22	6,63	8,07	8,98	2,6	7,88	2,6	**1**	8,87	7,07	4,3	6,15	8,1	6,82	2,88	6,68	3,13	**1**	10	6,82	5,23	5,12	8,97	7,33	3,27	3	4,23	**1,03**
'la28'	4,02	8,52	6,05	5,75	7,53	9,17	2,53	7,85	2,58	**1**	7,72	6,82	4,23	6,62	8,13	7,08	2,55	7,07	3,78	**1**	9,95	7,08	5,53	5,1	9,02	6,8	3,42	2,9	4,2	**1**
'la29'	3,9	8,77	6,52	4,98	7,27	8,77	2,47	8,52	2,52	**1,3**	7,68	7,8	4,68	5,83	7,63	6,9	2,88	7,05	3,53	**1**	10	6,82	5,15	5,47	8,98	7,05	3,18	2,7	4,6	**1,05**
'la30'	4,13	7,75	6,18	6,02	7,28	9,18	2,13	8,25	2,85	**1,22**	8,08	6,65	4,6	6,28	7,97	7,23	2,93	6,92	3,33	**1**	10	6,8	5,25	4,82	9	7,03	3,35	3,22	4,53	**1**
'orb01'	5,98	7,2	5,32	6,07	7,77	7,42	3,17	7,08	3,77	**1,23**	7,58	6,28	4,78	6,27	7,75	7,72	3,23	5,95	4,05	**1,38**	9,98	6,07	6,85	4,93	8,27	6,25	3,07	3,3	5,12	**1,17**
'orb02'	4,45	7,7	4,73	5,07	4,6	8,35	4,45	6,75	4,45	4,45	7,18	5,23	4,78	4,7	7,38	5,45	4,43	5,82	6,3	**3,72**	9,75	**4,32**	5,75	4,45	8,62	4,27	**3,63**	4,35	6,23	**3,63**
'orb03'	6,67	7,42	5,23	5,63	7,5	7,72	3	6,82	3,92	**1,1**	7,78	7,1	4,3	6,15	8,05	6,93	3,25	6,32	4,08	**1,03**	9,62	6,07	6,35	5,62	9,13	6,32	3,25	2,98	4,65	**1,02**
'orb04'	5,22	5,22	5,22	5,22	5,22	8,05	5,22	5,22	5,22	5,22	5,95	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	9,5	**4,85**	5	**4,85**	5,93	**4,85**	**4,85**	**4,85**	5,47	**4,85**
'orb05'	7,55	6,95	5,02	7,05	7,77	6,02	3,7	6,18	3,63	**1,13**	7,52	6,6	4,52	6,33	8,35	7,23	3,13	5,47	4,72	**1,13**	9,48	5,85	6,1	5,57	9,18	6,32	2,72	3,77	4,85	**1,17**
'orb06'	4,55	8,07	6,07	6,3	7,05	8,87	3,38	6,08	3,47	**1,17**	7,53	6,4	4,73	6,72	7,17	7,22	3,27	6,17	4,67	**1,13**	9,7	5,72	5,77	5	8,77	6,37	3,1	3,42	5,95	**1,22**
'orb07'	3,6	8,13	5,2	5,07	4,37	9,12	3,48	7,93	4,62	3,48	7,3	6,67	4,28	5,85	7,18	7	3,85	6,38	4,08	**2,4**	9,82	5,07	5,92	4,53	8,92	5,62	**2,75**	**3,45**	6,18	**2,75**
'orb08'	5,42	7,77	5,17	6,22	6,98	8,75	3,48	7,05	3,02	**1,15**	8,02	7,02	4,18	6,5	7,62	7,17	3,15	6,5	3,63	**1,22**	9,27	5,92	6,22	6,5	8,17	6,5	2,88	3,27	5,18	**1,1**
'orb09'	6,48	6,35	4,65	4,72	6,48	7,3	5,32	5,6	4,6	**3,5**	7,45	5,72	5,12	5,03	6,57	6	**4,1**	5,7	5,22	**4,1**	9,73	4,52	5,35	4,1	8,8	5,05	**3,85**	4,32	5,43	**3,85**
'orb10'	4,92	7,65	4,9	6,87	7,22	7,73	3,57	6,77	2,95	**2,43**	7,68	7,07	5,18	5,23	7,13	7,28	3,6	5,22	4,33	**2,27**	9,85	5,33	6,07	5,13	8,57	5,82	2,9	3,12	5,57	**2,65**

Çizelge 6.7 4 Tesis Sayısına Göre Friedman Test Sonuçları

	RK										ROV										SPV									
	PSO	TSA	ABC	LFD	JAYA	RFO	DSA	GWO	WOA	AAA	PSO	TSA	ABC	LFD	JAYA	RFO	DSA	GWO	WOA	AAA	PSO	TSA	ABC	LFD	JAYA	RFO	DSA	GWO	WOA	AAA
'ft06'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'ft10'	6,02	6,72	4,6	7,22	8,93	7,28	3,63	6,82	2,12	**1,67**	7,53	7,45	4,92	5,75	7,62	7,43	3,38	5,7	4	**1,22**	9,87	5,43	6,35	5,53	8,73	6,02	2,82	2,97	5,55	**1,73**
'ft20'	8,82	6,15	3,93	5,9	8,62	6,4	3,63	5,48	5	**1,07**	8,22	7,02	4,43	6,47	6,82	7,53	3,43	6,4	3,6	**1,08**	9,8	6,67	5,47	5,63	8,6	7,07	2,52	4,1	4,05	**1,1**
'abz5'	4,72	5,8	4,9	5,3	4,87	8,98	4,72	6,28	4,72	4,72	7,55	5,48	4,5	5,83	7,1	5,67	**4,25**	5,67	4,7	**4,25**	9,9	4,77	5,57	4,63	8,82	4,78	**3,95**	**3,95**	4,68	**3,95**
'abz6'	5,62	6,08	4,65	5,92	5,35	7,82	4,4	6,47	4,42	4,28	7,18	6,13	5,18	5,2	7,2	6,28	4,23	5,15	4,45	**3,98**	9,6	4,72	5,87	4,08	8,83	4,55	**3,75**	**4,12**	5,73	**3,75**
'abz7'	3,88	8,25	6,23	5,67	7,02	9,28	2,63	8,53	2,5	**1**	8,25	7,57	4,22	6,77	7,47	7,07	2,67	7,2	2,8	**1**	10	7,02	5,37	5,3	8,9	7,13	3,02	2,88	4,38	**1**
'abz8'	3,97	8,68	6,47	5,18	6,65	9,43	2,72	8,45	2,45	**1**	7,98	7,55	4,88	6,75	7,1	7,05	2,63	6,97	3,08	**1**	10	6,7	5,27	5,73	8,95	7,6	2,92	2,82	4	**1,02**
'abz9'	3,87	8,25	6,52	5,23	7,03	9,48	2,67	8,42	2,53	**1**	8,47	7,27	4,5	5,93	7,7	6,55	2,95	7,92	2,72	**1**	10	7,1	5,05	5,37	8,87	6,97	2,88	3,82	3,95	**1**
'la01'	9	**4,55**	**4,55**	**4,55**	5,87	5,13	5,05	**4,55**	7,2	**4,55**	6,87	5	5,17	5	5,98	5	5	5,33	6,65	5	5,17	5,17	5,17	5,17	6,32	5,17	5,17	5,5	7,02	5,17
'la02'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'la04'	6,57	5,12	5,12	5,12	5,77	6,22	5,12	5,12	5,75	5,12	6,43	5,27	5,27	5,27	5,92	5,27	5,27	5,27	5,78	5,27	7,47	**4,8**	4,93	**4,8**	7,12	**4,8**	**4,8**	**4,8**	6,68	**4,8**
'la05'	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**	**5,5**
'la06'	6,73	6,37	4,82	5,73	5,53	6,42	4,15	6,8	7,02	**1,43**	7,48	7,02	4,55	6,25	7,12	6,62	3,78	6,67	4,35	**1,17**	9,82	6,2	5,9	6,18	8,12	5,58	3,28	3,75	4,88	**1,28**
'la07'	4,7	7,02	5,45	5,87	7,67	9,3	3,2	6,6	4,1	**1,1**	7,27	7,23	4,3	6,3	7,37	6,85	3,58	6,55	4,22	**1,33**	9,77	6,38	5,3	5,7	8,78	6,95	3	3,62	4,13	**1,37**
'la08'	3,6	7,05	6,15	6,62	5,85	9,38	3,17	7,07	4,65	**1,47**	7,72	7,05	5,57	5,9	6,9	6,23	3,67	5,33	5,3	**1,33**	9,93	5,67	5,23	5,85	8,78	6,32	2,93	3,45	5,55	**1,28**
'la09'	3,62	6,48	5,15	7,35	7,38	8,58	3,28	6,27	3,72	**3,17**	7	7,13	4,42	6,35	6,85	5,98	2,88	6,5	5,23	**2,65**	9,9	4,47	4,97	5,45	8,9	5,8	3	3,9	5,95	**2,67**
'la10'	4,17	7,72	5,05	5,47	6,62	7,85	4,17	5,63	4,17	4,17	7,15	7,15	4	5,53	6,93	6,87	3,77	5,12	4,85	**3,63**	9,97	5,07	4,45	5,35	8,68	5,82	**3,45**	**3,95**	4,82	**3,45**
'la11'	3,97	8,23	5,58	6,8	7,77	7,03	4,22	7,57	2,77	**1,07**	7,75	6,92	4,53	5,47	7,55	7,72	3,52	6,7	3,72	**1,13**	10	6,53	5,02	5,23	8,82	7,2	3,05	3,13	4,85	**1,17**
'la12'	4,92	7,75	5,15	6,37	8,4	8,07	3	7,3	2,98	**1,07**	8,02	7,08	3,97	5,85	7,6	8	3,17	6,52	3,8	**1**	9,93	6,93	5,15	4,77	9,03	6,73	3,22	3,68	4,55	**1**
'la13'	6,5	7,22	4,85	7,07	8,72	7,87	3,48	5,88	2,33	**1,08**	7,97	7,53	4,2	6,55	7,33	6,93	2,95	6,83	3,63	**1,07**	10	6,57	5,5	4,95	8,95	6,32	3,23	3,62	4,82	**1,05**
'la14'	3,5	8,02	5,7	6,52	7,68	8,52	3,23	7,43	3,17	**1,23**	7,82	6,98	4,5	6,47	7,52	7,65	2,82	6,23	3,92	**1,1**	9,95	6,95	5,27	4,72	9,03	6,75	3	3,87	4,43	**1,03**
'la15'	4,48	8,2	6,18	5,78	7,08	9,48	3,12	7,37	2,3	**1**	7,27	7,5	4,7	6,7	7,33	7,82	3,35	6,13	3,13	**1,07**	9,9	6,6	4,85	5,45	9,02	6,95	3,1	3,3	4,82	**1,02**
'la16'	6,92	4,97	4,97	4,97	6,38	6,93	4,97	4,97	4,97	4,97	5,93	5,43	5,43	5,43	5,6	5,43	5,43	5,43	5,43	5,43	9,22	**4,82**	**4,82**	**4,82**	7,08	**4,82**	**4,82**	**4,82**	4,98	**4,82**
'la17'	5,4	5,4	5,57	5,4	5,57	5,4	5,4	6,07	5,4	5,4	6,97	5,12	5,12	5,12	6,6	5,28	5,12	5,45	5,12	5,12	7,6	**5,07**	5,7	**5,07**	6,07	**5,07**	**5,07**	**5,07**	5,23	**5,07**
'la18'	5,48	5,48	5,48	5,48	5,48	5,65	5,48	5,48	5,48	5,48	5,95	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	5,45	9,55	**4,73**	**4,73**	**4,73**	7,58	**4,73**	**4,73**	**4,73**	**4,73**	**4,73**
'la19'	7,87	5,62	4,85	5,52	8,73	6,72	4,95	5,33	3,55	**1,87**	7,68	6,53	4,43	5,83	8,13	6,07	3,43	5,78	5,13	**1,97**	9,85	5,05	6,65	5,05	8,93	4,92	2,72	3,13	6,2	**2,5**
'la20'	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	5,5	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	5,65	**5,48**	**5,48**	**5,48**	**5,48**	5,65	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**	**5,48**
'la21'	3,68	7,9	6,1	6,5	7,18	8,95	3,33	7,6	2,73	**1,02**	8,23	6,7	4,62	6,28	7,37	6,67	3,12	7,15	3,83	**1,03**	9,97	6,45	5,35	5,97	9,03	6,37	3,02	3,02	4,78	**1,05**
'la22'	3,18	8,53	6,15	6,05	6,58	9,13	2,85	8,2	3,15	**1,17**	8,05	6,77	4,43	6,08	7,92	7,28	2,98	6,63	3,85	**1**	9,97	6,65	5,95	4,72	8,77	6,73	2,95	3,32	4,88	**1,07**
'la23'	3,57	9,08	6,6	6,13	7,4	7,48	2,28	8,12	2,87	**1,47**	8,22	6,78	4,8	6,05	7,73	7,55	2,9	6,23	3,73	**1**	10	7,02	5,73	5,45	8,77	6,8	2,9	2,82	4,47	**1,05**
'la24'	3,52	8,18	6,02	6,35	7,05	9,3	2,93	7,55	3,02	**1,08**	8,2	6,5	5,05	5,9	6,9	7,63	2,92	6,62	4,25	**1,03**	9,98	6,55	6,07	4,67	9,02	6,6	2,7	2,82	5,5	**1,1**
'la25'	3,58	7,63	5,45	6,37	7,87	9,05	3,17	7,67	3,12	**1,1**	7,93	7,25	4,65	6,38	7,23	6,95	2,72	6,72	3,98	**1,18**	9,87	6,5	4,97	5,73	9,03	6,45	3,2	2,52	5,28	**1,45**
'la26'	3,77	8,57	6,37	5,08	7,77	8,3	2,15	8,67	3,12	**1,22**	7,92	7,67	4,87	5,9	8,05	7,13	2,87	6,87	2,73	**1**	10	6,82	5,6	5,17	8,95	7,25	3,1	3,17	3,95	**1**
'la27'	4,37	7,65	5,22	6,63	8,07	8,98	2,6	7,88	2,6	**1**	8,87	7,07	4,3	6,15	8,1	6,82	2,88	6,68	3,13	**1**	10	6,82	5,23	5,12	8,97	7,33	3,27	3	4,23	**1,03**
'la28'	4,02	8,52	6,05	5,75	7,53	9,17	2,53	7,85	2,58	**1**	7,72	6,82	4,23	6,62	8,13	7,08	2,55	7,07	3,78	**1**	9,95	7,08	5,53	5,1	9,02	6,8	3,42	2,9	4,2	**1**
'la29'	3,9	8,77	6,52	4,98	7,27	8,77	2,47	8,52	2,52	**1,3**	7,68	7,8	4,68	5,83	7,63	6,9	2,88	7,05	3,53	**1**	10	6,82	5,15	5,47	8,98	7,05	3,18	2,7	4,6	**1,05**
'la30'	4,13	7,75	6,18	6,02	7,28	9,18	2,13	8,25	2,85	**1,22**	8,08	6,65	4,6	6,28	7,97	7,23	2,93	6,92	3,33	**1**	10	6,8	5,25	4,82	9	7,03	3,35	3,22	4,53	**1**
'orb01'	5,98	7,2	5,32	6,07	7,77	7,42	3,17	7,08	3,77	**1,23**	7,58	6,28	4,78	6,27	7,75	7,72	3,23	5,95	4,05	**1,38**	9,98	6,07	6,85	4,93	8,27	6,25	3,07	3,3	5,12	**1,17**
'orb02'	**4,45**	7,7	4,73	5,07	4,6	8,35	**4,45**	6,75	**4,45**	**4,45**	7,18	5,23	4,78	4,7	7,38	5,45	4,43	5,82	6,3	**3,72**	9,75	**4,32**	5,75	4,45	8,62	4,27	**3,63**	4,35	6,23	**3,63**
'orb03'	6,67	7,42	5,23	5,63	7,5	7,72	3	6,82	3,92	**1,1**	7,78	7,1	4,3	6,15	8,05	6,93	3,25	6,32	4,08	**1,03**	9,62	6,07	6,35	5,62	9,13	6,32	3,25	2,98	4,65	**1,02**
'orb04'	**5,22**	**5,22**	**5,22**	**5,22**	**5,22**	**8,05**	**5,22**	**5,22**	**5,22**	**5,22**	5,95	**5,45**	**5,45**	**5,45**	**5,45**	**5,45**	**5,45**	**5,45**	**5,45**	**5,45**	9,5	**4,85**	5	**4,85**	5,93	**4,85**	**4,85**	**4,85**	5,47	**4,85**
'orb05'	7,55	6,95	5,02	7,05	7,77	6,02	3,7	6,18	3,63	**1,13**	7,52	6,6	4,52	6,33	8,35	7,23	3,13	5,47	4,72	**1,13**	9,48	5,85	6,1	5,57	9,18	6,32	2,72	3,77	4,85	**1,17**
'orb06'	4,55	8,07	6,07	6,3	7,05	8,87	3,38	6,08	3,47	**1,17**	7,53	6,4	4,73	6,72	7,17	7,22	3,27	6,17	4,67	**1,13**	9,7	5,72	5,77	5	8,77	6,37	3,1	3,42	5,95	**1,22**
'orb07'	3,6	8,13	5,2	5,07	4,37	9,12	3,48	7,93	4,62	3,48	7,3	6,67	4,28	5,85	7,18	7	3,85	6,38	4,08	**2,4**	9,82	5,07	5,92	4,53	8,92	5,62	**2,75**	**3,45**	6,18	**2,75**
'orb08'	5,42	7,77	5,17	6,22	6,98	8,75	3,48	7,05	3,02	**1,15**	8,02	7,02	4,18	6,5	7,62	7,17	3,15	6,5	3,63	**1,22**	9,27	5,92	6,22	6,5	8,17	6,5	2,88	3,27	5,18	**1,1**
'orb09'	6,48	6,35	4,65	4,72	6,48	7,3	5,32	5,6	4,6	**3,5**	7,45	5,72	5,12	5,03	6,57	6	**4,1**	5,7	5,22	**4,1**	9,73	4,52	5,35	4,1	8,8	5,05	**3,85**	4,32	5,43	**3,85**
'orb10'	4,92	7,65	4,9	6,87	7,22	7,73	3,57	6,77	2,95	**2,43**	7,68	7,07	5,18	5,23	7,13	7,28	3,6	5,22	4,33	**2,27**	9,85	5,33	6,07	5,13	8,57	5,82	2,9	3,12	5,57	**2,65**

Çizelge 6.8 Kodlama Şemalarına Göre Algoritma Sonuçlarının Ortalaması

	RK			ROV			SPV		
	2	**3**	**4**	**2**	**3**	**4**	**2**	**3**	**4**
PSO	711,68	618,06	579,95	738,12	632,54	587,20	820,79	682,93	621,81
TSA	743,43	634,56	589,54	742,18	633,04	589,29	740,39	633,37	587,16
ABC	727,52	622,77	584,18	726,25	623,66	583,91	740,91	634,75	590,04
LFD	724,66	621,56	583,25	731,04	627,68	585,64	715,54	620,45	581,5
JAYA	743,62	631,35	587,62	743,08	633,97	588,68	766,89	647,29	599,56
RFO	749,12	636,43	590,33	743,89	632,70	589,22	765,66	647,06	596,62
DSA	701,81	612,14	577,97	715,37	620,12	581,77	704,47	613,20	578,91
GWO	728,75	628,62	585,81	726,27	627,31	586,25	708,97	618,58	580,20
WOA	697,20	609,06	577,20	704,95	615,41	577,52	706,10	613,52	578,54
AAA	**677,83**	**601,06**	**573,93**	**677,20**	**600,54**	**573,85**	**676,25**	**601,06**	**573,93**

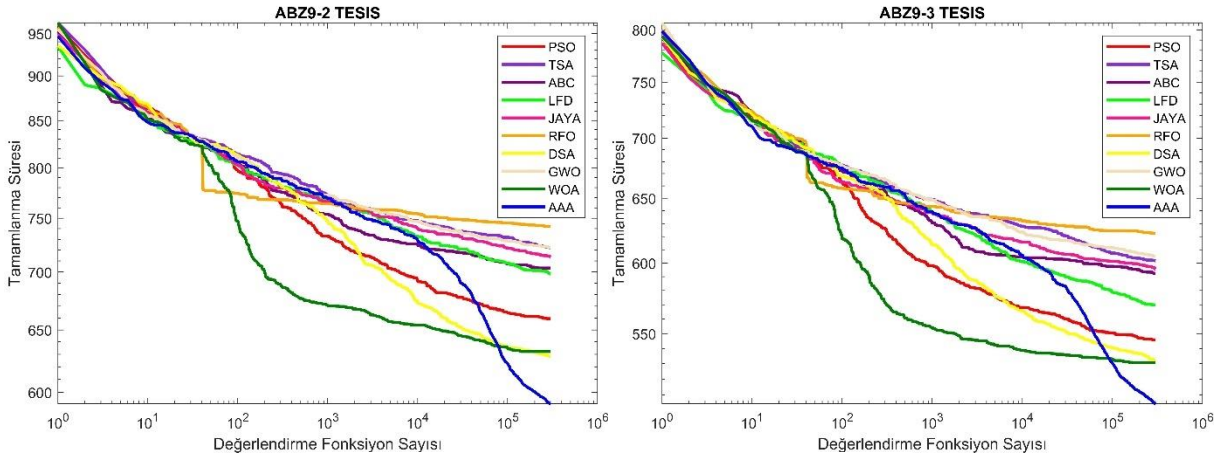
Uygulanan 10 algoritmanın performanslarının istatistiksel olarak anlamlı farklılıklar gösterip göstermediğini belirlemek için Friedman Testi kullanılmıştır. Friedman Testi çok değişkenli bir varyans analizi olup aynı örneklem uzayında birden fazla ölçüm yapıldığında kullanılmaktadır. Friedman testi sonuçlarına göre algoritmalar arasında anlamlı farklılıklar gözlenmiştir. Çizelge 6.5, 6.6, 6.7'de test sonuçları tesis sayısındaki farklılıklara göre karşılaştırılmakta ve en iyi performans gösteren değerler kalın olarak işaretlenmiştir. Özellikle AAA algoritması üç farklı kodlama şeması ve farklı tesis sayıları altında en düşük ortalama performans değerlerine ulaşarak DJSSP çözümünde en etkili yöntem olarak öne çıkmaktadır. Bunu takip eden algoritmalar arasında WOA ve GWO ön plana çıkmaktadır. Öte yandan TSA, RFO ve JAYA algoritmaları karşılaştırmalı testlerde zayıf performans göstermiştir. Bu bulgular algoritma seçiminin problem türüne ve sistem yapısına bağlı olarak önemli ölçüde değişebileceğini ortaya koymaktadır.

Her algoritmanın kendine özgü olması ve farklı parametreler kullanması sonuca ulaşırken farklılıklar yaratmaktadır. Her algoritmayı farklı kodlama şemalarında sadece algoritmanın kendi içerisinde 2-3-4 tesis durumuna göre 48 adet problemin ortalaması alınarak kıyasladığımız veriler, Çizelge 6.8'te verilmiştir. Çizelgede kalın renkte belirtilen veriler tesis sayısına göre bulunduğu algoritma ve kodlama şemasındaki problem uzayındaki ortalama sonuçlar arasında en iyi değere sahip verilerdir. Bu veriler incelendiğinde AAA algoritması tesis sayısına bakılmaksızın diğer algoritmalara göre en iyi tamamlama süresine sahiptir. Ayrıca, AAA'nın kısmen ROV kodlama şeması ile daha iyi sonuçlar verdiğini söyleyebiliriz.

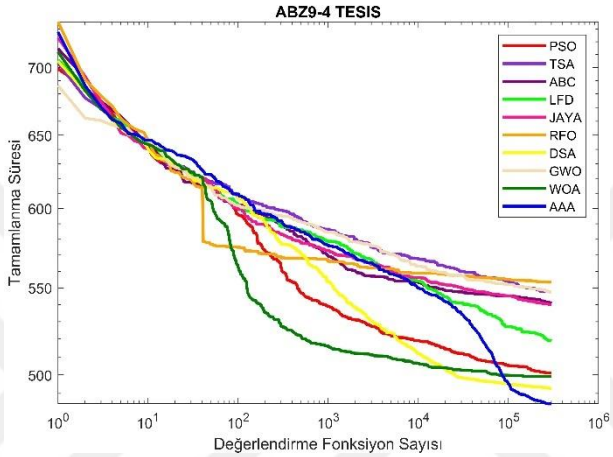
Çizelge 6.9 Kodlama Şemalarına Göre En İyi Sonuç Sayısı

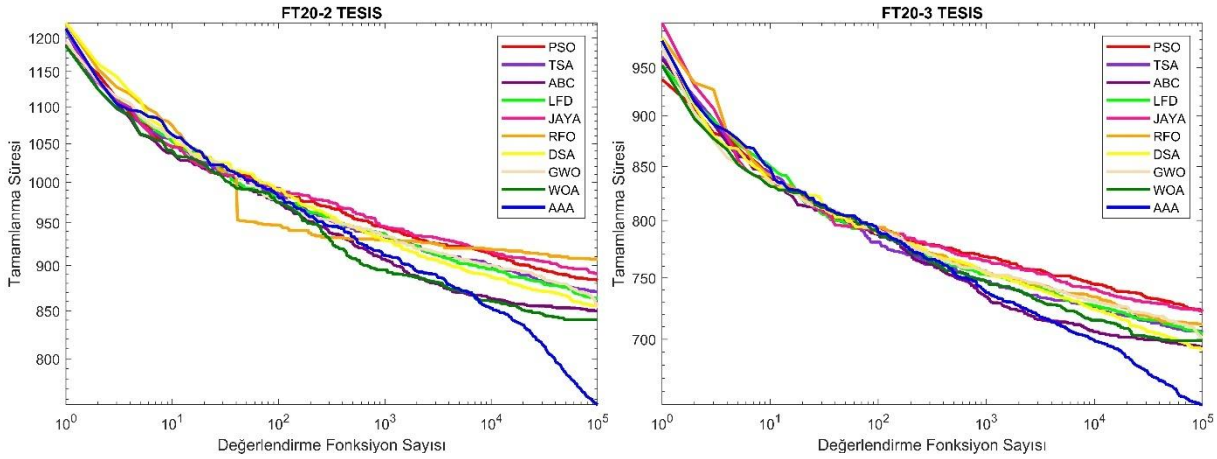
Tesis	RK			ROV			SPV			
Sayısı	**2**	**3**	**4**	**2**	**3**	**4**	**2**	**3**	**4**	**Toplam**
PSO	8	22	35	6	20	33	1	9	18	152
TSA	6	22	31	5	20	31	7	21	32	175
ABC	8	23	33	11	24	34	7	19	27	186
LFD	8	22	33	9	24	34	14	24	36	204
JAYA	3	21	29	8	19	30	4	15	25	154
RFO	4	20	31	6	21	31	3	17	26	159
DSA	16	25	37	15	24	35	18	29	36	235
GWO	13	24	33	9	25	35	12	26	33	210
WOA	17	28	37	15	28	38	16	28	36	243
AAA	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**48**	**432**
Toplam	131	255	347	132	253	349	130	236	317	2150

Bu çalışmada, DJSSP için 10 farklı algoritmanın 3 farklı kodlama şeması ve 3 farklı tesis sayısı ile 48 problem üzerinde performans değerleri ölçülmüştür. Algoritmaların performansları çeşitli kriterlere göre test edilmiştir. Bu kriterler arasında bu çalışma için en önemli kriter, üretim tesislerinde bir işin tamamlanması için gereken minimum zaman dilimidir. Kullanılan algoritmaların performansları Çizelge 6.2, 6.3, 6.4'te değerlendirilmiş olup, en iyi performans gösteren algoritma sonuçları koyu renkle işaretlenmiştir. Bu çizelgelerin kısa bir özeti olarak, Çizelge 6.9 algoritmaların verdiği en iyi sonuç sayısına göre düzenlenmiştir. Tüm bu çizelgelerde görüldüğü üzere AAA algoritması diğerlerine göre anlamlı bir sonuç elde etmiştir.

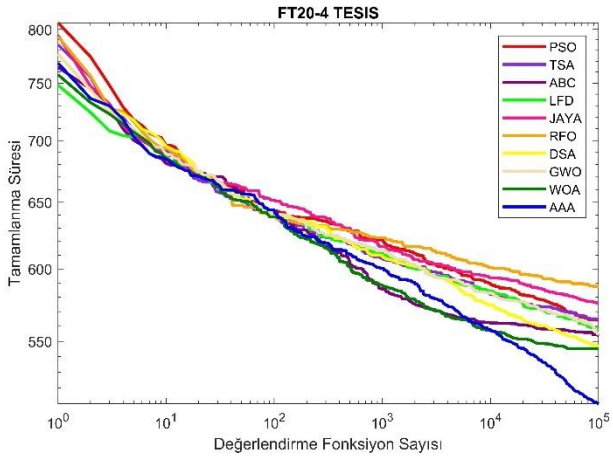

ABZ9-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

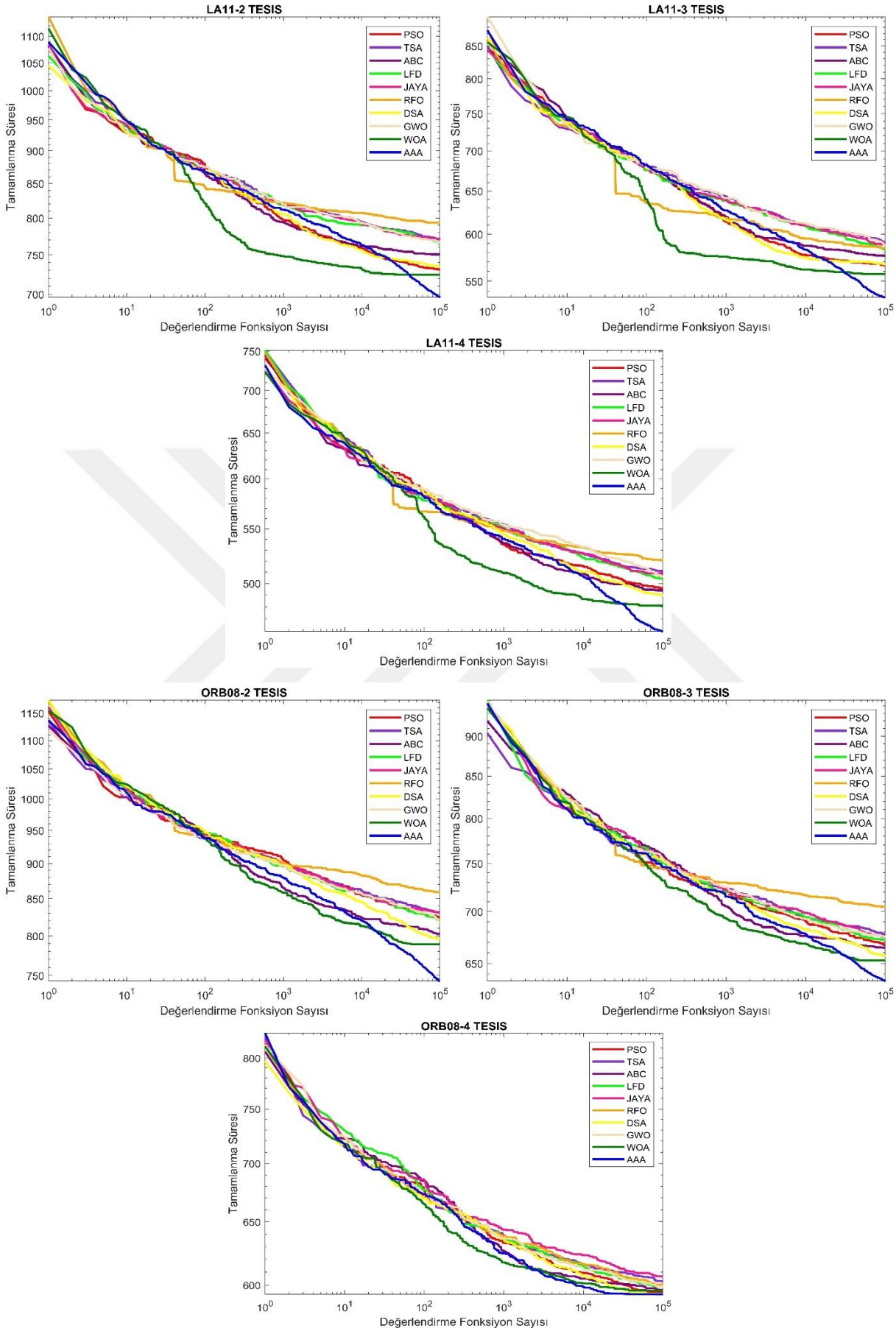
ABZ9-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


ABZ9-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


FT20-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

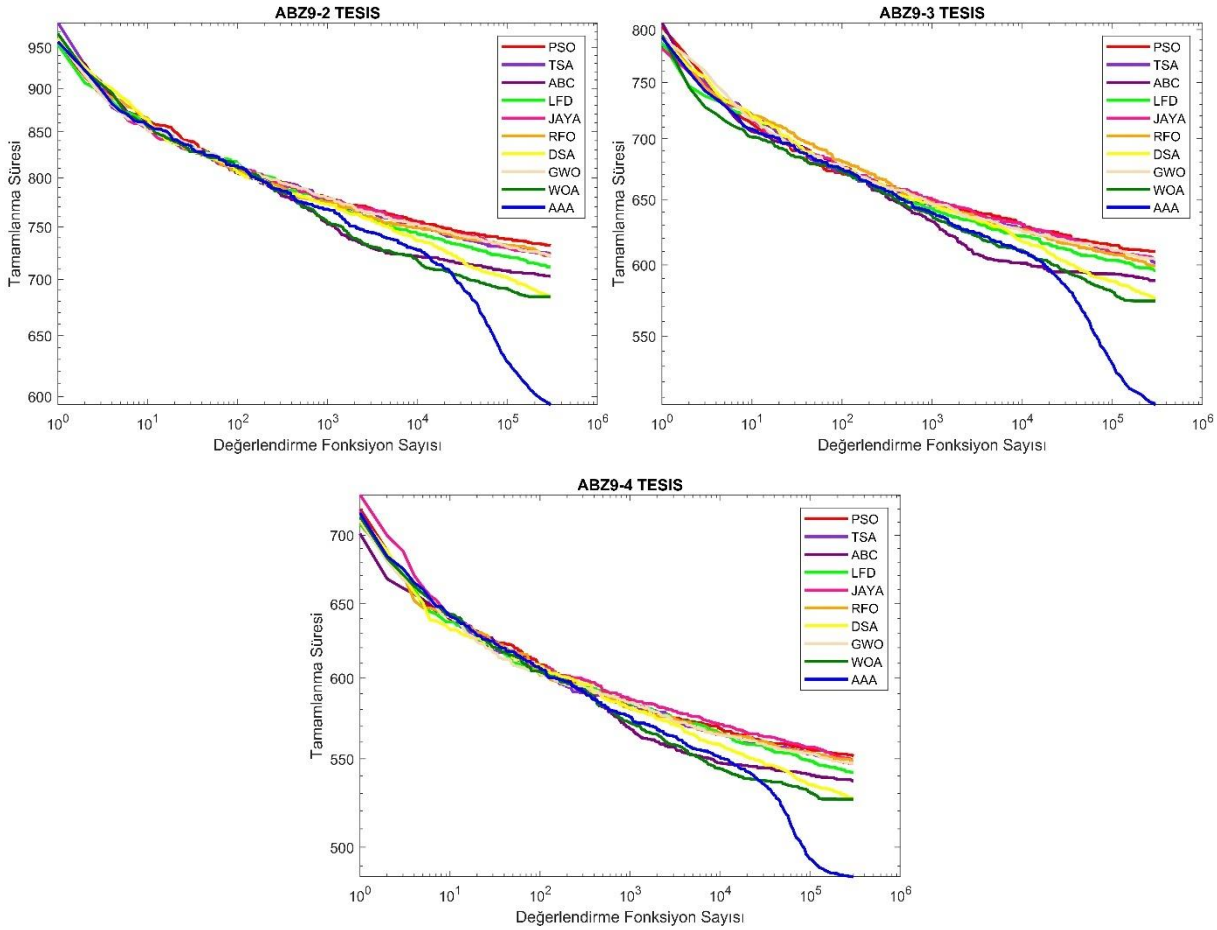
FT20-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

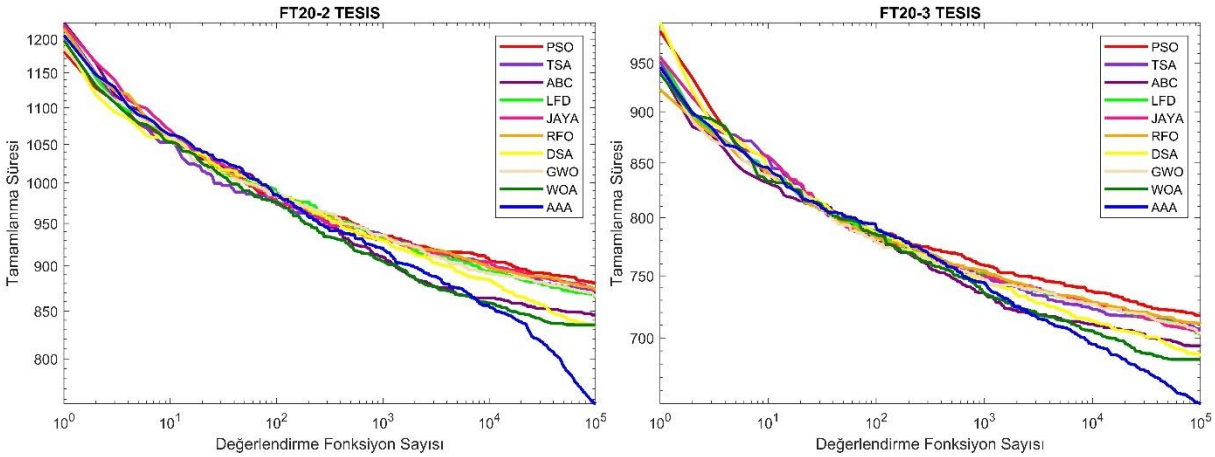

FT20-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA



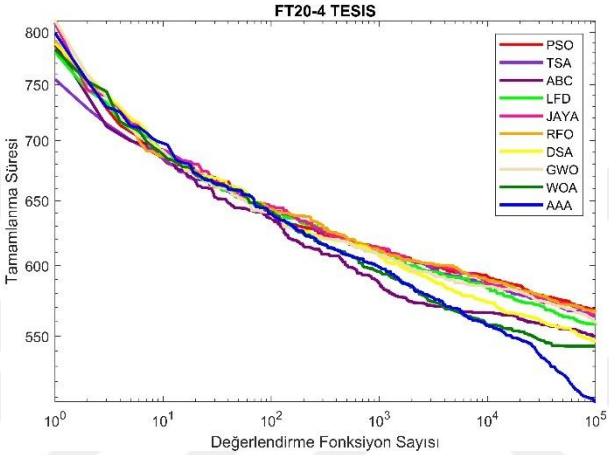
Şekil 6.1 RK Kodlama Şemasıyla algoritmaların (Abz9, Ft20, La11, Orb08) yakınsama grafikleri (Tesis:2-3-4)

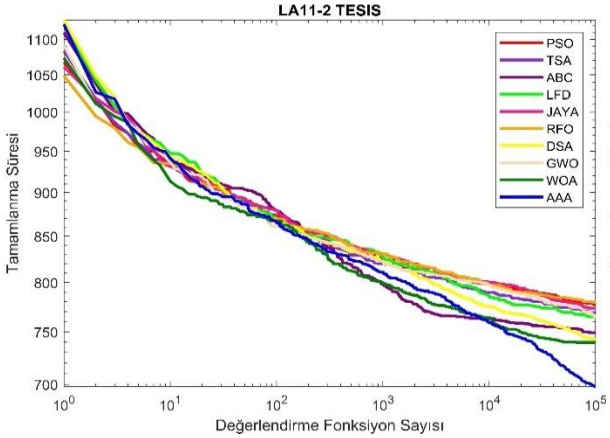
RK kodlama şemasının uygulanmasıyla 48 karşılaştırma problemi arasından örneklem olarak alınan 4 problemden elde edilen veriler Şekil 6.1 ile sunulmuştur. RK kodlama şeması uygulandıktan sonra ABZ09 probleminde ilk etapta LFD sonrasında DSA ve PSO hızlı düşüş gösterse de günün sonunda en iyi sonuç veren algoritma AAA oluştur. FT20 probleminde hemen hemen bütün algoritmalar aynı oranda iyileşme gösterse de ölçeklendirme arttıkça AAA algoritmasının daha ön plana çıktığı görülmektedir. LA11 probleminin çözümünde bütün algoritmalar birbirine yakın iyileşme süreci yaşarken ilk olarak RFO iyi bir sıçrama yapmış sonrasında yatay hareket ederek iyi bir sonuç çıkaramamıştır. En iyi sonucu AAA vermiştir, AAA algoritmasına en yakın sonucu ise LFD vermiştir. ORB08 problemini inceleyecek olursak özellikle 4 tesisli yapıda bütün algoritmaların birbirine yakın sonuçlar gösterdiği fakat günün sonunda AAA algoritmasının en iyi sonuç verdiği görülmüştür.

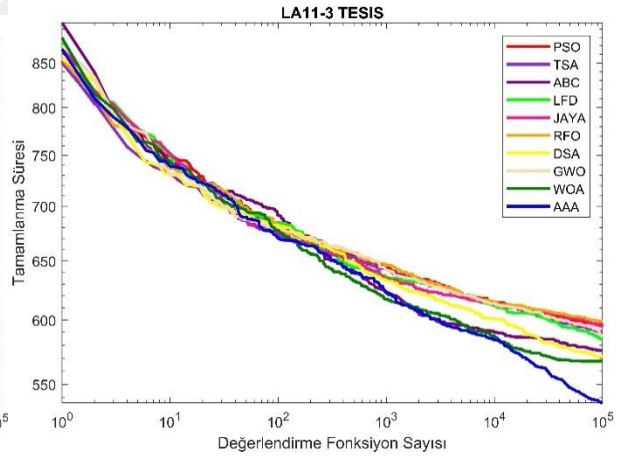


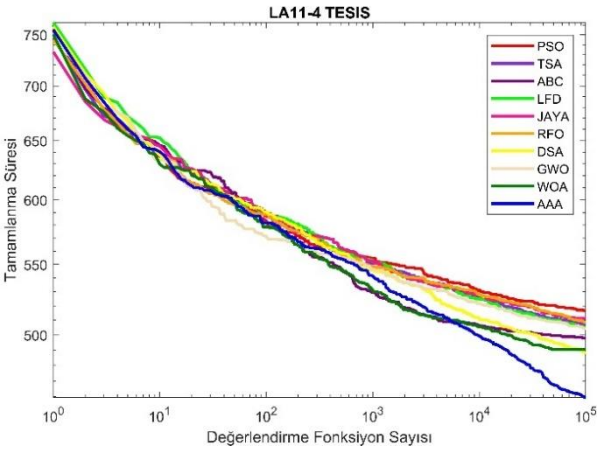

FT20-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

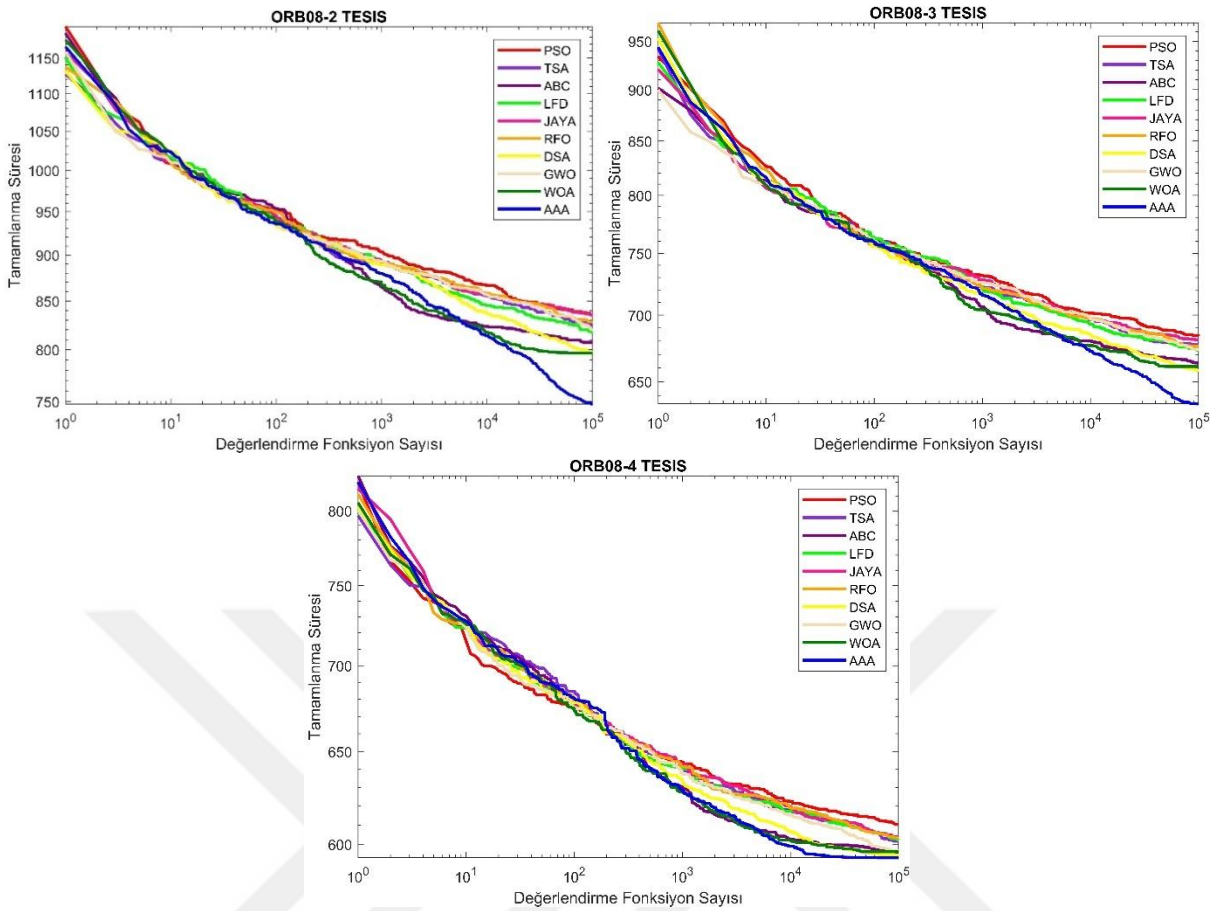
FT20-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


FT20-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


LA11-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

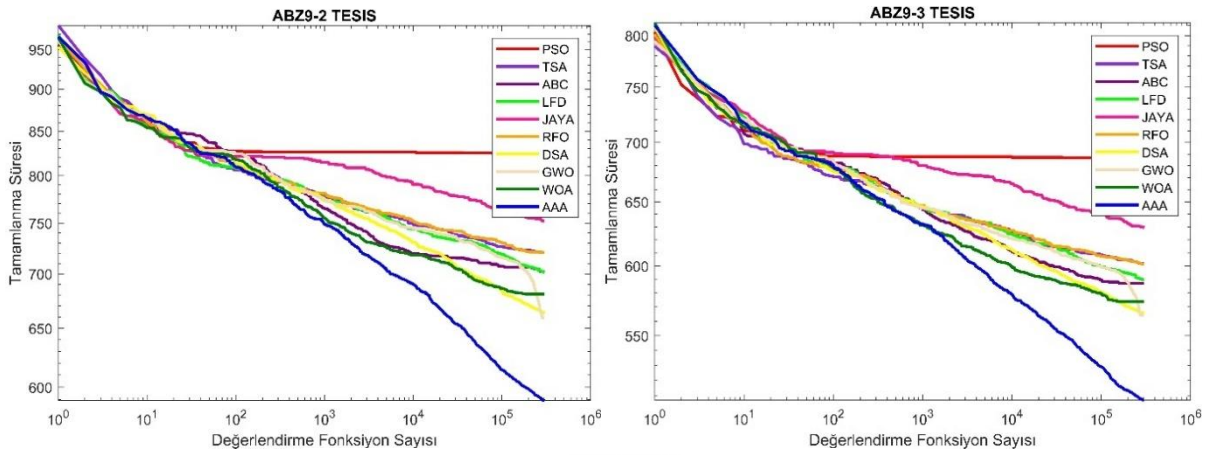

LA11-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


LA11-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

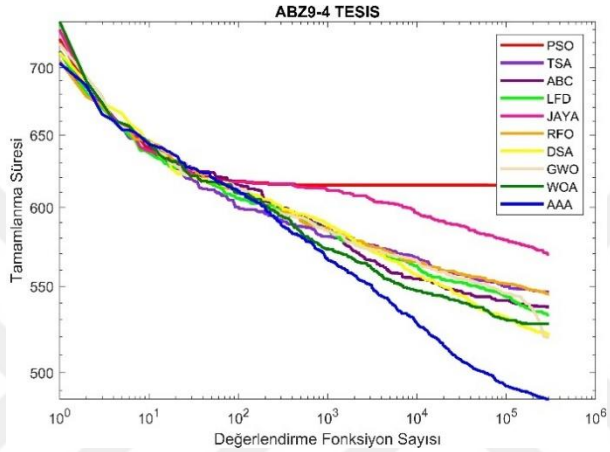


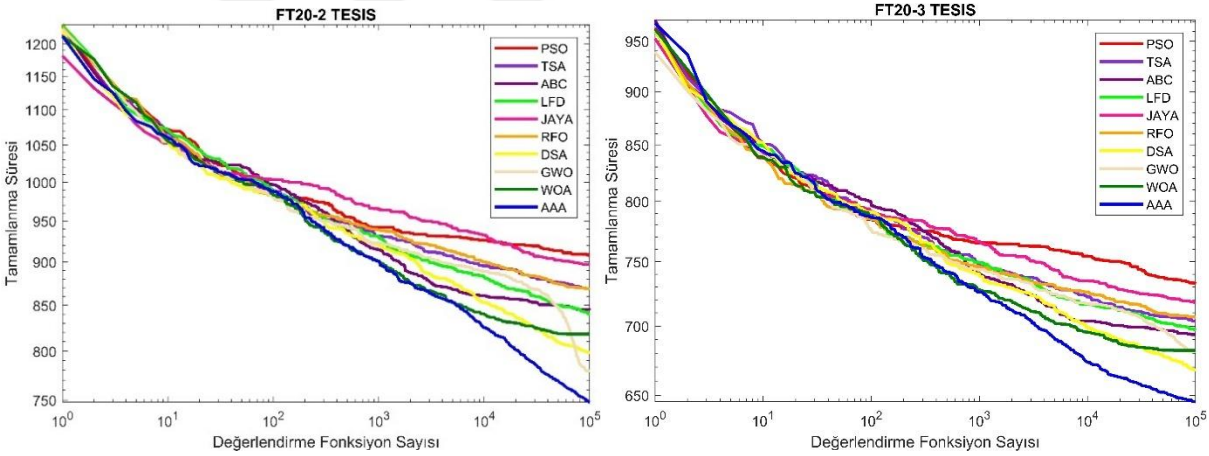
Şekil 6.2 ROV Kodlama Şemasıyla algoritmaların (Abz9, Ft20, La11, Orb08) yakınsama grafikleri (Tesis:2-3-4)

ROV kodlama şemasının uygulanmasıyla 48 karşılaştırma problemi arasından örneklem olarak alınan 4 problemden elde edilen veriler Şekil 6.2 ile sunulmuştur. Verileri inceleyecek olursak bütün algoritmaların birbirine yakın sonuçlar gösterdiği fakat günün sonunda özellikle ABZ09 probleminde AAA algoritmasının iyi bir sıçrama yaparak en iyi sonuç verdiği görülmüştür.

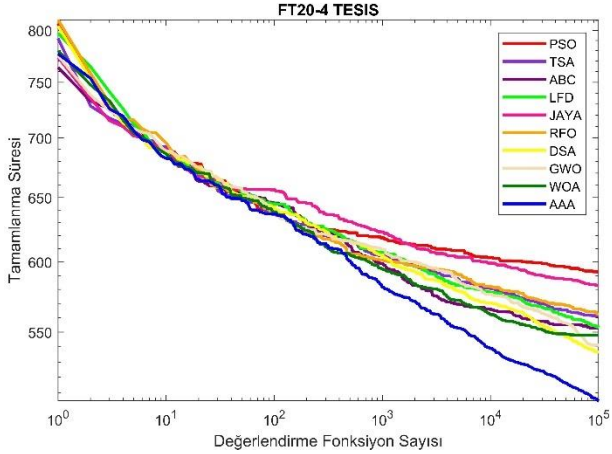

ABZ9-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

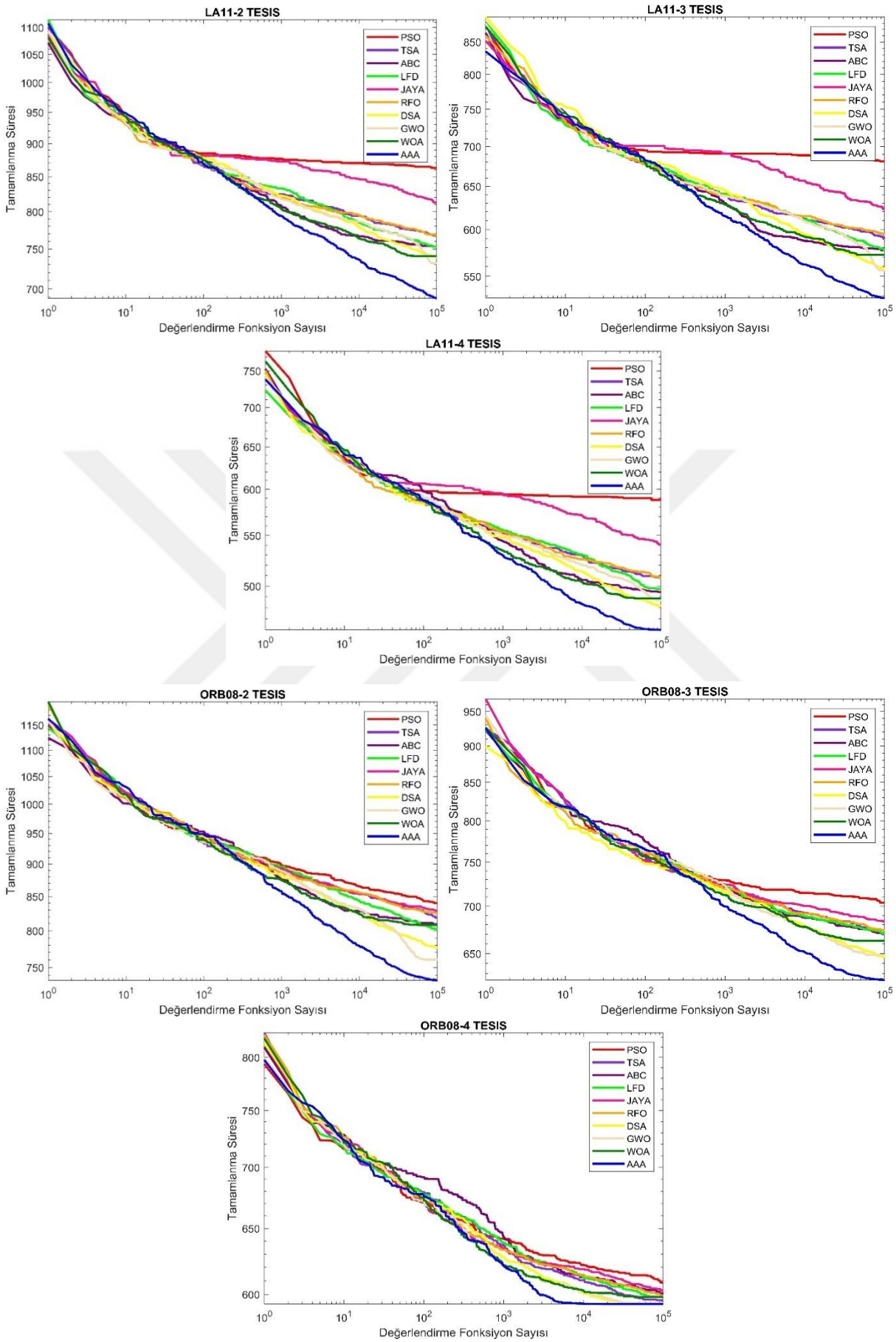
ABZ9-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


ABZ9-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


FT20-2 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

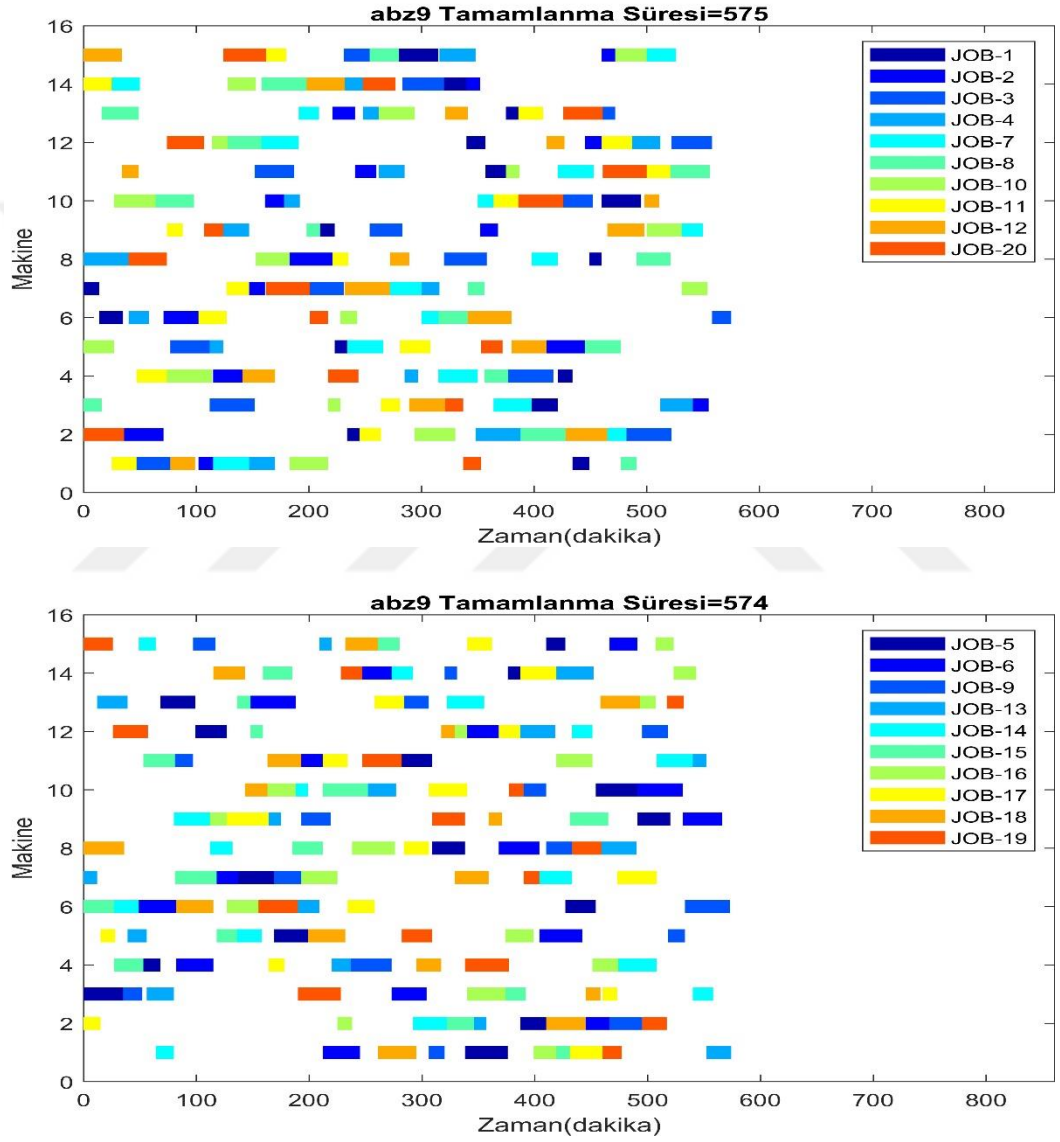
FT20-3 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA


FT20-4 TESIS
Tamamlanma Süresi
Değerlendirme Fonksiyon Sayısı
PSO
TSA
ABC
LFD
JAYA
RFO
DSA
GWO
WOA
AAA

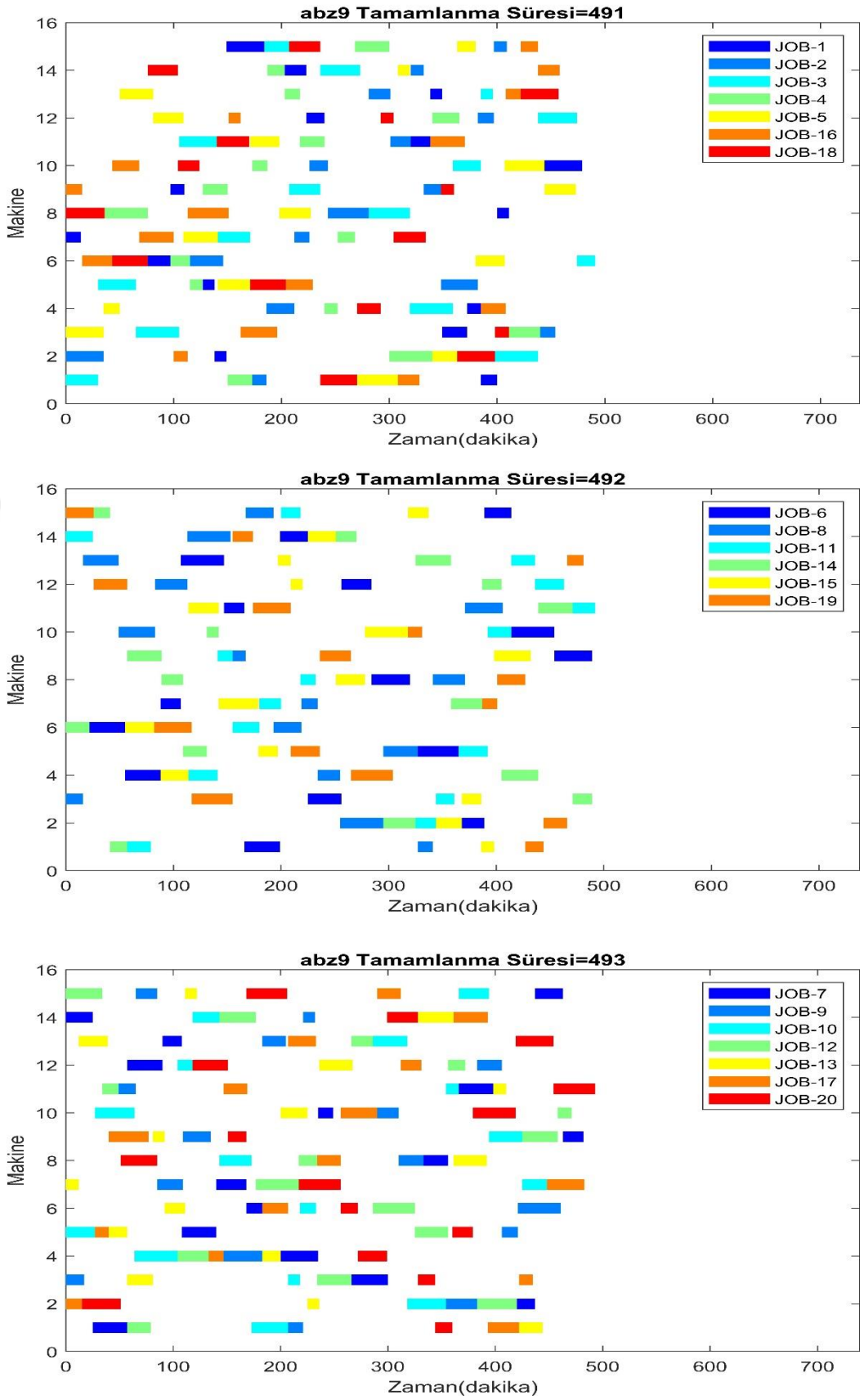


Şekil 6.3 SPV Kodlama Şemasıyla algoritmaların (Abz9, Ft20, La11, Orb08) yakınsama grafikleri (Tesis:2-3-4)

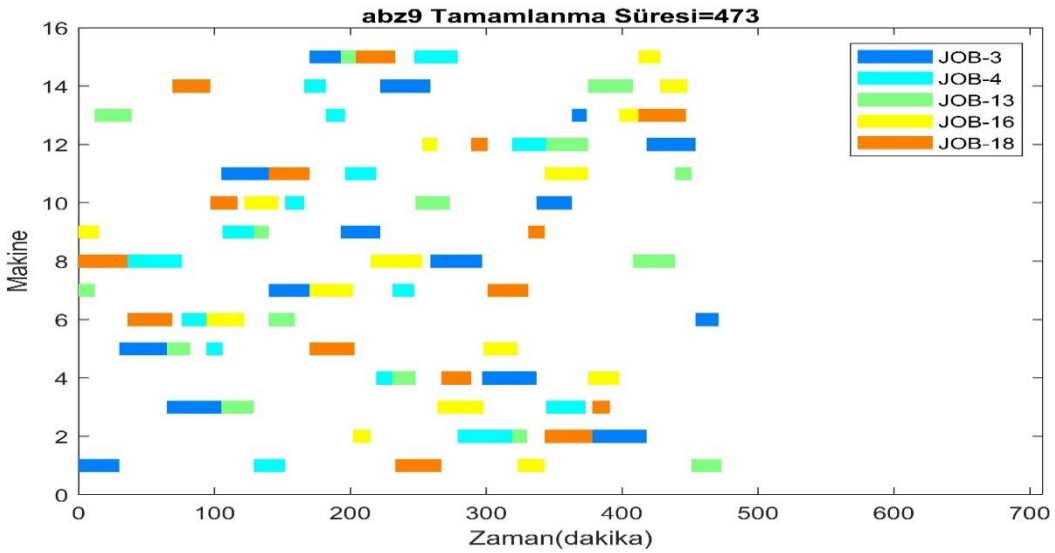
SPV kodlama şemasının uygulanmasıyla 48 karşılaştırma problemi arasından örneklem olarak alınan 4 problemden elde edilen veriler, Şekil 6.3 ile sunulmuştur. Veriler genel olarak incelendiğinde ABZ09 ve LA11 problemlerinde PSO ve JAYA algoritmaları güçlü bir performans gösterememiş olup, iyi bir yakınsama yapamamışlardır. Problemlerin hepsinde AAA algoritması özellikle ABZ09 probleminde iyi bir sıçrama yaparak en iyi yakınsamayı sağlamıştır. Bu da AAA'nın güçlü Keşfetme ve Kullanma mekanizmalarına sahip olduğunu göstermektedir.

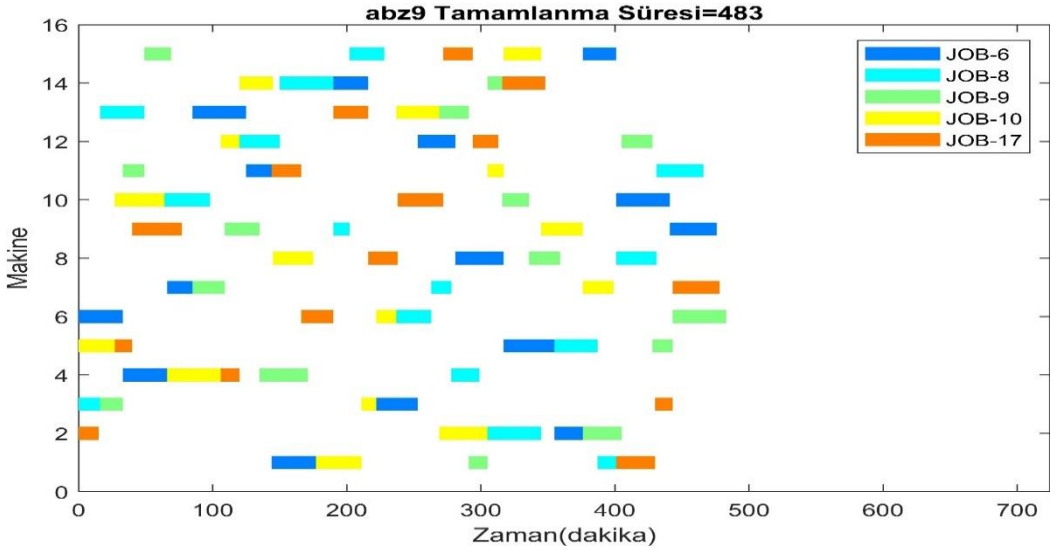


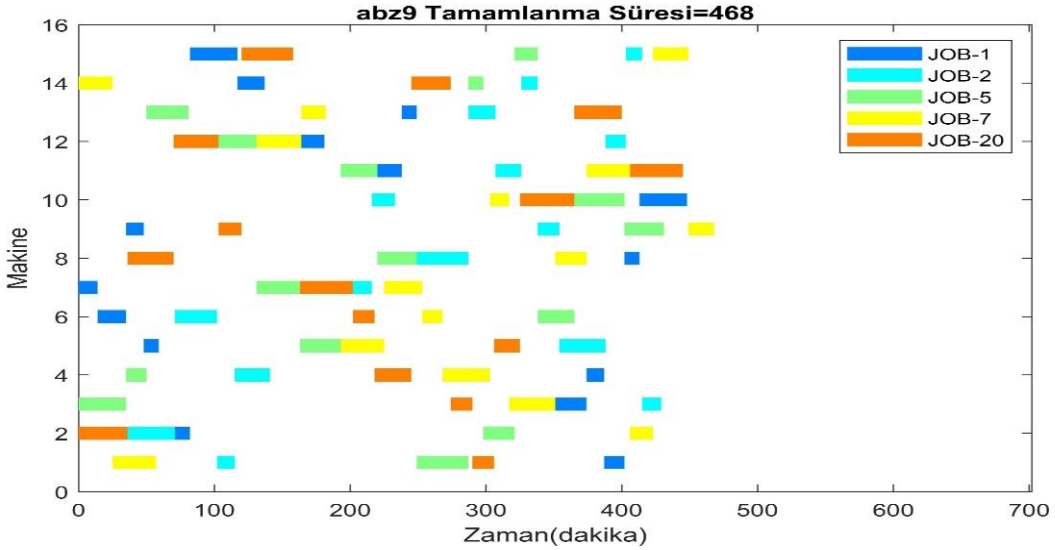
Şekil 6.4 Abz09 Probleminin AAA Algoritması ve RK Kodlama Şeması ile 2 Tesisli Gantt Şeması

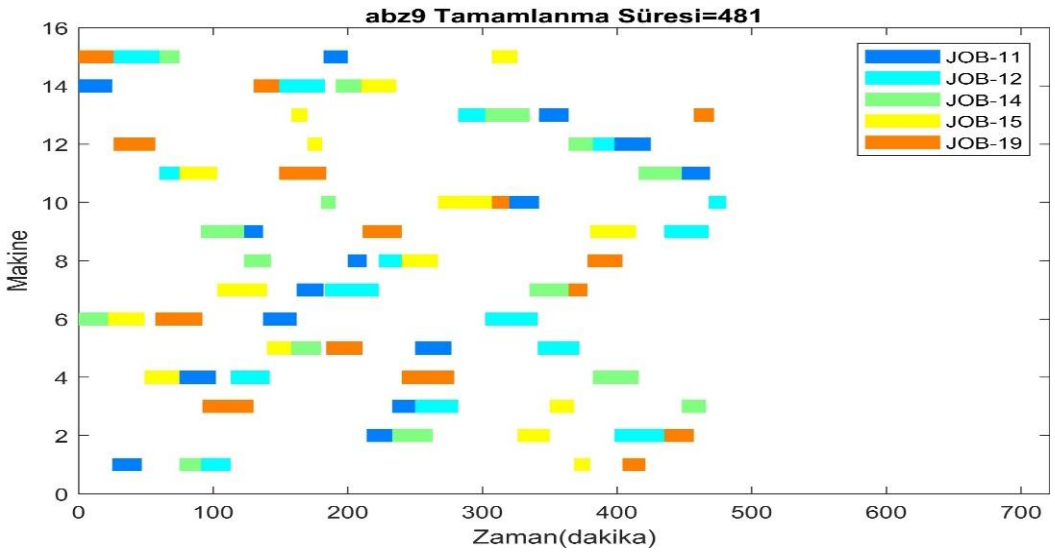


Şekil 6.5 Abz09 Probleminin AAA Algoritması ve RK Kodlama Şeması ile 3 Tesisli Gantt Şeması


abz9 Tamamlanma Süresi=473
JOB-3
JOB-4
JOB-13
JOB-16
JOB-18
Makine
Zaman(dakika)


abz9 Tamamlanma Süresi=483
JOB-6
JOB-8
JOB-9
JOB-10
JOB-17
Makine
Zaman(dakika)


abz9 Tamamlanma Süresi=468
JOB-1
JOB-2
JOB-5
JOB-7
JOB-20
Makine
Zaman(dakika)



Şekil 6.6 Abz09 Probleminin AAA Algoritması ve RK Kodlama Şeması ile 4 Tesisli Gantt Şeması

Gantt şeması, 20. Yüzyılın başlarında Henry L. Gantt tarafından önerilmiştir. Gantt şeması, bir sürecin zaman içerisinde dağılımını gösteren bir grafiktir. Bu çalışmada üretim sürecini görsel olarak anlamlı bir şekilde sunmak için Abz09 probleminin AAA algoritması ve RK kodlama şeması ile 2-3-4 tesisli halini Gantt şeması üzerinde gösterimi yapılmıştır. Şekil 6.4, 6.5, 6.6'da görüldüğü üzere iş parçalarının tesislere olan dağılımını, işleme başlama zamanı, makinede geçen işlem süresi ve sırası sunulmuştur. Tesis sayısı arttıkça problemin tamamlanma süresinde azalmalar gözüktüğü gibi tesis sayısı bütününde bakıldığında her tesisin birbirine yakın sonuç göstermesi sürece genel olarak bakıldığında işin tamamlanma süresini de minimize ettiği ve amaca ulaşıldığı görülmüştür. Fakat tesis sayısı arttığında makinelerdeki operasyonlar arasında boşlukların olduğu da görülmektedir. Çok amaçlı optimizasyon teknikleri kullanılarak, buradaki boşlukların azaltılması sağlanabilir.

7. SONUÇ VE ÖNERİLER

Teknolojinin gelişmesiyle birlikte küreselleşen dünyada, üretim yapan şirketlerin arasındaki rekabet gün geçtikçe artmaktadır. Bu şirketler rekabete devam edebilmek için üretim süreçlerini verimli hale getirmek zorunda kalmaktadır. Şirketler için maliyet azaltma, üretimi hızlandırma, tüketimden tasarruf etme gibi konular ön plan çıkmaktadır. Bu konularda iyileştirmeler yapabilmek için şirketler operasyonel iş planları hazırlamaları gerekmektedir. Böylece mevcut işler, eldeki makinelerle efektif bir şekilde tamamlanabilmektedir. İş atölyesi çizelgeleme problemi (JSSP) bu planlama ihtiyacının karşılanması için kullanılan NP-Hard bir optimizasyon problemidir. Ayrıca farklı konumdaki ve aynı özellikteki birden fazla tesisin kullanıldığı Dağıtık İş atölyesi çizelgeleme problemi (DJSSP) de daha kısa zamanda üretim yapabilmek için ele alınan bir optimizasyon problemidir.

Bu tez çalışmasında, DJSSP problemleri iyi bilinen 10 farklı metasezgisel yaklaşım kullanılarak çözümlenmiştir. Ayrıca bu problemler için literatürde yer alan 3 kodlama şeması kullanılmıştır. Her bir metasezgisel yaklaşımın 3 kodlama şemasındaki performansı karşılaştırılmıştır. DJSSP problemi ele alınırken üretim sürecinin minimuma indirgenmesine odaklanılmıştır. Metasezgisel algoritmalar kullanılarak minimuma düşürülmesi beklenen üretim süreci, farklı özelliklere sahip 48 adet kıyaslama problemi üzerinde optimize edilmiştir. Uygulanan algoritma ve kodlama şeması tesis sayısına göre değişkenlik göstermiş olup tesis sayısı 2 ile 4 arasında ayarlanmıştır.

Yapılan deneysel çalışmalar neticesinde elde edilen sonuçlar detaylı bir şekilde ele alınarak bu tez çalışmasında sunulmuştur. Kodlama şemalarına göre kıyaslama problemleri ayrı ayrı incelendiğinde en iyi performans gösteren algoritma AAA algoritması olmuştur. Bu çalışmanın gelecekteki adımları olarak daha fazla problem veri setinde yeni çıkan algoritmalar denenerek daha iyi sonuç veren farklı algoritmalar bulunabileceği gibi farklı kodlama şemaları da önerilerek daha iyi sonuçlar elde edilecek yaklaşımlar ortaya konulabilir.

KAYNAKLAR

Adams, J., Balas, E. ve Zawack, D., 1988, The shifting bottleneck procedure for job shop scheduling, *Management science*, 34 (3), 391-401.

Applegate, D. ve Cook, W., 1991, A computational study of the job-shop scheduling problem, *ORSA journal on computing*, 3 (2), 149-156.

Bean, J. C., 1994, Genetic algorithms and random keys for sequencing and optimization, *ORSA journal on computing*, 6 (2), 154-160.

Chaouch, I., Driss, O. B. ve Ghedira, K., 2017, A survey of optimization techniques for distributed job shop scheduling problems in multi-factories, *Cybernetics and Mathematics Applications in Intelligent Systems: Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 2 6*, 369-378.

Chaouch, I., Driss, O. B. ve Ghedira, K., 2020, A review of job shop scheduling problems in multi-factories, *International Journal of Operational Research*, 38 (2), 147-165.

Cheng, R., Gen, M. ve Tsujimura, Y., 1996, A tutorial survey of job-shop scheduling problems using genetic algorithms—I. Representation, *Computers & industrial engineering*, 30 (4), 983-997.

Cheng, R., Gen, M. ve Tsujimura, Y., 1999, A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies, *Computers & industrial engineering*, 36 (2), 343-364.

Chipperfield, A. ve Fleming, P., 1997, More integrated gas turbine engine controller design, *Second International Conference On Genetic Algorithms In Engineering Systems: Innovations And Applications*, 357-363.

Civicioglu, P., 2012, Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm, *Computers & Geosciences*, 46, 229-247.

Corsini, A., Porrello, A., Calderara, S. ve Dell'Amico, M., 2024, Self-labeling the job shop scheduling problem, *arXiv preprint arXiv:2401.11849*.

Crowston, W. B., Glover, F., Thompson, G. ve Trawick, J., 1963, Probabilistic and parametric learning combinations of local job shop scheduling rules, *ONR Research Memorandum*.

De Giovanni, L. ve Pezzella, F., 2010, An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, *European journal of operational research*, 200 (2), 395-408.

Du, Y., Li, J.-q., Luo, C. ve Meng, L.-l., 2021, A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations, *Swarm and Evolutionary Computation*, 62, 100861.

Gonçalves, J. F., de Magalhães Mendes, J. J. ve Resende, M. c. G., 2005, A hybrid genetic algorithm for the job shop scheduling problem, *European journal of operational research*, 167 (1), 77-95.

Grimes, D. ve Hebrard, E., 2010, Job shop scheduling with setup times and maximal time-lags: A simple constraint programming approach, *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, 147-161.

Houssein, E. H., Saad, M. R., Hashim, F. A., Shaban, H. ve Hassaballah, M., 2020, Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems, *Engineering Applications of Artificial Intelligence*, 94, 103731.

Huang, Y. ve Yao, X., 2012, Planning and scheduling of multiple flexible-shops based on analytical target cascading and particle swarm optimization, *Journal of Central South University (Science and Technology)*, 43 (1), 151-158.

Jia, H., Fuh, J. Y., Nee, A. Y. ve Zhang, Y., 2002, Web-based multi-functional scheduling system for a distributed manufacturing environment, *Concurrent Engineering*, 10 (1), 27-39.

Jia, H., Nee, A. Y., Fuh, J. Y. ve Zhang, Y., 2003, A modified genetic algorithm for distributed scheduling problems, *Journal of Intelligent Manufacturing*, 14, 351-362.

Jia, H., Fuh, J. Y., Nee, A. Y. ve Zhang, Y., 2007, Integration of genetic algorithm and Gantt chart for job shop scheduling in distributed manufacturing systems, *Computers & industrial engineering*, 53 (2), 313-320.

Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, *Technical report-tr06, Erciyes university, engineering faculty, computer …*.

Kennedy, J. ve Eberhart, R., 1995, Particle swarm optimization, *Proceedings of ICNN'95-international conference on neural networks*, 1942-1948.

Kiran, M. S., 2015, TSA: Tree-seed algorithm for continuous optimization, *Expert Systems with Applications*, 42 (19), 6686-6698.

Kotary, J., Fioretto, F. ve Van Hentenryck, P., 2022, Fast approximations for job shop scheduling: A lagrangian dual deep learning method, *Proceedings of the AAAI Conference on Artificial Intelligence*, 7239-7246.

Lawrence, S., 1984, Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement), *Graduate School of Industrial Administration, Carnegie-Mellon University*.

Li, J.-Q., Duan, P., Cao, J., Lin, X.-P. ve Han, Y.-Y., 2018, A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria, *IEEE Access*, 6, 58883-58897.

Li, Y., Huang, W., Wu, R. ve Guo, K., 2020, An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem, *Applied Soft Computing*, 95, 106544.

Liu, M., Yao, X. ve Li, Y., 2020, Hybrid whale optimization algorithm enhanced with Lévy flight and differential evolution for job shop scheduling problems, *Applied Soft Computing*, 87, 105954.

Luo, Q., Deng, Q., Gong, G., Zhang, L., Han, W. ve Li, K., 2020, An efficient memetic algorithm for distributed flexible job shop scheduling problem with transfers, *Expert Systems with Applications*, 160, 113721.

Marzouki, B., Driss, O. B. ve Ghédira, K., 2018, Solving distributed and flexible job shop scheduling problem using a chemical reaction optimization metaheuristic, *Procedia Computer Science*, 126, 1424-1433.

Meng, L., Zhang, C., Ren, Y., Zhang, B. ve Lv, C., 2020, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, *Computers & industrial engineering*, 142, 106347.

Mirjalili, S., Mirjalili, S. M. ve Lewis, A., 2014, Grey wolf optimizer, *Advances in engineering software*, 69, 46-61.

Mirjalili, S. ve Lewis, A., 2016, The whale optimization algorithm, *Advances in engineering software*, 95, 51-67.

Mohammed, H. ve Rashid, T., 2023, FOX: a FOX-inspired optimization algorithm, *Applied Intelligence*, 53 (1), 1030-1050.

Naderi, B. ve Azab, A., 2014, Modeling and heuristics for scheduling of distributed job shops, *Expert Systems with Applications*, 41 (17), 7754-7763.

Naderi, B. ve Azab, A., 2015, An improved model and novel simulated annealing for distributed job shop problems, *The International Journal of Advanced Manufacturing Technology*, 81, 693-703.

Rao, R. V. ve Patel, V., 2012, International Journal of Industrial Engineering Computations, *International Journal of Industrial Engineering Computations*, 3, 519-524.

Şahman, M. A., 2021, A discrete spotted hyena optimizer for solving distributed job shop scheduling problems, *Applied Soft Computing*, 106, 107349.

Şahman, M. A., 2022, Advanced Tree-Seed Algorithm for Large Sized Job Shop Scheduling Problems, *Gazi Journal of Engineering Sciences (GJES)/Gazi Mühendislik Bilimleri Dergisi*, 8 (2).

Şahman, M. A. ve Korkmaz, S., 2022, Discrete artificial algae algorithm for solving job-shop scheduling problems, *Knowledge-Based Systems*, 256, 109711.

Şahman, M. A. ve Dündar, A. O., 2024, Comparative Performance Analysis of Meta-Heuristic Algorithms in Distributed Job Shop Scheduling, *2024 59th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)*, 1-4.

Tasgetiren, M. F., Sevkli, M., Liang, Y.-C. ve Yenisey, M. M., 2006, A particle swarm optimization and differential evolution algorithms for job shop scheduling problem, *International Journal of Operations Research*, 3 (2), 120-135.

Toma, L., Zajac, M. ve Störl, U., 2024, Solving distributed flexible job shop scheduling problems in the wool textile industry with quantum annealing, *arXiv preprint arXiv:2403.06699*.

Uymaz, S. A., Tezel, G. ve Yel, E., 2015, Artificial algae algorithm (AAA) for nonlinear global optimization, *Applied Soft Computing*, 31, 153-171.

Wu, X., Liu, X. ve Zhao, N., 2019, An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem, *Memetic Computing*, 11, 335-355.

Ziaee, M., 2014, A heuristic algorithm for the distributed and flexible job-shop scheduling problem, *The Journal of Supercomputing*, 67, 69-83.