

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ



**BULUT SİSTEMLERDE GÖREV ÇİZELGELEME İÇİN
YENİLİKÇİ TEK VE ÇOK AMAÇLI METASEZGİSEL
YÖNTEMLERİN GELİŞTİRİLMESİ**

Cebrail BARUT

Doktora Tezi

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ARALIK 2024

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı

Doktora Tezi

**BULUT SİSTEMLERDE GÖREV ÇİZELGELEME İÇİN YENİLİKÇİ
TEK VE ÇOK AMAÇLI METASEZGİSEL YÖNTEMLERİN
GELİŞTİRİLMESİ**

Tez Yazarı
Cebrail BARUT

Danışman
Doç. Dr. Güngör YILDIRIM

ARALIK 2024
ELAZIĞ

T.C.
FIRAT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalı

Doktora Tezi

Başlığı:	Bulut Sistemlerde Görev Çizelgeleme İçin Yenilikçi Tek ve Çok Amaçlı Metasezgisel Yöntemlerin Geliştirilmesi
Yazarı:	Cebrail BARUT
İlk Teslim Tarihi:	12.11.2024
Savunma Tarihi:	12.12.2024

TEZ ONAYI

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına göre hazırlanan bu tez aşağıda imzaları bulunan jüri üyeleri tarafından değerlendirilmiş ve akademik dinleyicilere açık yapılan savunma sonucunda OYBİRLİĞİ ile kabul edilmiştir.

Danışman:	Doç. Dr. Güngör YILDIRIM Fırat Üniversitesi, Mühendislik Fakültesi	İmza Onayladım
Başkan:	Prof. Dr. Yakup DEMİR Fırat Üniversitesi, Mühendislik Fakültesi	Onayladım
Üye:	Prof. Dr. İlhan AYDIN Fırat Üniversitesi, Mühendislik Fakültesi	Onayladım
Üye:	Doç. Dr. Muhammed YILDIRIM Turgut Özal Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi	Onayladım
Üye:	Dr. Öğr. Üyesi Alper KILIÇ Konya Teknik Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi	Onayladım

Bu tez, Enstitü Yönetim Kurulunun/...../20..... tarihli toplantısında tescillenmiştir.

İmza

Prof. Dr. Burhan ERGEN
Enstitü Müdürü

BEYAN

Fırat Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım “ Bulut Sistemlerde Görev Çizelgeleme İçin Yenilikçi Tek ve Çok Amaçlı Metasezgisel Yöntemlerin Geliştirilmesi ” Başlıklı Doktora Tezimin içindeki bütün bilgilerin doğru olduğunu, bilgilerin üretilmesi ve sunulmasında bilimsel etik kurallarına uygun davrandığımı, kullandığım bütün kaynakları atıf yaparak belirttiğimi, maddi ve manevi desteği olan tüm kurum/kuruluş ve kişileri belirttiğimi, burada sunduğum veri ve bilgileri unvan almak amacıyla daha önce hiçbir şekilde kullanmadığımı beyan ederim.

12.12.2024

Cebrail BARUT



ÖNSÖZ

Bulut sistemler; internet aracılığıyla kaynak paylaşımı, donanımsal ve yazılımsal sanallaştırma, veri depolama, yönetme ve işleme hizmetleri sunan bir teknoloji altyapısıdır. Kullanıcılar, kendi donanım altyapılarına bağlı kalmadan bu sistemlere uzaktan erişebilir ve işlemlerini gerçekleştirebilirler. Bulut servislerinin son kullanıcılara sunulması sanallaştırma mekanizması ile yürütülür. Bu mekanizma hem bulut kaynaklarının verimli kullanılmasını hem de müşterilerin ihtiyaçlarına göre esnek ve ekonomik çözümlere ulaşmasına yardım eder. Ayrıca sanallaştırma mekanizması dinamik işletme şartları oluşturduğu için bulut yönetimi daha zorlaşır. Bu zorluklardan birisi de dinamik senaryolara göre yine dinamik görev çizelgelemelerin yapılabilmesidir.

Kullanıcı isteklerinin büyük ölçekli bulut sistemlerdeki sunucularda yürütülmesi, karmaşık ve zorlu bir süreçtir. Bu süreçte seçilen görev çizelgeleme stratejisi hem sağlayıcıyı hem de son kullanıcıyı doğrudan etkiler. Başarılı bir görev çizelgeleme stratejisi, sağlayıcı açısından enerji tasarrufu ve verimli kaynak kullanımı gibi avantajları getirirken son kullanıcılar için ise daha ekonomik ve düşük cevap sürelili hizmetler almasını sağlar. Bununla birlikte kullanılacak strateji, müşteri tipi, anlık sanal makine durumlarının değişken olabilmesi, yük dengeleyicilerin kaynak kullanımını anlık değiştirebilmesi gibi parametreleri her zaman dikkate alınmalıdır ki söz konusu bu parametreler, geleneksel bilgi işlem birimlerinde veya veri merkezlerinde sabit olmaktadır. Hizmet Düzeyi Sözleşmesi (Service Level Agreement-SLA) şartlarının sağlanamaması ve müşteri memnuniyetsizliği gibi firma itibarını sarsabilecek kadar önemli sorunlara yol açabilecek bu kritik sürecin yürütülmesi mühendisler ve iş dünyası kadar akademik dünyanın da odağındadır.

Bu tez çalışmasında, bulut sistemlerde görev çizelgeleme problemi incelenmiş ve bu amaçla iki farklı yöntem önerilmiştir. Önerilen ilk yöntem, tek amaçlı bir optimizasyon yaklaşımıdır ve kullanıcı taleplerinin bulutta minimum sürede çalıştırılmasını amaçlayan kural tabanlı metasezgisel bir mekanizma kullanır. İkinci yöntem ise çok amaçlı optimizasyon temelli olup görevlerin bulutta icra edilmesi esnasında sistemin harcadığı enerjiyi ve görevlerin icra süresinin minimizasyonunu hedefler.

Bu çalışmanın yapılmasında yaptığı yönlendirme ve katkılardan dolayı değerli hocam sayın Doç. Dr. Güngör YILDIRIM'a, tez izleme komitesi üyeleri Prof. Dr. Yakup DEMİR'e, Prof. Dr. İlhan AYDIN'a ve Prof. Dr. Yetkin TATAR'a teşekkür ve minnetimi özellikle belirtmek isterim. Ayrıca maddi ve manevi desteklerini yıllarca benden esirgemeyen başta değerli anneme, eşime ve evlatlarıma teşekkürü bir borç bilirim.

Cebrail BARUT

ELAZIĞ, 2024

İÇİNDEKİLER

Sayfa

ÖNSÖZ	iv
İÇİNDEKİLER.....	v
ÖZET	vii
ABSTRACT	viii
ŞEKİLLER LİSTESİ.....	ix
TABLOLAR LİSTESİ.....	x
SİMGELER	xi
KISALTMALAR.....	xiii
1. GİRİŞ.....	1
1.1. Tezin Amacı.....	1
1.2. Literatür Araştırması	2
1.2.1. Sezgisel Yöntemler	3
1.2.2. Oyun Teorisine Dayalı Yöntemler.....	4
1.2.3. Öğrenme Tabanlı Yöntemler	5
1.2.4. Hibrit Yöntemler.....	5
1.2.5. Metasezgisel Yöntemler	9
1.3. Tezin Yapısı.....	20
2. GENEL BİLGİLER.....	21
2.1. Bulut Bilişim	21
2.2. Çizelgeleme Problemleri.....	23
2.3. Optimizasyon	25
2.3.1. Tek Amaçlı Optimizasyon.....	26
2.3.2. Çok Amaçlı Optimizasyon.....	27
2.4. Metasezgisel Yöntemler.....	28
2.5. Bulut Sistemlerde Görev Çizelgeleme	28
2.6. Bulut Bilişimde Metasezgisel Tabanlı Görev Değiştirmenin Dezavantajları.....	31
3. KURAL TABANLI AKILLI METASEZGİSEL GÖREV ÇİZELGELEME.....	32
3.1. Irmts'nin Temel Prensipleri	32
3.2. PSO Temelli Görev çizelgeleme	33
3.3. K-Means.....	37
3.4. Ayçiçeği Optimizasyonuna Dayalı Kural Çıkarımı	39
3.5. Çevrimiçi Aşama	43
3.6. Deney Sonuçları ve Tartışma	44
3.7. Bölüm Sonu Değerlendirmesi	52
4. ÇOK AMAÇLI HİBRİT YÖNTEM	53
4.1. NSGA-2 (Nondominated Sorting Genetic Algorithm-2)	53
4.2. SPEA-2 (Strength Pareto Evolutionary Algorithm 2)	56
4.3. Önerilen Hibrit Paralel MOO Görev Planlama Yaklaşımı.....	58
4.4. Deneyler ve Tartışma	60
4.5. Bölüm Sonu Değerlendirmesi	69
5. TARTIŞMA.....	70
6. SONUÇLAR.....	72

ÖNERİLER	73
KAYNAKLAR.....	74
ÖZGEÇMİŞ	



ÖZET

Bulut Sistemlerde Görev Çizelgeleme İçin Yenilikçi Tek ve Çok Amaçlı Metasezgisel Yöntemlerin Geliştirilmesi

Cebrail BARUT

Doktora Tezi

FIRAT ÜNİVERSİTESİ

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Aralık 2024, Sayfa: xvi + 79

Klasik bilgi işlem birimleri ve veri merkezlerinden farklı olarak bulut sistemler oldukça dinamik bir yapıya sahiptir. Bu dinamizmi sağlayan ana faktörler; çok sayıda ve farklı taleplere sahip müşteri tipi, anlık değişen talep oranına göre aktif sanal makine sayısının değişebilmesi ve güvenlik gereksinimleridir. Bulut sistemlerce sunulan yazılım ve donanım servisleri bu dinamikliğe sanallaştırma teknolojilerinin sağlamış olduğu avantajlar ile cevap verirler. Ancak sanallaştırma teknolojileri, oluşacak her problemi çözmek için yeterli değildir. Kullanıcılardan gelen çok sayıda görev talebinin verimsiz bir şekilde sanal kaynaklara yönlendirilmesi, sağlayıcıya yüksek enerji maliyeti ve kaynak israfı, son kullanıcıya ise pahalı hizmet alımı ve memnuniyetsizlik olarak geri döner. Bulut sistemlerde başarılı görev dağılımını sağlayan iyi düşünülmüş bir görev çizelgeleme stratejisi, bu bakımdan, hayati bir öneme sahiptir. Oldukça karmaşık olan bulut sistemlerde görev çizelgeleme problemi için, hem pratikte hem de teoride son yıllarda ciddi çalışmalar yapılmıştır.

Bu tez çalışması bulut sistemlerin dinamik doğasını dikkate alan farklı çizelgeleme stratejileri için çalışılmıştır. Bu amaçla, bulut sistemlerde görev çizelgeleme için iki yenilikçi optimizasyon yöntemi önerilmiştir. Birinci yöntem, tek amaçlı bir optimizasyon yaklaşımıdır ve bulut sistemlerde görevlerin icra süresini minimize etmeyi amaçlar. Literatürde var olan yöntemlerden farklı olarak kural çıkarım tabanlı bir metodoloji izleyen bu yöntem, daha önce karşılaşılan senaryolardaki görev çizelgeleme çözümleri ve mevcut senaryo arasındaki benzerliği tespit ederek optimum çizelgelemeyi yapmaktadır. Önerilen bu yöntemin başarısı, hem klasik çizelgeleme yöntemleri hem de diğer metasezgisel yöntemlerin bulmuş olduğu çözümler ile karşılaştırmalı olarak sunulmuştur. Deneylerde elde edilen sonuçlar, önerilen yöntemin karar verme süreci yönünden diğer yöntemlere göre yaklaşık 18 kat daha hızlı olduğunu göstermiştir. Tezde önerilen ikinci yöntem ise görevlerin icra süresi ve enerji tüketimi amaçlarının eş zamanlı optimizasyonunu hedefler. Bu yöntemde NSGA-2 ve SPEA2 gibi iki farklı metasezgisel algoritmanın paralel ve hibrit çalışmasına dayalı bir yaklaşım kullanılmıştır. Önerilen yöntem iki algoritmayı eş zamanlı çalıştırarak birbirlerine olan üstünlüklerine göre seçim yapmaktadır. Önerilen yöntem NSGA-2 ve SPEA2 algoritmalarına göre daha başarılı sonuçlar almıştır.

Tez çalışması ile bulut sistemlerde görev çizelgeleme üzerine yenilikçi ve etkili metasezgisel yaklaşımlar sunulmuştur. Arşiv temelli bu bakış açısı, sadece bulut sistemler için değil aynı zamanda benzer dinamik sistemlere de adapte edilebilecek yapıdadır. Tezde önerilen yöntemlerin araştırmacılar için yeni çözüm fikirlerinin geliştirilmesine katkı vereceği ve farklı alanlarda bu yaklaşımların etkili olacağı düşünülmektedir.

Anahtar Kelimeler: Bulut Sistemler, Görev Çizelgeleme, Kural Tabanlı görev çizelgeleme, Çok amaçlı Hibrit Optimizasyon

ABSTRACT

Development of Innovative Single and Multi-Objective Metaheuristic Methods for Task Scheduling in Cloud Systems

Cebrail BARUT

Ph.D. Thesis

FIRAT UNIVERSITY
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

December 2024, Pages: xvi + 79

Unlike traditional computing units and data centers, cloud systems have a highly dynamic structure. The main factors that provide this dynamism are; customer types with many and different demands, the number of active virtual machines can change according to the instantly changing demand rate, and security requirements. Software and hardware services offered by cloud systems respond to this dynamism with the advantages provided by virtualization technologies. However, virtualization technologies are not sufficient to solve every problem that may arise. Inefficiently directing a large number of task requests from users to virtual resources results in high energy costs and resource waste for the provider, and expensive service purchases and dissatisfaction for the end user. A well thought out task scheduling strategy that ensures successful task distribution in cloud systems is of vital importance in this respect. Serious studies have been carried out in recent years for the task scheduling problem in cloud systems, which is quite complex, both in practice and in theory.

This thesis study is studied for different scheduling strategies considering the dynamic nature of cloud systems. For this purpose, two innovative optimization methods are proposed for task scheduling in cloud systems. The first method is a single-objective optimization approach and aims to minimize the execution time of tasks in cloud systems. Unlike the existing methods in the literature, this method, which follows a rule based methodology, performs optimum scheduling by detecting the similarity between the task scheduling solutions in previously encountered scenarios and the current scenario. The success of this proposed method is presented comparatively with the solutions found by both classical scheduling methods and other metaheuristic methods. The results obtained in the experiments show that the proposed method is approximately 18 times faster than other methods in terms of decision-making process. The second method proposed in the thesis aims at simultaneous optimization of task execution time and energy consumption objectives. In this method, an approach based on parallel and hybrid operation of two different metaheuristic algorithms such as NSGA-2 and SPEA2 is used. The proposed method runs the two algorithms simultaneously and makes a selection according to their superiority over each other. The proposed method has obtained more successful results than the NSGA-2 and SPEA2 algorithms.

Innovative and effective metaheuristic approaches on task scheduling in cloud systems have been presented in this thesis. This archive-based perspective is adaptable not only to cloud systems but also to similar dynamic systems. It is thought that the methods proposed in the thesis will contribute to the development of new solution ideas for researchers and that these approaches will be effective in different areas.

Keywords: Cloud Computing, Task Scheduling, Rule Based Task Scheduling, Multi Objective Hybrid Optimization

ŞEKİLLER LİSTESİ

	Sayfa
Şekil 1.1. Bulut sistemlerde kullanılan görev çizelgeleme türleri.....	3
Şekil 2.1. Bulut sistem mimarisi	21
Şekil 2.2. Bulut türlerinin sınıflandırılması.....	23
Şekil 2.3. Bulut sistemlerde görev çizelgeleme aşamaları	24
Şekil 2.4. Bulut sistemlerdeki çizelgeleyici türleri	25
Şekil 2.5. Metasezgisel yöntemlerin sınıflandırılması	28
Şekil 2.6. Makespan gösterimi.....	30
Şekil 2.7. Metasezgisel görev çizelgeleme	30
Şekil 3.1. IRMTS'nin iç adımları ve aşamaları	33
Şekil 3.2. Kümeleme sonrası çözüm alanı (Sarı: HD, Mor: LD, Yeşil: MD)	39
Şekil 3.3. Kullanılan tüm yöntemlerin ortalama üretim süresi sonuçları	46
Şekil 3.4. Ortalama karar alma süreleri.....	48
Şekil 3.5. CPU kullanımı yönünden yöntemlerin karşılaştırılması	50
Şekil 3.6. RAM kullanımı yönünden yöntemlerin karşılaştırılması	51
Şekil 4.1. Önerilen hibrit yöntemin genel aşamaları.....	59
Şekil 4.2. Senaryo 1 için pareto grafikleri.....	62
Şekil 4.3. Senaryo 2 için pareto grafikleri.....	63
Şekil 4.4. Senaryo 3 için pareto grafikleri.....	64

TABLÖLAR LİSTESİ

	Sayfa
Tablo 3.1. Ham veri setinde beklenen yarı-çevrimdışı görevlendirme senaryoları	36
Tablo 3.2. Her senaryo için PSO tabanlı görev çizelgeleme çözümleri	36
Tablo 3.3. Senaryoların K-Means ile sınıflandırılması.....	38
Tablo 3.4. TP, TN, FP ve FN değerlerinin hesaplanması	41
Tablo 3.5. Kural tabanlı SFO'dan elde edilen örnek kurallar	43
Tablo 3.6. Deney parametreleri	45
Tablo 3.7. IRMTS'nin makespan deneylerinin istatistiksel sonuçları	47
Tablo 4.1. Deneylerde kullanılan senaryolara ait parametreler	60
Tablo 4.2. NSGA-2 ve SPEA2 parametreleri.....	61
Tablo 4.3. Tüm yöntemler için karşılaştırmalı sonuçlar	66
Tablo 4.4. Önerilen yöntem, SPEA2 ve NSGA-2 için deney sonuçları performans metrikleri.....	68

SİMGELER

$\vec{O}_i$	: Güneşe yönelim
$\bar{P}_t$	: t 'ninci iterasyondaki arşiv
$\overrightarrow{X_{i+1}}$	: Adayın güneşe yönelimi
$ \cdot $	: Kümenin eleman sayısı
$\mathcal{E}_j$	: j 'ninci sanal makineye tahsis edilen tüm görevlerin toplam yürütüm süresi
S_i^l, S_i^u	: Adayın iterasyonlar boyunca bulduğu alt ve üst limitler
CD_i	: i 'ninci adayın kalabalık mesafesi
ET_{ji}	: i 'ninci görevin j 'ninci sanal makinedeki yürütüm zamanı
E_{ji}	: i 'ninci görevin j 'ninci sanal makinedeki yürütümü esnasında harcanan enerji miktarı
F_n	: n numaralı cephe
G_{best}	: Sürüde bulunan en iyi çözüm değeri
Itr_{max}	: Maksimum iterasyon
L^l	: Niteliğin veri setindeki minimum değeri
N_{pop}	: Popülasyon boyutu
$\bar{P}$	: Arşiv
P_{best}	: Parçacığın sahip olduğu en iyi çözüm değeri
$P_i(\ X_i + X_{i-1}\)$	: i 'ninci aday için tozlaşma olasılığı
P_t	: t 'ninci iterasyondaki popülasyon
S_i^b	: Hangi çözümlerin kurala dâhil edileceğini gösteren binary vektör
S_v	: Parçacığın pozisyonunun tamsayı hali
TE_j	: j 'ninci sanal makinede tahsis edilen görevlerin icrası için harcanan toplam enerji
U^u	: Niteliğin veri setindeki maksimum değeri
V_p	: Parçacığın hızı
X^*	: Güneş
X_i	: i 'ninci ayçiçeği
X_p	: Parçacığın pozisyonu
c_1, c_2	: Parçacık hareketinin hızlanma katsayısı
d_i	: Diğer adayların hareket adımı
eb_1, eb_2	: Bir ve iki numaralı ebeveynler
e_{ji}	: i 'ninci görev j 'ninci sanal makineye atanıp atanmadığını belirten ikili değer
$\max_{x \in X} f(x)$	: Fonksiyonun maksimum değeri
$\min_{x \in X} f(x)$	: Fonksiyonun minimum değeri
of_1, of_2	: Yeni üretilen bireyler
$rand_1, rand_2$	: Rastgele üretilen sayılar
$s_n^{(U)}, s_n^{(L)}$	: Genin üst ve alt sınırları
$s_n^{(m)}$	: m 'inci popülasyona ait olan n 'inci bireyi
t_i	: i 'ninci görev
v_j	: j 'ninci sanal makine
we_i	: Amacın katsayı değeri
x_L	: Değişkenin alt sınırı
x_U	: Değişkenin üst sınırı
α_n	: Mutasyon büyüklüğü
$+$	: Çoklu küme birleşimi
$>$	: Pareto dominasyonu

$f(.)$	: Amaç fonksiyonu
$h(x, u(x))$	: Eşitlik kısıtları
P	: Popülasyon
S_i	: Ayçiçeği optimizasyon algoritmasında her bir çözüm
$t(.)$	: Rastgele üreteç fonksiyonu
β	: Eşik değeri
ϕ	: Yeni bireylerin ebeveynlerine benzerlik oranı
$F(i)$	: İlgili adayın amaç değeri
$G(x, u(x))$	: Hal denklemlerini içeren diğer eşitlik kısıtları
MS	: Makespan
$P(\delta)$	: Yayılma faktörü
$R(i)$	: Ham amaç değeri
$S(i)$	: İlgili adayın popülasyon içerisindeki dominasyon durumu
SE	: Görev çizelgelemede harcanan toplam enerji
T	: Görev vektörü
V	: Sanal sunucu vektörü
W	: Parçacık atalet katsayısı
$d(p, q)$	: p ve q vektörleri arasındaki öklit mesafesi
$f(x, (u(x)))$	: Amaç fonksiyonu
$g(x, u(x))$	: Eşitsizlik kısıtları
$u(x)$	: Hal denklemi
x	: Tasarım yöneyi
γ	: Karışıklık parametresi
λ	: Atalet katsayısı

KISALTMALAR

ACO	: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)
AEHO	: Uyarlanabilir Fil Sürü Optimizasyonu (Adaptive Elephant Herding Optimization)
AGA	: Uyarlanabilir Genetik Algoritma (Adaptive Genetic Algorithm)
ALO	: Karınca Aslanı Optimize Edici (Ant Lion Optimizer)
ANN	: Yapay Sinir Ağı (Artificial Neural Network)
AO	: Aquila Optimizasyonu (Aquila Optimization)
AOA	: Aritmetik Optimizasyon Tekniği (Arithmetic Optimization Technique)
AWS	: Amazon Web Hizmetleri (Amazon Web Services)
BB-BC	: Büyük Patlama Büyük Çatırtı (Big-Bang Big-Crunch)
BEDTS	: Biyo-İlham Veren Enerji Verimli Dinamik Görev Çizelgeleme (Bio-Inspired Energy Efficient Dynamic Task Scheduling)
BWO	: Siyah Pencere Optimizasyonu (Black Window Optimization)
CA	: Şempanze Algoritması (Chimp Algorithm)
CD	: Kalabalık Mesafesi (Crowding Distance)
COA	: Şempanze Optimizasyon Algoritması (Chimp Optimization Algorithm)
CMSOS	: Kaotik Simbiyotik Organizmalar Arama (Chaotic Symbiotic Organisms Search)
CPU	: Merkezi İşlem Birimi (Central Processing Unit)
CRPSO	: Kimyasal Reaksiyon Kısmi Sürü Optimizasyonu (Chemical Reaction Partial Swarm Optimization)
CRSOA	: Bukalemun ve Remora Arama Optimizasyon Algoritması (Chameleon and Remora Search Optimization Algorithm)
CS	: Guguk Kuşu Araması (Cuckoo Search)
CSSA	: Kaotik Sincap Arama Algoritması (Chaotic Squirrel Search Algorithm)
DAPSO	: Dinamik Uyarlanabilir Parçacık Sürü Optimizasyonu (Dynamic Adaptive Particle Swarm Optimization)
DE-ANN	: Diferansiyel Evrim Yapay Sinir Ağı (Differential Evolution Artificial Neural Network)
DFA	: Yusufçuk Algoritması (Dragonfly Algorithm)
EMOBBO	: Verimli Çok Amaçlı Biyocoğrafya Tabanlı Optimizasyon (Efficient Mult Objective Biogeography-Based Optimization)
ESFO	: Geliştirilmiş Ayçiçeği Optimizasyon Algoritması (Enhanced Sunflower Optimization Algorithm)
FA	: Ateş Böceği algoritması (Firefly Algorithm)
FCFS	: İlk Gelen İlk Hizmet (First Come First Serve)
FN	: Yanlış Negatif (False Negative)
FP	: Yanlış Pozitif (False Positive)
GA	: Genetik Algoritma (Genetic Algorithm)
GA-ACO	: Genetik-Karınca Kolonisi Çizelgeleme Algoritması (Genetic-Ant Colony Scheduling Algorithm)
GALB	: Genetik Algoritma Tabanlı Yük Dengeleme (Genetic Algorithm-Based Load Balancing)
GBO	: Gradyan Tabanlı Optimizasyon (Gradient-Based Optimization)
GCP	: Google Bulut Platformu (Google Cloud Platform)

GPA	: Genelleştirilmiş Öncelik Algoritması (Generalized Priority Algorithm)
GPAWOA	: Güdümlü Nüfus Arşivi Balina Optimizasyon Algoritması (Guided Population Archive Whale Optimization Algorithm)
GPU	: Grafik İşlem Birimi (Graphics Processing Unit)
GSA	: Yerçekimi Arama Algoritması (Gravitational Search Algorithm)
GWO	: Gri Kurt Algoritması (Grey Wolf Algorithm)
HAPSO	: Hibrit Uyarlanabilir Parçacık Sürü Optimizasyonu (Hybrid Adaptive Particle Swarm Optimization)
HCTSO	: Hibrit Bulut Görev Çizelgeleme Optimizasyon Modeli (Hybrid Cloud Task Scheduling Optimization Model)
HD	: Yüksek Yoğunluk (High Density)
HESGA	: Genetik Algoritmayla Hibrit Elektro Arama (Hybrid Electro Search with a Genetic Algorithm)
HFSGA	: Genetik Algoritmayla Hibrit Flamingo Araması (Hybrid Flamingo Search with a Genetic Algorithm)
HGS	: Açlık Oyunu Arama Algoritması (Hunger Game Search algorithm)
HHG	: Hibrit Sezgisel ve Genetik (Hybrid Heuristic and Genetic)
HIGA	: Hibrit Metasezgisel Şema Uyumundan Esinlenen Genetik Algoritma (Hybrid Metaheuristic Scheme Harmony-Inspired Genetic Algorithm)
HPSOGA	: Hibrit Parçacık Sürü Optimizasyonu Genetik Algoritması (Hybrid Particle Swarm Optimization Genetic Algorithm)
HSA	: Harmony Arama Algoritması (Harmony Search Algorithm)
HSR	: Tarihsel Çizelgeleme Kaydı (Historical Scheduling Record)
IaaS	: Hizmet Olarak Altyapı (Infrastructure as a Service)
IDEA	: Geliştirilmiş Diferansiyel Evrim Algoritması (Improved Differential Evolution Algorithm)
InMaOEA	: Aralıklı Çok Amaçlı Evrimsel Algoritma (Interval Many-Objective Evolutionary Algorithm)
IoT	: Nesnelerin İnterneti (Internet of Thing)
IPSO	: Geliştirilmiş Parçacık Sürü Optimizasyon (Improved Particle Swarm Optimization)
IRMTS	: Yorumlanabilir Kural Tabanlı Metasezgisel Görev Çizelgeleme (Interpretable Rule-Based Metaheuristic Task Scheduling)
IT	: Bilgi Teknolojisi (Information Technology)
JSO	: Denizanası Arama Optimize Edici (Jellyfish Search Optimizer)
KnEA	: Diz Noktasına Dayalı Çok Amaçlı Evrimsel Algoritma (Knee Point-Driven Many-Objective Evolutionary Algorithm)
LD	: Düşük Yoğunluk (Low Density)
LOA	: Aslan Optimizasyon Algoritması (Lion Optimization Algorithm)
MCSOFLB	: Çok Amaçlı Kedi Sürü Optimizasyon Algoritması (Multi-Objective Cat Swarm Optimization Algorithm)
MCT	: Minimum Tamamlanma Süresi (Minimum Completion Time)
MCWOA	: Değiştirilmiş Şempanze-Balina Optimizasyon Algoritması (Modified Chimp-Whale Optimization Algorithm)
MD	: Orta Yoğunluk (Medium Density)
MIPS	: Saniyede Milyon Komut (Million Instructions Per Second)

MO-ACO	: Çok Amaçlı Karınca Kolonisi Optimizasyonu (Multi Objective-Ant Colony Optimization)
MoBLA	: Çok Amaçlı Arıların Yaşam Algoritması (Multi-objective Bees Life Algorithm)
MoDCSA	: Çok Amaçlı Ayrık Guguk Kuşu Arama Algoritması (Multi-Objective Discrete Cuckoo Search Algorithm)
MOEA	: Çok Amaçlı Evrimsel Algoritma (Multi Objective Evolutionary Algorithm)
MOELS	: Çok Amaçlı Evrimsel Yerel Arama (Multi-Objective Evolutionary Local Search)
MoGA	: Çok Amaçlı Genetik Algoritma (Multi-objective Genetic Algorithm)
MoGSA	: Çok Amaçlı Genetik Simüle Tavlama (Multi-objective Genetic- Simulated Annealing)
MoGWO	: Çok Amaçlı Gri Kurt Optimizasyonu (Multi Objective Grey Wolf Optimization)
MOO	: Çok Amaçlı Optimizasyon (Multi Objective Optimization)
MOPSO	: Çok Amaçlı Parçacık Sürü Optimizasyonu (Multi Objective Particle Swarm Optimization)
MoSA	: Çok Amaçlı Simüle Tavlama (Multi Objective Simulated Annealing)
MOTS	: Çok Amaçlı Tabu Arama (Multi Objective Tabu Search)
MPA	: Deniz Yırtıcı Algoritması (Marine Predator Algorithm)
MVO	: Çoklu Evren Optimize Edici (Multi-Verse Optimizer)
NSGA-2	: Domine Olmayan Sıralama Genetik Algoritması-2 (Non-dominated Sorting Genetic Algorithm-2)
NSGA-3	: Domine Olmayan Sıralama Genetik Algoritması-3 (Non-dominated Sorting Genetic Algorithm-3)
NSLS	: Domine Olmayan Sıralama Yerel Arama (Nondominated Sorting Local Search)
OBL	: Karşıtlık Temelli Öğrenme Algoritması (Oppositional-Based Learning Algorithm)
OSFO	: Muhalefet Tabanlı Ayçiçeği Optimizasyon Algoritması (Opposition based Sunflower Optimization Algorithm)
PaaS	: Hizmet Olarak Platform (Platform as a Service)
PFA	: Yol Bulucu Algoritması (Pathfinder Algorithm)
PSO	: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)
QEEC	: Q-Öğrenim Enerji Verimli Bulut Bilişim (Q-Learning Energy-Efficient Cloud Computing)
QoS	: Hizmet Kalitesi (Quality of Service)
RAM	: Rasgele Erişim Belleği (Random Access Memory)
RMPSO	: Rastgele Matris Parçacık Sürü Optimizasyonu (Random Matrix Particle Swarm Optimization)
RR	: Round Robin
RRM	: Yuvarlak Robin Modelleri (Round Robin Models)
RSA	: Sürüngen Arama Algoritması (Reptile Search Algorithm)
SaaS	: Hizmet Olarak Yazılım (Software as a Service)
SFO	: Ayçiçeği Optimizasyon Algoritması (Sunflower Optimization Algorithm)
SJF	: Önce En Kısa İş (Shortest Job First)
SOA	: Martı Optimizasyon Algoritması (Seagull Optimization Algorithm)
SOS	: Simbiyotik Organizmalar Arama (Symbiotic Organisms Search)
SPEA2	: Güç Pareto Evrimsel Algoritması 2 (Strength Pareto Evolutionary Algorithm 2)
SS	: Sincap Arama (Squirrel Search)
SSA	: Sincap Arama Algoritması(Squirrel Search Algorithm)

SSO	: Sinerjik Sürü Optimizasyonu (Synergistic Swarm Optimization)
TN	: Gerçek Negatif (True Negative)
TP	: Gerçek Pozitif (True Positive)
VM	: Sanal Makine (Virtual Machine)
VMM	: Sanal Makine Yöneticisi (Virtual Machine Manager)
WOA	: Balina Optimizasyon Algoritması (Whale Optimization algorithm)



1. GİRİŞ

Günümüz dijital dünyasının vazgeçilmezleri arasına giren bulut sistemler yapısı itibariyle karmaşıktır. Bulut sistemler, çok fazla sayıda bilgisayarın yüksek mühendislikle bir araya getirilmesi ile oluşur. Bu bilgisayarlar, kişisel bilgisayarlar olabileceği gibi ağ sunucuları da olabilmektedir. Bulut sistemler bünyesinde barındırdıkları kaynaklar ile kullanıcılarının farklı görev taleplerine cevap verebilirler. Bu sistemlerin donanım alt yapısı ve sunmuş oldukları hizmet göz önünde bulundurulduğunda yönetimi oldukça zor ve karmaşıktır. Bulut sistemlerin optimum bir şekilde yönetilmesinde enerji tüketiminin minimize edilmesi, servis sağlayıcı işletim giderlerinin minimizasyonu, bulut hizmetlerinden faydalanan kullanıcıların ödedikleri hizmet bedelinin minimizasyonu, hizmet kalitesinin artırılması vs. gibi amaçlar göz önünde bulundurulur.

Bulut sistemlerde görev çizelgeleme (task scheduling), bulut altyapısındaki kaynakların verimli bir şekilde kullanılmasını ve iş yüklerinin optimize edilmesini sağlayan kritik bir süreçtir. Görev çizelgeleme, bulut ortamındaki farklı kullanıcı taleplerini dikkate alarak kaynak yönetimi yaparak performansı artırır, maliyetleri düşürür ve hizmet kalitesi (Quality of Service, QoS) iyileştirir. Görev çizelgelemenin uygun bir şekilde yapılması durumunda kaynakların verimli kullanılması, maliyetlerin minimize edilmesi, hizmet kalitesinin artırılması ve performans artışının olması gibi önemli avantajları bulunmaktadır.

1.1. Tezin Amacı

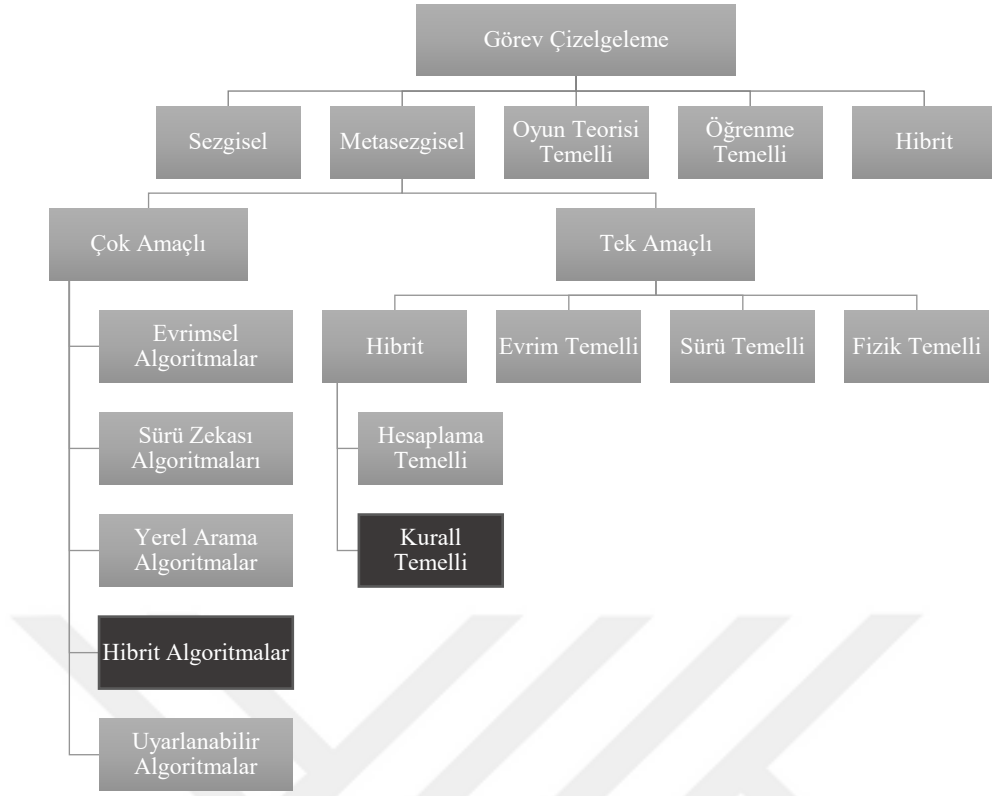
Bu tez çalışmasında, bulut sistemlerde görev çizelgeleme için önerilen tek ve çok amaçlı optimizasyon yöntemleri tanıtılmış ve tipik senaryolar dikkate alınarak performansları test edilmiştir. Tez çalışmasında önerilen ilk yöntem bulut sistemlerde görev çizelgeleme yapılırken görev gruplarının minimum sürede icra edilmesini amaçlayan kural tabanlı bir optimizasyon yöntemidir. Bu yöntem yarı çevrim dışı ve çevrim içi olmak üzere iki safhadan oluşmaktadır. Yarı çevrim dışı safhada sentetik bir şekilde oluşturulan farklı senaryolar üzerinde çeşitli kurallar oluşturulur. Çevrim içi safhada ise bu kurallardan faydalanarak bulutun mevcut durumu ve icra edilecek görevlerin sınıfını tespit edilir. Tespit edilen sınıf içerisindeki senaryolar ile mevcut senaryo arasındaki benzerlik araştırılır ve en yakın görev çizelgeleme modeli oluşturulur. Oluşturulan bu model referans alınarak görev çizelgeleme gerçekleştirilir.

Önerilen ikinci yöntem çok amaçlı hibrit bir yöntem olup; görevlerin icra süresi (makespan) ve görevlerin icrası için harcanan enerji miktarını optimize etmektedir. Önerilen yöntem literatürde Domine Olmayan Sıralama Genetik Algoritması-2 (Non-dominated Sorting Genetic Algorithm-2, NSGA-2) ve Güç Pareto Evrimsel Algoritması 2 (Strength Pareto Evolutionary Algorithm2, SPEA2) olarak bilinen algoritmalar paralel bir şekilde çalıştırır ve bu algoritmaların bulmuş olduğu en iyi çözümleri bir hibrit havuzunda tutar. İterasyonlar boyunca iki algoritmanın bulmuş olduğu en iyi

sonular ile hibrit havuzunda bulunan en iyi sonular kıyaslanır. Ana sre NSGA-2 ve SPEA2 algoritmalarında yeni srelerin oluřturulup oluřturulmayacaėını kontrol eder. Ana sre hibrit havuzunda bulunan sonuları iyileřtiren algoritma ile yeni sreler oluřturur ve uygulamaya dahil eder. Hibrit havuza katkı saėlamayan algoritmadan yeni sre oluřturulması ana sre tarafından durdurulur. Hibrit havuz boyutunun ok fazla bymemesi iin belirli bir sınırlandırma getirilmiřtir. İterasyonlar sonunda hibrit havuzda bulunan zmler elde edilen en iyi zmler olarak sunulur.

1.2. Literatr Arařtırması

Bulut biliřim, insan etkileřimi olmadan, gerek zamanlı bilgi iřlem yeteneklerini tek taraflı olarak saėlayabilir ve ynetebilir. Bunu yaparken kullandıėı en nemli mekanizmalardan birisi de grev izelgelemedir. Bulut sistemlerde grev izelgeleme, belirli bir stratejiye gre kullanıcı grevlerinin en uygun sanal makinelere atanması srecidir. Kullanılacak grev izelgeleme stratejisi, bulut sistemin dinamik ve heterojen yapısını dikkate almak zorundadır. Bununla birlikte kullanılacak grev izelgeleme stratejisinin hangi amaca ynelik yapıldıėı da nemlidir. Bulut sistemlerde grev izelgeleme, makespan, enerji tketimi, parasal maliyet gibi amalardan birini (tek amalı) veya bunlardan aynı anda birden fazlasını (ok amalı) dikkate alacak řekilde geliřtirilebilir. Bununla birlikte spesifik kullanıcı kısıtlarını dikkate alan hizmet kalitesi (Quality of Service, QoS) temelli grev izelgeleme de bulut sistemlerde nemli bir yere sahiptir. Kullanıcı grevleri sanal makinalara belirli bir sreliėine atanabilir veya grev yrtm iin diėer grevlerin sanal makinalara tamamlanması beklenebilir. Genel olarak bulut sistemlerde kullanılan grev izelgeleme yntemleri ve bu tez alıřmasında nerilen yaklařımların sınıfı řekil 1.1'deki gibi gsterilebilir.



Şekil 1.1. Bulut sistemlerde kullanılan görev çizelgeleme türleri

1.2.1. Sezgisel Yöntemler

Sezgisel algoritmalar, statik ve çok karmaşık olmayan sistemler için uygundur. Min-Min [1], minimum tamamlanma süresi (Minimum Completion Time, MCT) [2] gibi iyi bilinen sezgisel algoritmalar, beklenen yürütme süresi değerini dikkate alırlar. Bu algoritmalarda temel prensip, görevleri en kısa sürede tamamlayan sanal makinelerle atamaktır. Ancak bu tip sezgisel algoritmalarda büyük boyutlu görevlerin tamamlanması gecikir veya daha yavaş olan sanal makineler boşa kalabilir. Round-Robin (RR) [3] gibi zaman bölmesi kullanan önleyici sezgisel algoritmalar her bir görevin kaynaklara ulaşmasına olanak sağlar. Öte yandan bu algoritmalarındaki zaman bölmesi süreleri, QoS gereksinimleri ve performans açısından önemlidir. Kısa zaman bölmesi, düşük işlemci kapasitesi (Central Processing Unit, CPU) verimine sebep olurken yüksek zaman bölme ise uzun cevap sürelerine neden olabilir. Sezgisel önce en kısa görev (Shortest Job First, SJF) [4] daha iyi cevap sürelerine sahiptir ancak küçük görevlerin sayısının fazla olması ciddi performans kaybına yol açabilir.

Pachipala ve ekibi [5] bulut sistemlerde görev çizelgeleme problemi için SJF algoritmasının gelişmiş halini önermiştir. Bu çalışma görev çizelgelemede uyarlanabilirlik ve performansı arttırmak için bazı yenilikler getirmiştir. Çalışma görevlerin özelliklerini ve bulut sistemin mevcut kaynak durumunu dikkate alarak görevlerin icra süresi ve kaynak kullanımını optimum olarak gerçekleştirmeyi hedeflemektedir. Deneysel sonuçlar çalışmanın klasik SJF, RR, İlk Gelen İlk Alır (First Come First

Serve, FCFS) yöntemlerine göre gerek kaynak kullanımı yönünden gerekse görevlerin icra süresi yönünden daha iyi olduğunu göstermiştir.

Agarwal ve Jain [6] bulut sistemlerde görev çizelgeleme için geliştirilmiş öncelik algoritması (Generalized Priority Algorithm, GPA) önermiştir. CloudSim aracı üzerinde gerçekleştirilen simülasyonlarda bu çalışmada önerilen yaklaşım FCFS ve RR ile kıyaslanmıştır. Elde edilen sonuçlarda bu yaklaşımın bulut sistemlerde görev çizelgelemede FCFS ve RR'den daha iyi performans gösterdiği belirtilmiştir.

1.2.2. Oyun Teorisine Dayalı Yöntemler

Oyun teorisine dayalı yöntemler [7,8], görev çizelgelemeyi birden fazla varlık (kullanıcılar, görevler veya kaynaklar) arasında bir oyun olarak modeller ve görev atamalarını optimize etmek için oyun teorisindeki kavramları kullanır. Bu yöntemler, çizelgeleme sürecinde kaynak tahsis stratejileri, teşvikleri ve adalet gibi faktörleri dikkate alır. Bununla birlikte oyun teorisine dayalı yöntemlerde karmaşıklık, bilginin mevcut olması ve stratejik davranış gibi bazı zorlukları da dikkate almak gereklidir. Bu yöntemler, derin uzmanlık gerektiren karmaşık matematiksel modeller içerebilir. Bununla birlikte başarımlar için oyuncuların güncel bilgileri önem arz eder ki; dinamik bulut sistemlerde bunu sağlamak güçtür. Bunun en önemli nedeni ise oyuncu davranışlarının tahmin etmenin zor olmasıdır.

Literatürde bulut sistemlerde oyun teorisine dayalı yöntemlere örnek olarak Yang ve ekibi [9] yapmış olduğu çalışma gösterilebilir. Bu çalışmada bulut sistemlerde enerji tüketiminin minimize edilmesi için oyun teorisine dayanan bir görev çizelgeleme algoritması geliştirilmiştir. Geliştirilen algoritmada kullanıcılardan gelen tüm görevler bir oyuncu; sanal makineler ise oyun stratejisi olarak seçilmektedir. Görev yürütüm süresi ise amaç fonksiyonu olarak seçilmiştir. Deneysel sonuçlar bu çalışmanın karınca kolonisi algoritması (Ant Colony Optimization, ACO) ve genetik algoritmalar (Genetic Algorithm, GA) göre daha iyi sonuçlar verdiğini göstermiştir.

Attiya ve ekibi [10] nesnelerin interneti (Internet of Things, IoT) sistemlerinde bulut ve sis ortamlarında görevlerin etkin bir şekilde planlanması için HGSMMPA yöntemini önermiştir. Bu yaklaşım, temelini açlık oyunu arama algoritmasından (Hunger Game Search algorithm, HGS) almaktadır. HGS algoritmasının güçlü yönlerini koruyarak, deniz yırtıcı algoritması (Marine Predator Algorithm, MPA) ile birleştirilmesi sayesinde daha da gelişmiş bir görev çizelgeleme performansı elde edilmektedir. Deneysel sonuçlar, HGSMMPA'nın enerji tüketimi ve makespan süresi açısından kıyaslanan diğer yöntemlere göre önemli avantajlar sağladığını göstermiştir. Elde edilen sonuçlara göre, HGSMMPA, hem sentetik hem de gerçek senaryolarda sırasıyla %19.31 ve %17.47 oranında daha kısa makespan değeri elde ederken, enerji tüketimini ise %14.72 ve %17.68 oranında azaltmaktadır.

1.2.3. Öğrenme Tabanlı Yöntemler

Öğrenme tabanlı yöntemler [11,12], görev çizelgeleme planları oluşturmak için veriye dayalı makine öğrenime tekniklerinden faydalanır. Kaynak taleplerini, görev performansını veya optimum çizelgeleme kararlarını tahmin etmek için takviyeli öğrenme, derin öğrenme veya denetimli öğrenme gibi teknikleri kullanırlar. Bu yöntemlerin başarımları yüksek olabilmekle birlikte bazı stratejik zayıflıkları da vardır. Bunların başında, bu yöntemlerin veri bağımsız olması ve kullanılan modellerin genelleştirmesinin zor olması gelir. Ayrıca derin sinir ağları gibi bazı modellerin karmaşık olması yorumlanabilirliği de azaltmaktadır.

Ding ve arkadaşları [13], bulut sistemlerde enerji verimliliğini artırmayı hedefleyen Q-Öğrenim enerji verimli bulut bilişim (Q-Learning Energy-Efficient Cloud Computing, QEEC) yöntemini önermiştir. QEEC iki aşamalı bir yapıdan oluşmuştur. İlk aşamada, merkezi bir görev yönlendirici kullanarak gelen kullanıcı taleplerinin her bir sunucuya dağıtılmasını sağlayan M/M/S kuyruklama modeli uygulanır. İkinci aşamada, her sunucuda yer alan Q-öğrenme tabanlı bir zamanlayıcı, önce istekleri görev gevşekliliği ve yaşam süresine göre sıralar, ardından görevleri sanal makinelere atamak için sürekli güncellenen bir politika kullanır. Bu politika, görev yanıt süresini minimize eden ve sunucunun CPU kullanımını maksimize eden atamaları ödüllendirir. Deneysel sonuçlar, M/M/S kuyruklama modelinin bulut ortamında ortalama görev yürütüm süresini azalttığını doğrulamıştır. Ayrıca, QEEC'nin diğer görev planlama yaklaşımlarına kıyasla en enerji verimli yöntem olduğunu ve bu avantajın büyük ölçüde M/M/S kuyruklama modeli ve Q-öğrenme stratejisinden kaynaklandığını göstermiştir.

1.2.4. Hibrit Yöntemler

Hibrit yöntemler farklı yöntemlerin güçlü yönlerini birleştirerek daha etkili ve verimli çözümler üretmeyi amaçlar. Metasezgisel algoritmaların kullandığı iç mekanizma ve rastgelelik barındıran süreçleri bir birinden farklılık gösterebilir. Bu algoritmaların beraber kullanılması, özellikle karmaşık problemlerde avantajlar sağlayabilmektedir. Chen ve Long [14] Parçacık Sürü Optimizasyonu (Particle Swarm Optimization, PSO) ve ACO algoritmalarının hibrit kullanılması ile bulut sistemler görev çizelgeleme performansının arttırılabileceğini göstermişlerdir. Tsai ve ekibi [15] buluttaki makespan değerini azaltmak için GA, ACO ve PSO'yu tek bir çerçeveye entegre ederek hiper sezgisel çizelgeleme algoritmasını geliştirmiştir.

Shukri ve ekibi [16], Çoklu Evren Optimize Edici (Multi-Verse Optimizer, MVO) geliştirilmiş bir versiyonu olan EMVO algoritmasını bulut sistemlerde görev çizelgelemesi problemi için önermiştir. Çalışmada önerilen EMVO algoritması, hem orijinal MVO hem de PSO algoritmaları ile karşılaştırılmıştır. Simülasyon sonuçları, EMVO algoritmasının görevlerin makespan süresini önemli ölçüde azaltırken, aynı zamanda sistem kaynaklarının kullanımını üst düzeye çıkardığını göstermiştir. Böylece hem iş yükü yönetimi hem de maliyet etkinliği açısından önemli avantajlar sağlanmaktadır.

Agarwal ve arkadaşları [17] tarafından işlemciler üzerinde daha iyi görev çizelgelemesi sağlamak amacıyla hibrit genetik algoritma ve enerji bilinçli çizelgeleme adını taşıyan yeni bir yöntem önermiştir. Sadece GA ile görev çizelgelemesi yapıldığında hesaplama maliyeti artar ve bu durum enerji tüketimini de arttırır. Hibrit genetik algoritma ile bu zorluklar aşılarak minimum makespan ve enerji tüketimi değerlerine ulaşılabilir. Öncelik yaklaşımları kullanılarak üç ana kromozom oluşturmak için GA kullanılır. Tahsis edilen kaynaklar enerji bilinçli planlama modeli aracılığıyla optimize edilir. Bu çalışmada geliştirilen yaklaşımın PSO, GA, Yerçekii Arama Algoritması (Gravitational Search Algorithm, GSA), ACO ve Round Robin Modelleri (Round Robin Models, RRM) gibi yöntemlerle karşılaştırıldığında daha iyi sonuç verdiği gözlemlenmiştir.

Gupta ve ekibi [18] metasezgisel tekniklerle sinir ağı tabanlı bir hibrit model önermiştir. Sunulan sinir ağından esinlenen Diferansiyel Evrim Yapay Sinir Ağı (Differential Evolution Artificial Neural Network, DE-ANN) hibrit tekniği, bulut sistemlerde görevlerin optimum şekilde atanmasını sağlamaktadır. DE-ANN hibrit yaklaşımının performansı, ortalama başlangıç süresi, ortalama bitiş süresi, ortalama yürütme süresi, toplam tamamlanma süresi, simülasyon süresi ve ortalama kaynak kullanımı gibi performans ölçütlerine göre değerlendirilmiştir. DE-ANN yaklaşımı, deneysel sonuçlarda önerilen yöntem Büyük Patlama Büyük Çatırtı (Big-Bang Big-Crunch, BB-BC) ve GA karşılaştırılarak test edilmiştir ve mevcut metasezgisel tekniklerden daha iyi performans göstermiştir. DE-ANN yaklaşımı, toplam tamamlanma süresinde ortalama %19.15, ortalama bitiş süresinde %32.23, ortalama yürütme süresinde %51.95 ve ortalama kaynak kullanımında %33.24 oranında iyileştirme sağlamıştır. Ayrıca, DE-ANN tekniği, dinamik ve belirsiz ortamlarda da etkili bir şekilde çalışarak, gerçek dünya bulut altyapıları için uygun bir çözüm sunmuştur.

Krishnadoss ve ekibi [19], bulut bilişim ortamında çizelgelemenin etkin bir şekilde yapılması amacıyla Guguk Kuşu Araması (Cuckoo Search, CS) ve Martı Optimizasyon Algoritması (Seagull Optimization Algorithm, SOA) özelliklerini birleştiren yenilikçi bir optimizasyon algoritması önermiştir. Bu yöntem, heterojen bulut ortamında görev çizelgelemesi sırasında maliyet ve zaman parametrelerini en aza indirmeyi hedeflemektedir. Algoritmanın performansı, CloudSim aracı ile Çok Amaçlı Karınca Kolonisi Optimizasyonu (Multi Objective-Ant Colony Optimization, MO-ACO), ACO ve Min-Min algoritmaları karşılaştırılmıştır. Sonuçlar, SOA-CS tekniğinin, 300 sanal makine göz önünde bulundurulduğunda makespan süresinde sırasıyla %1.06, %4.2 ve %2.4 iyileştirme sağladığını ve genel maliyetin PSO, ACO ve Geliştirilmiş Diferansiyel Evrim Algoritması (Improved Differential Evolution Algorithm, IDEA) algoritmaları ile karşılaştırıldığında sırasıyla %1.74, %3.93 ve %2.77% oranında azaldığını göstermiştir.

Talha ve ekibi [20] Yol Bulucu Algoritması (Pathfinder Algorithm, PFA) ve Karşıtlık Temelli Öğrenme Algoritması (Oppositional-Based Learning Algorithm, OBL) algoritmasını kullanan yeni bir yöntem geliştirmiştir. Bu yöntem, toplam yürütme süresi, maliyet ve kaynak kullanımını optimize etmeyi amaçlamıştır. Deney sonuçları, OBLPFA algoritmasının toplam yürütme süresi, maliyet ve kaynak kullanımı açısından, PSO, yusufçuk Algoritması (Dragonfly Algorithm, DFA), AOA Aritmetik

Optimizasyon Tekniđi (Arithmetic Optimization Technique, AOA), Sürüngen Arama Algoritması (Reptile Search Algorithm, RSA), Aquila Optimizasyonu (Aquila Optimization, AO) ve Aslan Optimizasyon Algoritması (Lion Optimization Algorithm, LOA) gibi diđer algoritmalara kıyasla daha iyi performans gösterdiğini kanıtlamıştır.

Hussain ve Begh [21], görev planlama probleminde maliyeti en aza indirmek amacıyla yeni bir hibrit metasezgisel algoritma olan Genetik Algoritmayla Hibrit Flamingo Araması (Hybrid Flamingo Search with a Genetic Algorithm, HFSGA) önermiştir. HFSGA, flamingo aramasının küresel arama yeteneđi ile genetik algoritmanın yerel arama yeteneđini birleştiren etkili bir çözüm sunmuştur. Algoritmanın performansını deđerlendirmek için yedi farklı benchmark fonksiyonu üzerinde deneyler yapılmıştır. Friedman sıralama testi ile elde edilen sonuçlar, HFSGA'nın ACO, PSO, GA ve RR gibi yaygın olarak kullanılan algoritmalara göre, görevlerin tamamlanma süresi, son icra tarihi karşılama oranı ve toplam maliyet açısından daha iyi sonuçlar verdiğini göstermiştir.

Mansouria ve arkadaşları [22], bulut bilişim ortamlarında görevlerin etkin bir şekilde çizelgelenmesi için bulanık sistem ve deđiştirilmiş PSO tekniklerini birleştiren yeni bir hibrit algoritma olan FMPSO önermiştir. FMPSO, PSO algoritmasının küresel arama yeteneđini artırmak için dört farklı hız güncelleme yöntemi ve rulet tekerleđi seçimi mekanizması kullanmaktadır. Ayrıca, yerel optimum noktalarından kaçınmak için çaprazlama ve mutasyon operatörleri ile desteklenmektedir. Bulanık mantık sistemi ise, görevlerin özelliklerine (uzunluk, CPU hızı, rasgele erişim belleđi (Random Access Memory, RAM) vb.) göre daha esnek ve akıllı kararlar verebilmek için sisteme dâhil edilmiştir. Böylece, FMPSO hem görevlerin yürütme sürelerini minimize etmiş hem de kaynakların daha etkin kullanımını sağlamıştır. CloudSim aracı ile gerçekleştirilen simülasyonlarda, FMPSO algoritmasının, diđer yöntemlere göre daha iyi bir performans sergilediđini ve özellikle büyük ölçekli problemlerde avantaj sağladığını göstermiştir.

Velliangiri ve ekibi [23], görev çizelgeleme sürecini makespan süresi, yük dengeleme, kaynak kullanımı ve bulut maliyeti gibi parametreleri dikkate alarak optimize etmek amacıyla, Genetik Algoritmayla Hibrit Elektro Arama (Hybrid Electro Search with a Genetic Algorithm, HESGA) önermiştir. Bu yöntem, genetik algoritmanın ve elektro arama algoritmasının avantajlarını bir araya getirir. GA en iyi yerel optimum çözümleri sunarken, elektro arama algoritması en iyi küresel optimum çözümleri sağlar. Bu algoritma, Hibrit Parçacık Sürü Optimizasyonu Genetik Algoritması (Hybrid Particle Swarm Optimization Genetic Algorithm, HPSOGA), GA, ES ve ACO gibi mevcut çizelgeleme algoritmalarına kıyasla daha üstün bir performans sergilemiştir.

Abualigah ve arkadaşları [24] bulut sistemlerde görev çizelgelemesi için uyarlanmış gelişmiş bir optimizasyon algoritması önermiştir. Bu yöntem, Jaya algoritması ile Sinerjik Sürü Optimizasyonu (Synergistic Swarm Optimization, SSO) birleştirerek, keşif-sömürü dengelerini geliştirmek ve yakınsama hızını artırmak amacıyla bir Levy uçuş mekanizması entegre etmektedir. Jaya algoritmasının mevcut en iyi çözümleri kullanma yeteneđi, SSO'nun işbirlikçi arama stratejisi ile desteklenmiştir, bu da sinerjik bir optimizasyon çerçevesi oluşturur. Levy uçuşlarının eklenmesi, arama sürecine

stokastiklik katarak algoritmanın yerel optimuma takılma riskini azaltır ve karmaşık çözüm alanlarında daha etkili gezinmesine olanak tanır. Bu algoritma, bulut ortamlarında görev çizelgeleme problemleri üzerinde test edilmiş ve en son yaklaşımlarla kıyaslanmıştır. Deneysel sonuçlar, çözüm kalitesi, yakınsama hızı ve ölçeklenebilirlik açısından algoritmanın üstünlüğünü ortaya koymuştur. Genel olarak, bu yaklaşım, bulut sistemlerde gelişmiş kaynak kullanımı ve sistem performansını iyileştirmektedir. Yaklaşım, orijinal yöntemle kıyasla %10 iyileştirme ile %88 doğruluk değeri elde etmiştir.

Sathiskumar ve arkadaşları [25] bulut sistemlerde işbirlikçi öğrenmeye uygun, yeterli planlama ve performans iyileştirmesi sağlamak amacıyla benzersiz bir Sincap Arama (Squirrel Search, SS) tabanlı AlexNet Çizelgeleyici (AlexNet Scheduler) algoritmasını (SSbANS) önermiştir. Sistem, bulut kullanıcılarının talep ettiği görevleri işleyerek, başlangıçta her görevin önceliğini kontrol eder ve düzenler. Ayrıca, veri hızı ve iş planı dikkate alınarak, SS algoritmasının uygunluk fonksiyonu kullanılarak optimum kaynaklar seçilir. Çizelgeleme süreci boyunca sistem, aşırı yüklenmiş kaynakları sürekli olarak kontrol eder ve sincap dağıtım fonksiyonuna göre kaynaklar arasında denge sağlar. Modelin etkinliği, yanıt süresi, kaynak kullanımı, üretim süresi ve verimlilik gibi kriterler üzerinden değerlendirilmiştir. Deneysel sonuçlar, SSbANS modelinin daha düşük yanıt ve makespan süresiyle daha yüksek çıktı ve kaynak kullanım oranı sağladığını göstermiştir.

Hamed ve ekibi [26], bulut ortamlarında görevlerin etkin bir şekilde çizelgelemesi için yeni bir yöntem önermiştir. Bu yöntem, işbirliği arama algoritmasının gücünü kullanarak heterojen bulut kaynakları üzerindeki görevlerin dağılımını optimize etmeyi hedeflemektedir. Metasezgisel algoritmaların avantajlarından yararlanan bu yaklaşım, farklı görev sayılarına sahip çeşitli senaryolarda değerlendirilmiştir. Deneylerde, bu yaklaşım makespan metriğine göre yeni GA, GA, Balina Optimizasyon Algoritması (Whale Optimization Algorithm, WOA) GSA ve Hibrit Sezgisel ve Genetik (Hybrid Heuristic and Genetic, WOA) gibi mevcut yöntemlere kıyasla sırasıyla %7.9, %2.1, %8.8, %7.7 ve %3.4 oranında daha iyi sonuçlar vermiştir.

Gupta ve ekibi [27], bulut kaynaklarının en iyi şekilde kullanımı için hata farkında bir sistem oluşturmak amacıyla Harmony Arama Algoritması (Harmony Search Algorithm, HSA) ve ANN Yapay Sinir Ağı (Artificial Neural Network, ANN) önermiştir. Bulut sistemler gibi hatalı ortamlar için mümkün olan en verimli ve zaman tasarruflu sistemin geliştirilmesi amaçlanmıştır. Deneysel sonuçlar, HSA'dan esinlenen ANN modelinin, Red Fox ve Crow aramasından esinlenen modele kıyasla daha düşük yürütme süresi, daha az görev başarısızlığı, daha düşük güç tüketimi ve yüksek kaynak kullanımı sağladığını göstermektedir.

Liu ve Wu [28], bulut sistemlerde görev çizelgelemesinin özelliklerine odaklanarak, Genetik-Karınca Kolonisi Çizelgeleme Algoritması'na (Genetic-Ant Colony Scheduling Algorithm, GA-ACO) dayalı bir görev çizelgeleme algoritması önermiştir. Algoritma, ACO'nun güçlü pozitif geri bildirim avantajını, özellikle yakınsama oranı üzerinde dikkate almıştır. Bu algoritma, optimum çözümü hızla elde etmek için genetik algoritmanın küresel arama yeteneğini kullanır ve bu çözümü ACO'nun başlangıç feromonuna dönüştürür. Simülasyon sonuçları, bu algoritmanın GA ve ACO ile

karşılaştırıldığında, özellikle büyük ölçekli ortamlarda, daha yüksek verimlilik sağladığını ve buda algoritmanın bulut sistemlerde etkili bir görev çizelgeleme yöntemi olduğunu göstermektedir.

1.2.5. Metasezgisel Yöntemler

Pek çok NP-Zor (NP-Hard) problemin çözümünde etkin olarak kullanılan metasezgisel yaklaşımlar bulut sistemlere de başarılı bir şekilde uyarlanmıştır. Literatürde farklı mekanizmalar kullanan metasezgisel algoritmalar, uygun temsil biçimlerinin geliştirilmesi durumunda görev çizelgeleme problemlerine uyarlanabilmektedir. Literatürde çalışılan metasezgisel yöntemler farklı sayıda amacı optimize etmeyi hedeflemiştir. Bu bölümde tek amaçlı ve çok amaçlı metasezgisel yöntemler ile ilgili literatür çalışmalarına yer verilmiştir.

Tek Amaçlı Metasezgisel Yöntemler

Tek amaçlı optimizasyon, belirli bir amaç fonksiyonunu en iyi hale getirmeyi hedefleyen bir optimizasyon türüdür. Literatür çalışmaları incelendiğinde tek amaçlı birçok metasezgisel optimizasyon yöntemi geliştirilmiştir. Bu yöntemler 4 farklı sınıfa ayrılmaktadır.

1. Fizik temelli Yöntemler

Bu yöntemler, doğada gözlemlenen fiziksel olayları taklit ederek çözümler üretir ve büyük, karmaşık problem alanlarında iyi sonuçlar verebilirler. Fizik temelli yöntemler geniş bir arama uzayından yerel minimumlardan kurtulma ve optimum çözümlere ulaşabilme yeteneğine sahiptirler. Bulut sistemler gibi çok sayıda ve çeşitte görevin bulunduğu karmaşık sistemlere kolayca adapte olabilmeleri ve hızlı çözüm üretmeleri önemli avantajlar sağlamaktadır.

Huang ve ekibi [29] önermiş olduğu çalışma fizik temelli bir yöntem olup; bu çalışmada yerel optimumlara erken düşmeyi engelleyebilecek ve daha hızlı yakınsamaya sahip olan Gradyan Tabanlı Optimizasyon (Gradient-Based Optimization, GBO) yönteminden faydalanarak makespan değerinin optimize edilmesi amaçlanmıştır. GBO sürekli optimizasyon için önerilen bir yöntem olup; GBO'nun vektör değerini en yakın tam sayıya dönüştürmek için yuvarlama fonksiyonu kullanılmıştır. Elde edilen bu değer görev çizelgeleme için bir çözümü temsil etmektedir. Deneysel sonuçlar bu yöntemin özellikle büyük ölçekte görevlerin varlığında optimum çözümü aramada daha iyi yakınsama hızına ve doğruluk değerine sahip olduğunu göstermiştir.

2. Sürü Temelli Yöntemler

Sürü temelli optimizasyon yöntemleri doğadan gözlemlenen etkileşimleri ve davranışları taklit eden ve bunları optimizasyon süreçlerine dahil eden yöntemlerdir. Bu yöntemler dağıtık, ölçeklenebilir ve güçlü oldukları için karmaşık problemler için uygulanabilir. Literatürde sürü temelli metasezgisel yöntemlerde algoritmalar PSO [30], Gri Kurt Algoritması (Grey Wolf Algorithm, GWO) [31], Ateş Böceği Algoritması (Firefly Algorithm, FA) [32], ACO [33], WOA [34] ve Denizanası Arama Optimize

Edici (Jellyfish Search Optimizer, JSO) [35] başarılı bir şekilde bulut görev çizelgeleme problemlerine uyarlanmıştır. Bu algoritmalarda olası çözümler popülasyon olarak adlandırılır. Her bir iterasyonda, popülasyonun en başarılı çözümü veya çözümleri diğer adayların çözüm önerilerini de etkiler.

Chandrashekar ve arkadaşları [36] bulut sistemlerde görev çizelgeleme için Şempanze Optimizasyon Algoritması (Chimp Optimization Algorithm, COA) ve WOA algoritmalarını birleştiren Değiştirilmiş Şempanze-Balina Optimizasyon Algoritması (Modified Chimp-Whale Optimization Algorithm, MCWOA) yöntemini önermiştir. MCWOA'nın tanımlama hassasiyetini artırmak amacıyla popülasyon başlatma aşamasında sobol dizisi kullanılmıştır. Böylece popülasyonun çözüm alanı boyunca eşit bir şekilde dağılması sağlanmıştır. Ayrıca, geleneksel MCWOA'nın yerel arama yetenekleri, balina optimizasyon algoritmasının kabarcık ağı avcılığı ve rastgele arama mekanizmalarının MCWOA'nın konum güncelleme sürecine dahil edilmesiyle geliştirilmiştir. Simülasyonlarda, özellikle çoklu hasar tespit senaryolarında yeni yöntemin orijinal MCWOA'dan daha iyi performans göstermiştir. MCWOA, yanlış pozitiflerden kaçınmada ve hesaplama hızını artırmada üstün olmuştur. MCWOA'nın verimliliği, enerji kullanımı, hesaplama maliyeti, görev süresi ve gecikme gibi kriterler çerçevesinde değerlendirilmiştir. Simülasyonlarda, yeni MCWOA'nın tüm kriterlerde WOA, Şempanze Algoritması (Chimp Algorithm, CA), Karınca Aslanı Optimize Edici (Ant Lion Optimizer, ALO), GA ve GWO algoritmalarına göre daha iyi sonuçlar vermiştir.

Sanaj ve Prathap [37], Hizmet Olarak Altyapı (Infrastructure as a Service-IaaS) bulut ortamında çoklu görev planlamasını optimize etmek için Kaotik Sincap Arama Algoritmasını (Chaotic Squirrel Search Algorithm, CSSA) önermiştir. Bu yöntem, mevcut yaklaşımları daha uygun maliyetli hale getiren sürekli iş planları üretir. Daha fazla küresel yakınsama sağlamak amacıyla, erken aşamada, verimli bir ekosistem oluşturmak için kaotik optimizasyonla geliştirilmiştir. Önerilen CSSA, Sincap Arama Algoritması (Squirrel Search Algorithm, SSA) keşif yeteneklerini artırmak için kaotik yerel arama ile birleştirilmiştir. Deneysel sonuçlar bu yöntemin zaman, enerji, maliyet ve kaynak yönetimi yönünden karşılaştırılan yöntemlere göre daha iyi performans gösterdiğini göstermiştir.

Panda ve arkadaşları [38], Macar algoritması olarak bilinen optimizasyon yöntemine dayanan çift tabanlı bir görev planlama algoritması önermiştir. Bu algoritma, bulut sistemlerde eşit olmayan sayıda görev ve bulut kaynaklarını dikkate alarak, görevleri en uygun şekilde eşleştirmeyi hedefler. Algoritma farklı veri kümeleri ile simüle edilmiş ve üç farklı yöntemle kıyaslanmıştır. Elde edilen sonuçlar algoritmanın, mevcut algoritmalarla kıyasla daha kısa bekleme süresi sağladığını göstermektedir.

Daqin [39] bulut sistemlerde görev çizelgeleme için Geliştirilmiş Parçacık Sürü Optimizasyonu (Improved Particle Swarm Optimization, IPSO) algoritmasını önermiştir. Bu yöntem yerel bir optimuma düşmekten mümkün olduğunca kaçınmakta ve yakınsama hızı oldukça iyidir. CloudSim simülasyon aracıyla gerçekleştirilen simülasyonlarda, algoritmanın sonuçları iyileştirme ve hızlı sonuç alma noktasında avantajlara sahip olduğunu göstermiştir. Zhan ve Hou [40], bulut sistemlerde kaynak planlama problemi için PSO'nun gelişmiş bir türünü önermiştir. Deneysel sonuçlar bu yöntemin görev

ortalama çalışma süresini azaltabileceğini ve kaynakların kullanılabilirlik oranını artırabileceğini göstermiştir.

Al-Maamari ve Omara [41], belirli bir görev kümesinin makespan süresini en aza indirirken kaynak kullanımını en üst düzeye çıkarmak amacıyla temel PSO algoritmasının performansını artırmak için Dinamik Uyarlanabilir Parçacık Sürü Optimizasyonu (Dynamic Adaptive Particle Swarm Optimization, DAPSO) algoritmasını önermiştir. Ayrıca, bulut sistemler üzerinde bağımsız görevleri çözelgelemek için yeni bir görev çözelgeleme algoritması önerilmiştir. Bu algoritma, PSO algoritması ile CS algoritmasının bir birleşimi olup MDAPSO olarak adlandırılmıştır. Deneysel sonuçlar, MDAPSO ve DAPSO algoritmalarının orijinal PSO algoritmasından daha iyi performans gösterdiğini ortaya koymuştur.

Chandrashekar ve Krishnadoss [42], maliyet, enerji ve tamamlanma süresi açısından hâlihazırda mevcut görev çözelgeleyicilerin verimini artırmak için bir Muhalefet Tabanlı Ayçiçeği Optimizasyon Algoritması (Opposition based Sunflower Optimization Algorithm, OSFO) önermiştir. Bu yöntem standart Ayçiçeği Optimizasyon Algoritması (Sunflower Optimization Algorithm, SFO), ACO, MIN-MIN ve FCFS yöntemleri ile kıyaslanmıştır. Deneysel sonuçlar bu yöntemin makespan süresi, maliyet ve enerji tüketimi açısından diğer yöntemlere göre daha iyi sonuçlar verdiğini göstermiştir.

Tang ve ekibi [43] maliyet kısıtları altında sanal makinelerde (Virtual Machine, VM) bulut hizmetlerinin zamanlanması problemini ele almışlardır. Bu problemi çözmek için, Rastgele Matris Parçacık Sürü Optimizasyonu (Random Matrix Particle Swarm Optimization, RMPSO) tabanlı yeni bir algoritma önerilmiştir. RMPSO, görevlerin sanal makinelere atanmasını, bir rastgele tam sayı matrisi ile temsil ederek ve PSO'nun gücünü kullanır. Ancak, büyük ölçekli sistemlerde RMPSO'nun zaman karmaşıklığı yüksek olabilir. Bu sorunu aşmak için, paralel hesaplama teknikleri kullanılarak iki farklı varyant önerilmiştir: çok çekirdekli CPU'larda çalışan M-RMPSO ve yüksek performanslı grafik İşlem Birim'lerde (Graphics Processing Unit, GPU) çalışan G-RMPSO. Kapsamlı deneyler sonucunda, G-RMPSO algoritmasının, bulut hizmetlerinin toplam maliyetini minimize ederken, aynı zamanda yürütme süresini de önemli ölçüde azalttığı görülmüştür.

Liu [44], ACO'ya dayalı bir görev çözelgeleme yöntemi önermiştir. Polimorfik ACO temelinde, algoritmanın yakınsama hızını artırmak ve yerel en iyi çözümlerin ortaya çıkmasını etkili bir şekilde önlemek için bir feromon güncelleme mekanizması eklenmiştir. Yöntem, kullanıcılar tarafından gönderilen görevleri daha kısa yürütme süresi, daha düşük maliyet ve dengeli yük oranı ile planlamayı hedeflemiştir. Deneysel veriler, iyileştirilmiş uyarlanabilir ACO'nun, bulut kaynak zamanlama sorunlarına en iyi çözümü hızla bulabildiğini, görev tamamlama süresini kısalttığını, yürütme maliyetini azalttığını ve tüm bulut sisteminin yük dengesini koruyabildiğini göstermektedir. Yöntem büyük ölçekli görev çözelgeleme problemlerinde daha iyi performans gösterdiği belirtilmiştir.

Emami [45], görev çözelgeleme performansını iyileştirmek için Geliştirilmiş Ayçiçeği Optimizasyon Algoritması (Enhanced Sunflower Optimization Algorithm, ESFO) algoritmasını önermiştir. Deneysel sonuçlar ESFO'nun kendisine benzeyen en iyi yöntemle karşılaştırıldığında

makespan süresi ve enerji tüketimi açısından sırasıyla %0.73 ve %2.24 oranında iyileştirme yaptığını göstermiştir.

3. Evrimsel Algoritmalar

Evrimsel algoritmalar doğal seçim ve genetik gibi biyolojik süreçlerden ilham alarak karmaşık problemlere çözüm bulmaya çalışan bir tür yapay zeka algoritmalarıdır. Bu yöntemler birçok probleme uyarlanabilir olması tek bir çözüm yerine birden fazla çözüm bulmaları ve güçlü yöntemler olması nedeniyle birçok alanda kullanılmaktadır.

Tamanna ve Mohanty [46] çoklu bulut sistemlerde kaynak sağlama ve görev çizelgelemede GA temelli müşteri bilinçli yöntem önermiştir. Yöntem GA temelli kaynak tahsisi ve en kısa görevin ilk olarak icra edildiği iki aşamadan oluşmaktadır. Çoklu bulut ortamında görevler sanal sunucularla eşleştirilirken görevlerin icra süresinin minimize edilmesi ve kullanıcı memnuniyetinin maksimize edilmesi amaçlanmıştır. Yapay veriler üzerinde gerçekleştirilen deneysel sonuçlar önerilen algoritmanın mevcut algoritmalara göre daha iyi sonuçlar verdiğini göstermiştir.

Yiqiu ve arkadaşları [47], ikili kodlanmış kromozomlar kullanılarak Uyarlanabilir Genetik Algoritma (Adaptive Genetic Algorithm, AGA) çapraz ve mutasyon olasılıkları üzerinde bir başlangıç optimizasyon yöntemi önermiştir. Yöntem, mevcut AGA ve GA ile karşılaştırılmıştır. Deneysel sonuçlar, geliştirilen algoritmanın bulut bilişim görev planlama süreçlerinde etkili bir yöntem olduğunu ortaya koymaktadır.

Sharma ve Garg [48] bulut sistemlerde görev çizelgeleme yaparken enerji verimliliğini dikkate alan yeni bir hibrit metasezgisel şema olan Hibrit Metasezgisel Şema Uyumundan Esinlenen Genetik Algoritma (Hybrid Metaheuristic Scheme Harmony-Inspired Genetic Algorithm, HIGA) önermiştir. HIGA, genetik algoritmanın keşif yeteneğini, uyum aramasının ise yerel ve küresel optimum bölgeleri hızlı bir şekilde tespit ederek zaman kaybetmeden yakınsama sağlama kapasitesini bir araya getirmektedir. Çalışmadaki temel hedef, makespan süresi ve hesaplama enerjisini azaltmaktır. Diğer hedefler ise hesaplama kaynakları dışındaki kaynaklar tarafından tüketilen enerjiyi ve zamanlayıcıyla ilişkili yürütme ek yükünü azaltmaktır. Simülasyon sonuçları, hem bağımsız görev uygulamaları hem de CyberShake, Epigenomics ve Montage gibi gerçek dünya bilimsel uygulamaları üzerinde gerçekleştirilmiştir. Sonuçlar, HIGA'nın %33'e kadar enerji tasarrufu sağladığını ve uygulama performansını (makespan) %47'ye kadar iyileştirdiğini, bunun da %39 daha az yürütme yüküyle elde edildiğini göstermiştir.

Agarwal ve Gupta [49] bulut sistemlerde görevlerin etkin bir şekilde çizelgelemesi ve kaynakların optimal kullanımı amacıyla yeni bir uyarlanabilir Genetik Algoritma Tabanlı Yük Dengeleme (Genetic Algorithm-Based Load Balancing, GALB) yöntemini önermiştir. Bu yöntem, sistemin toplam performansını artırıp, aynı zamanda makespan süresi, performans iyileştirme oranı ve sistem dengesizliği gibi kritik performans göstergelerini de iyileştirmektedir. Algoritmada kullanılan uyarlanabilir çaprazlama ve mutasyon operatörleri sayesinde, mevcut neslin en uygun bireylerine

odaklanılarak çözüm uzayında daha etkili bir arama gerçekleştirilmektedir. CloudSim simülasyon ortamında gerçekleştirilen kapsamlı deneyler, GALB algoritmasının, geleneksel yöntemlere kıyasla tüm performans metriklerinde belirgin bir üstünlük sergilediğini göstermiştir.

Muhannad ve arkadaşları [50] Siyah Pencere Optimizasyonu (Black Window Optimization, BWO) yönteminin iyileştirilmiş bir türünü önermiştir. BWO'nun zayıf noktası, rastgele bir seçim tekniği kullanmasının, çeşitliliğin azalmasına, hızlı yakınsamaya ve yerel optimumlarda takılma riskine sahip olmasıdır. Önerilen yöntem, BWO algoritmasını geliştirmek amacıyla alternatif seçim şemalarını entegre eden yeni bir yaklaşım sunarak mevcut seçim metodolojisinin sınırlamalarını aşmayı hedeflemiştir. Bu önerilen varyantların etkinliğini değerlendirmek için, standart değerlendirme ölçütü olarak CEC 2019 kıyaslama fonksiyonları kullanılmıştır. Daha sonra, bulut görev çizelgeleme sorunlarını çözmek amacıyla, en iyi performans gösteren BWO sürümü olan PIBWO kullanılmıştır. Bir dizi karşılaştırmalı deneyde, PIBWO'nun mevcut algoritmalara kıyasla üstün performans gösterdiği, makespan süresini azaltmada, enerji tüketimini en aza indirmede ve maliyet verimliliğini artırmada önemli iyileştirmeler sağladığını göstermiştir.

Saravanan ve Ramesh ve arkadaşları [51] sanallaştırılmış bulut ortamındaki gerçek zamanlı görevler için, Biyo-İlham Veren Enerji Verimli Dinamik Görev Çizelgeleme (Bio-Inspired Energy Efficient Dynamic Task Scheduling, BEDTS) yöntemini önermiştir. BEDTS algoritmasında, kaynak kısıtlamaları ve VM'ler için en uygun seçimi sağlamak amacıyla Uyarlanabilir Fil Sürü Optimizasyonu (Adaptive Elephant Herding Optimization, AEHO) tekniği uygulanmıştır. İlk aşamada, önceki planlama kayıtları kullanılarak heterojen işler ve VM'ler tanımlanır. Görev ve VM türlerinin belirlenmesinde Bayes sınıflandırıcı ve Tarihsel Çizelgeleme Kaydı (Historical Scheduling Record, HSR) kullanılarak görev kategorizasyonu gerçekleştirilir. Ardından, ilgili görevler birleştirilir ve ana bilgisayarın operasyonel durumundan en iyi şekilde yararlanmak üzere planlanır. Önceki yaklaşımlarla karşılaştırıldığında, deneysel sonuçlar BEDTS yönteminin planlama performansını önemli ölçüde artırdığını, bulut sistemlerin kaynak kullanımını iyileştirdiğini, ortalama yanıt süresini azalttığını ve enerji tüketimini düşürdüğünü göstermektedir.

Kaur ve Verma [52], görev yürütüm süresini ve yürütme maliyetini en aza indiren metasezgisel tabanlı bir çizelgeleme algoritması önermiştir. İşlem elemanlarının hesaplama karmaşıklığı ve hesaplama kapasitesi dikkate alınarak görevleri planlamak amacıyla mevcut iki planlama algoritmasının birleştirilmesiyle iyileştirilmiş bir genetik algoritma geliştirilmiştir. Deneysel sonuçlar, önerilen algoritmanın ağır yükler altında iyi bir performans sergilediğini ortaya koymaktadır.

Çok Amaçlı Metasezgisel Yöntemler

Literatürde bulut sistemlerde görev çizelgelemenin çok amaçlı optimizasyonu ile ilgili birçok çalışma yapılmıştır. Bu çalışmalar 5 başlık altında sınıflandırılabilir.

1. Evrimsel Algoritmalar

Çok Amaçlı Evrimsel Algoritma (Multi Objective Evolutionary Algorithms, MOEA) performansının yerel arama ile geliştirilebileceği gösterilmiştir. MOEA'lar ile hibrit çözümler, pareto cephesine yakınsama hızını artırır. Materwala ve Ismail [53] , çelişen performans ve enerji hedeflerini dengelemek amacıyla çok amaçlı bir yöntem önermiştir. Önerilen modelin performansı, gerçekçi bir bulut veri kümesi kullanılarak bulut bilişim ortamında değerlendirilmiştir. Deneysel sonuçlar, önerilen modelin, mevcut en son teknoloji algoritmalarla karşılaştırıldığında daha yüksek performans ve daha düşük enerji tüketimi sağladığını göstermiştir.

Zhang ve arkadaşları [54], bulut sistemlerde görev çizelgelemenin belirsizlikleri yönetmek için yeni bir yaklaşım önermiştir. Oluşturma süresi, çizelgeleme maliyeti, yük dengesi ve görev tamamlama oranı gibi çok amaçlı bir optimizasyon problemi olarak formüle edilen görev çizelgeleme problemi için, Aralıklı Çok Amaçlı Evrimsel Algoritma (An Interval Many-Objective Evolutionary Algorithm, InMaOEA) önerilmiştir. InMaOEA, aralık güvenilirliği stratejisi ve aralık tıkanıklık mesafesi tabanlı hiper hacim ve örtüşme derecesi gibi mekanizmalarla güçlendirilerek, çözüm uzayında daha etkili bir arama yapmaktadır. Yapılan kapsamlı simülasyon deneyleri, InMaOEA algoritmasının diğer algoritmalarla göre daha iyi performans gösterdiğini ve karar vericilere daha güvenilir ve esnek çözümler sunduğunu kanıtlamıştır.

Wang ve ekibi [55] verimli sanal makine atama ve görev çizelgeleme problemlerini ortak bir çizelgeleme problemi olarak modelleyen ve makespan süresinin, maliyet ve toplam gecikmenin minimize edilmesini amaçlayan bir yöntem (MOEA/D-PD) önermiştir. Bu yöntem, problemi daha küçük parçalara ayırarak ve her parçayı ayrı ayrı çözerek daha etkili bir yaklaşım sunmuştur. Ayrıca, hangi parçanın daha önemli olduğuna karar vererek, çözüm sürecinde kaynakları verimli bir şekilde kullanmıştır. Deneysel sonuçlar yöntemin benzer yöntemlere göre daha başarılı olduğunu göstermiştir.

Sharma ve arkadaşları tarafından [56] sinir ağları ile güçlendirilmiş yeni bir ACO tabanlı çok amaçlı bir algoritma önerilmiştir, Yöntem, belirli sayıda görev ve sınırlı kaynaklar altında, görevlerin en iyi şekilde planlanması için küresel arama gerçekleştirir. Simülasyon sonuçları, önerilen algoritmanın diğer yöntemlere göre daha kaliteli çözümler ürettiğini ve özellikle büyük ölçekli problemlerde daha iyi performans gösterdiğini ortaya koymuştur.

2. Sürü Zekâsı Algoritmalar

Sürü zekâsı algoritmaları, doğadan esinlenen ve sürü içerisindeki çok sayıdaki bireylerin birbirleriyle olan etkileşimlerini karmaşık problemleri çözmeye çalışan yöntemlerdir. Sürü zekâsı yöntemleri, genellikle çok amaçlı optimizasyon problemlerinde etkin çözümler bulmak için kullanılır.

Tamilarasu ve Singaravel [57] bulut sistemlerde maliyet ve zaman kriterlerini dikkate alarak görev çizelgeleme yapan bir yöntem önermiştir. Bu çok amaçlı optimizasyon yöntemi görevlerin icra süresini minimize ederken kaynak kullanımını maksimize etmeye çalışmıştır. Mevcut metasezgisel

yöntemler ile bu yaklaşım kıyaslandığında kıyaslanan yöntemlere göre bariz bir üstünlük kurduğu belirtilmiştir.

Imene ve ekibi [58] ilk kez, buluttaki bir dizi kullanılabilir VM-kullanıcı görevlerini çizelgelemek için çalışma süresi, güç tüketimi ve maliyeti en aza indiren çok amaçlı bir optimizasyon yöntemi olan Domine Olmayan Sıralama Genetik Algoritması-3 (Non-dominated Sorting Genetic Algorithm-3, NSGA-3) kullanmıştır. Ayrıca, NSGA-3'ün performansı, önceki sürüm olan Domine Olmayan Sıralama Genetik Algoritması-2 (Non-dominated Sorting Genetic Algorithm-2 NSGA-2) ile karşılaştırılmış ve NSGA-3 sonuçlarının NSGA-2 sonuçlarından daha iyi olduğu beirtilmiştir.

Hosseini ve arkadaşları [59] sis platformlarında IoT uygulamalarının yürütülmesi için çizelgeleyici önermiştir. Bu yöntem makespan süresini en aza indirirken güvenilirlik yürütmesini en üst düzeye çıkarmayı amaçlamaktadır. Görevlerin çizelgelemesi problemini, makespan süresi ve güvenilirlik optimizasyonu bakış açısıyla çok amaçlı ayrık bir forma dönüştürülmüştür. Bu kombinasyonel problemi çözmek için Çok Amaçlı Ayrık Guguk Kuşu Arama Algoritması (Multi-Objective Discrete Cuckoo Search Algorithm, MoDCSA) geliştirilmiştir. MoDCSA optimizasyon algoritması, akıllıca entegre edilmiş çeşitli ayrık operatörlerin avantajlarını kullanır. Geniş çaplı simülasyon senaryolarından elde edilen sonuçlar, MoDCSA planlama algoritmasının etkinliğini doğrulamıştır. Bu algoritma, Çok Amaçlı Arıların Yaşam Algoritması (Multi-objective Bees Life Algorithm, MoBLA), Çok Amaçlı Genetik Algoritma (Multi-objective Genetic Algorithm, MoGA), Çok Amaçlı Gri Kurt Optimizasyonu (Multi-objective Grey Wolf Optimization, MoGWO), Çok Amaçlı Simüle Tavlama (Multi-objective Simulated Annealing, MoSA) ve Çok Amaçlı Genetik Simüle Tavlama (Multi objective Genetic-SA, MoGSA) makespan süresi açısından sırasıyla %2.27, %3.89, %5.45, %5.40 ve %4.55; güvenilirlik açısından ise sırasıyla %1.99, %5.11, %6.39, %7.95 ve %7.78 oranında iyileştirme sağlamıştır.

Abdullahi ve arkadaşları [60] IaaS bulut ortamlarında büyük ölçekli çok amaçlı görev planlama problemini çözmek için yeni bir metasezgisel algoritma olan Kaotik Simbiyotik Organizmalar Arama (Chaotic Symbiotic Organisms Search, CMSOS) algoritması önermiştir. CMSOS, kaotik sistemlerin dinamik ve kaotik davranışlarından yararlanarak, çözüm uzayında daha geniş bir alanı araştırarak daha iyi çözümlere ulaşmayı amaçlar. Algoritmada, başlangıç popülasyonunun oluşturulması için kaotik bir yaklaşım kullanılmış ve Simbiyotik Organizmalar Arama (Symbiotic Organisms Search, SOS) algoritmasının fazları, kaotik dizilerle değiştirilerek çözüm çeşitliliği artırılmıştır. Ayrıca, kaotik yerel arama stratejisi sayesinde, algoritma yerel optimum noktalarından kaçınarak daha iyi pareto cephelerine ulaşmaktadır. Simülasyon deneyleri, CMSOS algoritmasının, standart iş yükleri ve büyük ölçekli sentetik iş yükleri üzerinde, diğer çok amaçlı görev çizelgeleme algoritmalarına göre daha iyi performans gösterdiğini ortaya koymuştur. Özellikle, yürütme süresi ve maliyet gibi çelişen amaçlar arasında daha iyi bir denge kurarak, QoS gereksinimlerini daha iyi karşılamaktadır.

Ramezani ve ekibi [61] bulut sistemlerde görev çizelgelemesini optimize etmek amacıyla görev yürütme süresi, görev transfer süresi ve görev yürütme maliyetini en aza indiren kapsamlı çok amaçlı

bir model geliřtirmiřtir. Önerilen model PSO algoritmasının hız ve doğruluk avantajları dikkate almaktadır. Yöntem optimum çözüm için Çok Amaçlı Parçacık Sürü Optimizasyonu (Multi Objective Particle Swarm Optimization, MOPSO) yöntemine dayanmaktadır. Simülasyon sonuçları, yöntemin çatışan hedefler arasında en iyi takas sağlayarak çok amaçlı görev planlama problemlerinde optimal takas çözümleri sunduğunu ve QoS önemli ölçüde artırdığını ortaya koymuştur.

Entisar ve arkadaşları [62], çok amaçlı optimizasyon ve PSO algoritması kullanarak görev planlamasını optimize eden yeni bir yöntem önermiştir. Bu yöntem temel olarak, görevlerin bekleme süresini en aza indirmek ve sistem verimini en üst düzeye çıkarmak için çizelgeleme yapmaktadır. Algoritma, yürütme süresinde %20, bekleme süresinde %30 oranında azaltılmış ve mevcut teknoloji düzeyine kıyasla verimde %40'a kadar iyileřtirmeler göstermiştir.

Mangalampalli ve arkadaşları [63], bulut sistemlerde görevlerin etkin bir şekilde çizelgelemesi için çok amaçlı bir güven farkında zamanlayıcı önermiştir. Önerilen zamanlayıcı, görevlerin sanal makinelere atanması ve zamanlamasının, hem makespan süresini hem de enerji tüketimini en aza indirerek, sistemin güvenilirliğini de artırmayı hedeflemiřtir. Zamanlama problemi, WOA kullanılarak modellenmiştir. Simulasyonlar CloudSim aracı kullanılarak gerçekleştirilmiştir. Elde edilen sonuçlar, önerilen yöntemin ACO, GA ve PSO gibi yaygın kullanılan metasezgisel yöntemlere kıyasla makespan süresi, enerji tüketimi, toplam çalışma süresi ve güvenilirlik metriklerinde önemli ölçüde iyileřme sağladığını göstermiştir.

Khan ve Rasool [64] bulut sistemlerde görev çizelgelemesi yapmak için çok amaçlı GWO'ya dayalı bir yöntem önermiştir. Yöntem, görevlerin yürütülmesi için gereken çizelgeleme süresini, enerji tüketimini ve maliyeti en aza indirmeyi hedeflemiřtir. CloudSim aracı kullanılarak gerçekleştirilen deneyler önerilen algoritmanın, planlama uzunluğu, maliyet ve enerji tüketimi gibi kriterler açısından mevcut algoritmalara kıyasla %3.98 ile %16.07 arasında deęişen performans iyileřtirmeleri sağladığını göstermiştir.

Zuo ve arkadaşları [65], makespan süresini ve kullanıcı maliyetlerini optimize edilecek amaçlar olarak alan bir gelişmiş bir ACO algoritması önermiştir. Performans ve bütçe maliyetini deęerlendirmek ve geri bildirim sağlamak için iki kısıtlama fonksiyonu kullanılmıştır. Bu iki kısıtlama fonksiyonu, algoritmanın optimum çözümü elde etmek için geri bildirim dayalı olarak çözümün kalitesini zamanında ayarlamasını sağlamıştır. Yöntemin performansı makespan, maliyet, son icra tarihi ihlal oranı ve kaynak kullanımı deęerleri üzerinden deęerlendirilmiştir. Deneysel sonuçlar, bu dört ölçüte dayalı olarak çok amaçlı bir optimizasyon yönteminin dięer benzer yöntemlerden daha iyi olduğunu, özellikle de en iyi senaryoda %56.6 oranında arttığını göstermiştir.

Got ve arkadaşları [66] WOA'nın gelişmiş yeni bir türünü önermiştir. Önerilen algoritma, GÜdümlü Nüfus Arşivi Balina Optimizasyon Algoritması (Guided Population Archive Whale Optimization Algorithm, GPAWOA) adını taşır ve pareto baskınlığına dayanmaktadır. Optimizasyon sırasında elde edilen baskın olmayan çözümler, harici bir arşivde depolanır. Liderler, popülasyonu arama alanının potansiyel bölgelerine yönlendirmek amacıyla bu arşivden seçilir. GPAWOA, üç iyi

bilinen algoritmayla (MOEA/D, MOGWO ve MOPSO) karşılaştırılmıştır. Deneysel sonuçlar, önerilen algoritmanın rekabet gücünün yüksek olduğunu ve pareto cephesine yakınsama ve çeşitlilik açısından bu son teknoloji algoritmalarından daha iyi performans gösterdiğini ortaya koymaktadır.

3. Yerel Arama Algoritmaları

Yerel arama yöntemleri mevcut bir çözümden hareketle çevresindeki komşu çözümler iyileştirme yapmak amacıyla kullanılır. Bu yöntemler optimizasyon problemlerinde daha etkili çözümler sağlamak için yerel arama tekniklerini ve çoklu hedefleri birleştirir. Bu algoritmalar, çok amaçlı optimizasyonun zorluklarını ele almak için yerel arama stratejilerini kullanarak, çözüm uzayında daha iyi sonuçlar elde etmeyi hedefler. MOSA, Çok Amaçlı Tabu Arama(Multi-Objective Tabu Search, MOTS) ve Çok Amaçlı Evrimsel Yerel Arama (Multi-Objective Evolutionary Local Search, MOELS) bu türden yöntemlere örnek olarak gösterilebilir.

Chen ve arkadaşları [67] Domine Olmayan Sıralama Yerel Arama (Nondominated Sorting Local Search, NSLS) temelli yinelemelere dayanan yeni çok amaçlı bir optimizasyon yapısı önermiştir. NSLS, popülasyon boyutunu sabit tutar. Her yinelemede, mevcut popülasyondaki her çözüm için, önce mevcut çözümdeki her değişken için iki komşuluk oluşturmak üzere önerilen bir yerel arama şemasını kullanır. Ardından, iki komşuluğun mevcut çözümü değiştirip değiştirmeyeceğine ve mevcut çözümü değiştirmek için hangisinin seçileceğine karar vermek için bir değiştirme stratejisi kullanılır. Bu süreç, mevcut çözümün tüm değişkenleri işlenene kadar devam eder. Popülasyondaki tüm çözümler işlendiğinde, bir sonraki yineleme için yeni popülasyonu seçmek üzere hızlı baskın olmayan sıralama ile birleştirilmiş en uzak aday yaklaşımı benimsenir. Karşılaştırma amacıyla, NSLS'in bir başka versiyonu olan NSLS-C kullanılmıştır. NSLS-C, en uzak aday yöntemini, NSGA-2 tarafından sunulan kalabalık karşılaştırma mekanizması ile değiştirmiştir. Önerilen NSLS yöntemi on yedi iki amaçlı ve üç üç amaçlı test problemi üzerinde karşılaştırılmıştır. Deneysel sonuçlar, NSLS'in, karşılaştırılan diğer algoritmalarla kıyasla çözümlerin daha iyi bir yayılımını ve gerçek pareto-optimal ön yüzüne daha iyi bir yakınsama sağladığını göstermiştir.

4. Hibrit Algoritmalar

Çok amaçlı hibrit optimizasyon yöntemleri, çeşitli optimizasyon tekniklerini birleştirerek çoklu hedefleri aynı anda optimize etmeye yönelik yaklaşımlardır. Bu yöntemler, farklı algoritmaların güçlü yönlerini birleştirerek daha iyi sonuçlar elde etmeyi hedefler. Hibrit optimizasyon, genellikle global ve yerel arama stratejilerinin entegrasyonunu içerir ve böylece hem geniş bir arama alanı hem de yerel iyileştirme sağlanır.

Pabitha ve arkadaşları [68] görev çizelgemelemesi yaparken sanal makine performansını doğrudan etkileyen komut sayısı, ağ bant genişliği gibi etkenlerin etkisini dikkate alarak Bukalemun ve Remora Arama Optimizasyon Algoritması (Chameleon and Remora Search Optimization Algorithm, CRSOA) önermiştir. Bu yöntem görev tamamlama oranı, yük dengesi, planlama maliyeti ve görev

yürütüm süresini eş zamanlı optimize eden çok amaçlı bir yöntemdir. Deneysel sonuçlar yöntemin görev yürütüm süresinin minimize edilmesi, mevcut sanal makineler arasında yük dengelemede başarılı olduğunu göstermiştir.

Behera ve Sobhanayak [69] bulut sistemlerde görev çizelgeleme için GWO ve GA yöntemlerini kombine etmiştir. Önerilen GWO-GA yöntemi çok amaçlı bir optimizasyon yöntemi olup; görevlerin icra süresinin, enerji tüketiminin ve maliyetlerin minimize edilmesini amaçlar. Bu yöntem genetik algortmada kullanılan mutasyon ve çaprazlama operatörlerini kullanır. Büyük çaplı görev çizelgeleme problemlerinde optimum çözüme daha hızlı bir yakınsama gösterir. Deneyler hem sentetik veriler hem de gerçek veriler üzerinde gerçekleştirilmiştir. Deneysel sonuçlar makespan, enerji tüketimi ve hesaplama maliyetinin minimize edilmesi yönünden yöntemin başarılı olduğunu doğrulamıştır. Bu yöntem GWO, GA ve PSO algoritmaları yönünden makespan, maliyet ve enerji tüketimi yönünden kıyaslanmıştır. Yöntem makespan açısından GWO, GA ve PSO'ya göre %19, %21 ve %15 daha iyiyken enerji tüketimi açısından %17, %19, %23 toplam çizelgeleme maliyeti açısından %13, %17 ve %22 daha iyi olduğu belirtilmiştir.

Dubey ve Sharma [70] birden fazla sanal makinede çalıştırılması gereken bağımsız görevlerin, maliyet, enerji tüketimi ve tamamlanma süresi gibi kriterlere göre en uygun şekilde planlanması amaçlanmıştır. Bu amaçla, kimyasal reaksiyon optimizasyonu ve kısmi sürü optimizasyonu algoritmalarının güçlü yanlarını bir araya getiren hibrit bir algoritma olan Kimyasal Reaksiyon Kısmi Sürü Optimizasyonu (Chemical Reaction Partial Swarm Optimization- CRPSO) geliştirilmiştir. CRPSO algoritması, görevlerin talep ve son icra tarihlerini dikkate alarak, en uygun çizelgelemeyi bulmayı hedeflemektedir. Simülasyon sonuçlarından CRPSO algoritmasının, ortalama yürütme süresini %1-%6 makespan süresini %5-%12, toplam maliyeti %2- %10 ve enerji tüketimini %1-%9 oranında iyileştirdiği gözlemlenmiştir.

Suresh ve arkadaşları [71] Çok Amaçlı Kedi Sürü Optimizasyon algoritması (Multi-Objective Cat Swarm Optimization Algorithm, MCSOFLB) kullanılarak bir hata toleranslı yük dengeleme yöntemini önermiştir. MCSOFLB yöntemi, kıyaslama algoritmalarına göre ortalama %31 daha kısa makespan, %6 daha iyi kaynak kullanımı, %12 daha düşük maliyet, %6 daha yüksek başarı oranı ve %32 daha fazla ortalama verimlilik sağlamıştır.

Zhao ve arkadaşları [72] risk oranı, kaynak kullanımı, toplam maliyet ve görev tamamlama süresini bir araya getiren çok amaçlı Hibrit Bulut Görev Çizelgeleme Optimizasyon Modeli (Hybrid Cloud Task Scheduling Optimization Model- HCTSO) önermiştir. Bu modelin etkinliğini arttırmak için, Karşıtlık Temelli Öğrenme Algoritması (Opposition-Based Learning, OBL) ve Diz Noktasına Dayalı Çok Amaçlı Evrimsel Algoritma (Knee Point-Driven Many-Objective Evolutionary Algorithm, KnEA) birleştirilerek OBL-KnEA adı verilen bir algoritma geliştirmiştir. OBL-KnEA, başlangıç popülasyonlarını daha hızlı yakınsamaya ulaşmak için muhalefet tabanlı öğrenme ile oluşturulmuştur. Ayrıca, arama alanını genişletmek amacıyla bozulma tabanlı çok noktalı çaprazlama operatörü ve dinamik aralık mutasyon operatörü tasarlanmıştır. Yapılan deneyler, OBL-KnEA algoritmasının, diğer

üstün algoritmalarla karşılaştırıldığında değerlendirme ölçütleri, başlangıç popülasyonları ve model optimizasyonu açısından üstün performans sergilediğini göstermektedir.

Sadip ve ekibi [73] yol kullanıcılarından gelen çok sayıda görev isteğini verimli bir şekilde karşılamak ve aynı zamanda QoS iyileştirmek için optimize edilmiş bir kaynak tahsisi ve görev çizelgeleme algoritması önermiştir. Bu görev istekleri, üç bulut katmanı arasındaki bulut kaynaklarına en iyi şekilde eşlenir. Optimizasyon süreci, GA ve Uyarlanabilir PSO'nun bir kombinasyonu olan önerilen Hibrit Uyarlanabilir Parçacık Sürü Optimizasyonu (Hybrid Adaptive Particle Swarm Optimization, HAPSO) algoritması kullanılarak gerçekleştirilir. Deneysel sonuçlar ayrıca görev yanıt süresinde ~%34 iyileşme ve ~%32.5'e kadar enerji tüketiminde azalma olduğunu göstermektedir.

Zheng ve arkadaşları [74], kurtarma görevi çizelgelemesinin bulanık çok amaçlı optimizasyon problemini ele almışlardır. Amaç, görev çizelgeleme verimliliğini en üst düzeye çıkarırken, kurtarma ekipleri için operasyonel riskleri en aza indirmektir. Problemi çözmek amacıyla, Verimli Çok Amaçlı Biyocoğrafya Tabanlı Optimizasyon (Efficient Multiobjective Biogeography-Based Optimization, EMOBBO) algoritması önerilmiştir. Belirsizlikleri yönetmek için, üç ilişkili bulanık sıralama kriteri ve çözümlerin baskınlık ilişkisini değerlendirmek için bulanık baskınlık kavramı kullanılmıştır. EMOBBO algoritmasında, permütasyon tabanlı çözümleri iyileştirmek amacıyla yeni göç ve mutasyon operatörleri tanımlanmış, ayrıca verimsiz çözümleri elemek için soruna özgü çözüm yeniden düzenleme mekanizması uygulanmıştır. Ayrıca, erken yakınsamayı önlemek için yerel bir mahalle yapısı kullanılmıştır. Deneylerde, EMOBBO algoritmasının son teknoloji evrimsel çok amaçlı optimizasyon algoritmalarına kıyasla daha üstün performans göstermiştir.

Ramezani ve arkadaşları [75] görev transfer süresini, görev yürütme maliyetini, güç tüketimini ve görev kuyruğu uzunluğunu en aza indirme gibi dört çelişen hedefi ele alarak görev çizelgelemesi için kapsamlı çok amaçlı bir optimizasyon modeli önermiştir. Bu model, yürütme ve güç maliyetini göz önünde bulundurarak hem müşteri hem de sağlayıcı açısından maliyetleri azaltır. Önerilen model, MOPSO ve MOGA yöntemlerine uygulanmış ve sonuçlar değerlendirilmiştir. CloudSim aracı kullanılarak gerçekleştirilen simülasyonlarda, önerilen çok amaçlı modelin dört çelişen hedef arasında optimum takas çözümleri bulunduğunu ve bu da iş yanıt süresini ve üretim süresini önemli ölçüde azalttığını göstermiştir. Bu model yalnızca QoS'u arttırmanın yanında, aynı zamanda sağlayıcılar için maliyeti de azaltır. Deney sonuçları, MOPSO'nun bu tür sorunları çözmek için MOGA'dan daha hızlı ve daha doğru bir evrimsel algoritma olduğunu göstermiştir.

5. Adaptif Algoritmalar

Problem çözme süreçlerinde, değişen koşullara dinamik olarak uyum sağlayabilen ve birden fazla amacı aynı anda optimize etmeye çalışan algoritmalarlardır. Çözüm kalitesini artırmak için çeşitli adaptasyon stratejileri kullanır ve problem yapısına göre kendilerini yeniden yapılandırırlar. Bu sayede, çözüm uzayında daha etkili bir arama yaparak daha iyi sonuçlara ulaşmayı hedeflerler.

Guo [76], bulut sistemlerde ortamlarında çok amaçlı görev planlama problemi için, bulanık öz savunma algoritmasına dayalı yeni bir optimizasyon yöntemini önermiştir. En kısa tamamlanma süresi, kaynak yük dengesi ve toplam görev tamamlama maliyeti gibi birden fazla çelişen amacı aynı anda optimize etmek üzere tasarlanan bu yöntem, bulanık mantık ve evrimsel algoritmaların güçlü yanlarını bir araya getirmektedir. Deneysel sonuçlar bu yöntemin çok amaçlı görev planlama probleminde, özellikle de görevlerin tamamlanma süreleri, kaynak kullanım etkinliği ve son icra tarihi ihlal oranları açısından önemli ölçüde iyileşme sağladığını göstermektedir.

1.3. Tezin Yapısı

Tez, yapısal olarak şu şekilde hazırlanmıştır; Giriş bölümünde tezin amacı, bulut sistemlerde görev çizelgeleme ile ilgili önerilen yöntemler türlerine ve amaç sayılarına göre kategorize edilmiş ve geniş bir literatür bilgisi ve tezin planı verilmiştir. 2. bölümde bulut sistemler, görev çizelgeleme, metasezgisel yöntemler, optimizasyon ve görev çizelgeleme süreci hakkında bilgiler verilmiştir. 3. Bölümde önerilen kural tabanlı yöntem ve 4. Bölümde önerilen hibrit yönteme ait detaylar verilmiştir. 5. Bölümde bölümünde önerilen yöntemlere ilişkin deneysel sonuçlar irdenelerek genel sonuçlar verilmiştir. Tez 6. bölümdeki tartışma bölümü ile sonlanmıştır.

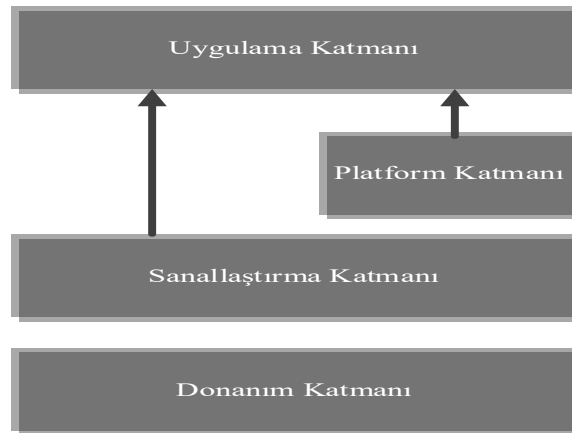
2. GENEL BİLGİLER

Bu bölüm bulut sistemlere, bulut sistemlerde görev çözelgelemeye sürecine ait detaylı bir bakış sağlar. Ayrıca optimizasyon ve metasezgisel yöntemlere ilişkin genel bilgiler bu bölümde verilmiştir.

2.1. Bulut Bilişim

Bulut bilişim, sahip olunan bilgi işlem kaynaklarını (sunucular, depolama kaynakları ve uygulamalar vs.) kullanıcılarına minimum yönetim ve güvenlik çabası ile kiralayabilen\paylaşabilen Nesnelerin İnterneti (Internet of Things, IoT) çözümleridir. Sağladığı donanımsal ve yazılımsal avantajlarla bulut bilişim, bugün Bilgi Teknolojisi (Information Technology, IT) dünyasının en büyük sektörlerinden biridir. 2022 yılında 400 milyar doların üzerinde gelire ulaşan ve çok sayıda sağlayıcının hizmet verdiği bulut piyasasının ekonomik büyüklüğünün 2025 yılında 800 milyar doların üzerine çıkacağı beklenmektedir [77–79]. Bununla birlikte, 2026 yılına kadar kurumların bilişim teknolojileri harcamalarının %45’inden fazlasının bulut teknolojileri için olacağı da öngörülmektedir [80].

Bulut sistemler bünyesinde barındırdığı ve birbiriyle bağlı olan çok sayıda bilgisayar ile muazzam miktarda veri işleme ve depolama kapasitesine sahiptir. Bu bilgisayarlar kişisel bilgisayarlar olabileceği gibi güçlü sunucular da olabilir. Bu bilgisayar bulutu tarafından sunulan uygulamalar geniş bir kullanıcı kitlesi veya kuruluşlar tarafından kullanılır. Bulut sistemlere erişim yetkisi bulunan herhangi bir kullanıcı internet aracılığıyla sisteme bağlanabilir ve işlemlerini yürütebilir. Kullanıcı bulut sisteminde kullanılan alt yapı, teknoloji ve sistemin yürütümü ile ilgili işlemlerden bağımsızdır. Bulut sistemlerin sistem mimarisi Şekil 2.1’de gösterilmiştir. En alt katman olan donanım katmanı veri iletme, bellek ve ağ kaynaklarının olduğu kısımdır [81]. İkinci katman kullanıcı isteklerine göre kaynakların sunulabilmesi için sanallaştırılmıştır. Platform katmanı sanallaştırma katmanı katmanı üzerine özel araçlar, hizmetler ve yazılımlar ekler. Uygulama katmanı bulut sistemlerde sunulan uygulamaların olduğu katmandır.

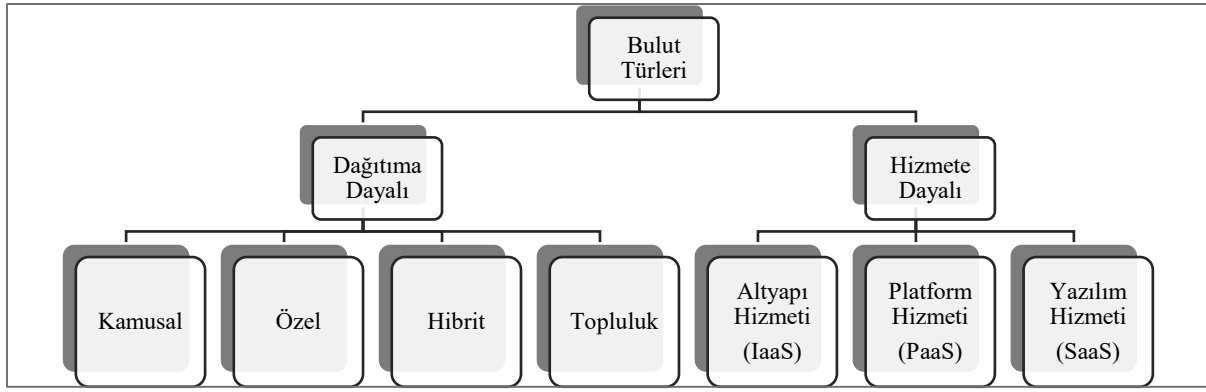


Şekil 2.1. Bulut sistem mimarisi

Uygulamalarda bulut sistemlere alternatif olarak kullanılabilecek başka yöntemler (dağıtık veri işleme ve depolama sistemleri vs.) mevcuttur. Bulut sistemleri diğer sistemlerden farklı kılan önemli özellikleri şöyle sıralanabilir:

- **Talep Üzerine Hizmet:** Kullanıcıların ihtiyaç duydukları kaynakları (veri işleme kapasitesi, depolama veya ağ bant genişliği) herhangi bir destek almadan kendileri ayarlayabilirler.
- **Kaynak Havuzu:** Bulut sistemler kaynaklarını dinamik olarak birden fazla kullanıcı arasında paylaşabilirler. Bu durum maliyetlerin düşmesine ve kaynakların daha verimli kullanılmasına olanak sağlar.
- **Geniş Ağ Erişimi:** Bulut sistemlere internet üzerinden çeşitli cihazlarla (tablet, mobil cihaz ya da bilgisayarlar) erişim sağlanabilmektedir. Bu durum kullanıcılarına esnek çalışma imkânı sağlamaktadır.
- **Kullandığın Kadar Ödeme:** Bulut sistemler kullanıcının kullanmış olduğu kaynak miktarına göre ödeme yapma imkânı sağlar. Bu durum işletmelerin ihtiyaç duyduğu kaynak miktarı için ödeme yapmasına olanak sağlar.
- **Otomasyon:** Bulut sistemlerde yürütülen operasyonel süreçler otomatikleştirilmiştir. Bu durum insan müdahalesi olmadan otomatik olarak gerçekleştirilir. Süreçlerin hızlanması ve insan hatasının minimize edilmesi gibi çeşitli avantajları vardır.
- **Esneklik ve Entegrasyon:** Bulut sistemler farklı platformlara veya yazılımlara kolayca entegre edilebilir. Bu durum kullanıcıların değişen ihtiyaçlarına hızlı adapte olabilme imkânı sağlar.
- **Hızlı Esneklik ve Ölçeklenebilirlik:** Kullanıcıların bulut sistemlerinden talep ettiği kaynak miktarı zamana bağlı olarak değişkenlik gösterebilir. Bulut sistemler kullanıcılarının kaynak taleplerine karşılık verirken, kullanıcı tarafından talep edilen kaynak miktarındaki azalma durumunda ise kaynakların otomatik olarak azaltılmasına olanak sağlar.
- **Güvenlik:** Bulut sistemler verilerin veya hizmetlerin yedeklenmesi, platformun stabil çalışması için gerekli güvenlik önlemlerini alırlar. Bu durum veri kayıplarının önlenmesine ve kullanıcıların almış oldukları hizmetin kesintisiz bir şekilde verilmesine imkân sağlar.

Bulut sistemler hizmet türüne ve kullanım biçimine göre sınıflandırılması Şekil 2.2’de gösterilmiştir. Bulut bilişim alt yapısının dağıtılma biçimi ve kullanıcılarına sunulma şekline göre 3 sınıfa ayrılırlar. Kamusal bulut (Public Cloud) olarak isimlendirilen bulut türünde bulut hizmetleri internet üzerinden kullanıcılarına sunulur. Bu tür bulut sistemler düşük maliyetli olup; geniş bir kullanıcı yelpazesine sahiptirler. Kamusal bulutun sunmuş olduğu maliyet avantajı, esneklik ve ölçeklenebilirlik çok sayıda sektördeki kullanıcıların ilgisini çekmektedir. Özel bulut (Private Cloud) olarak adlandırılan bulut türleri ise sadece bir kullanıcının kullanımına sunulan bulut türlerini ifade eder. Bu tür bulut türlerinin en önemli avantajları kamusal bulut türlerine göre daha yüksek güvenli olmaları ve veri gizliliğinin sağlamalarıdır. Ayrıca kullanıcı alt yapı üzerinde tam yetkilidir. Kamusal ve özel bulut ortamlarının birlikte kullanıldığı hibrit bulut sistemleri ise bu iki tür bulutun avantajlarının bir arada kullanılmasına olanak sağlar. Topluluk Bulut (Community Cloud) ise aynı alanda veya benzer alt yapı gereksinimlerine ihtiyaç duyan kuruluşlar tarafından paylaşılan bulut türüdür. Topluluk bulutu genel bulutun esnekliğini özel bulutun güvenlik avantajlarıyla birleştirir ve belirli bir grubun ihtiyaçlarına cevap verir. Bu durum kullanıcının kendi ihtiyacına göre bulut ortamını düzenlemesine imkân sağlar.



Şekil 2.2. Bulut türlerinin sınıflandırılması

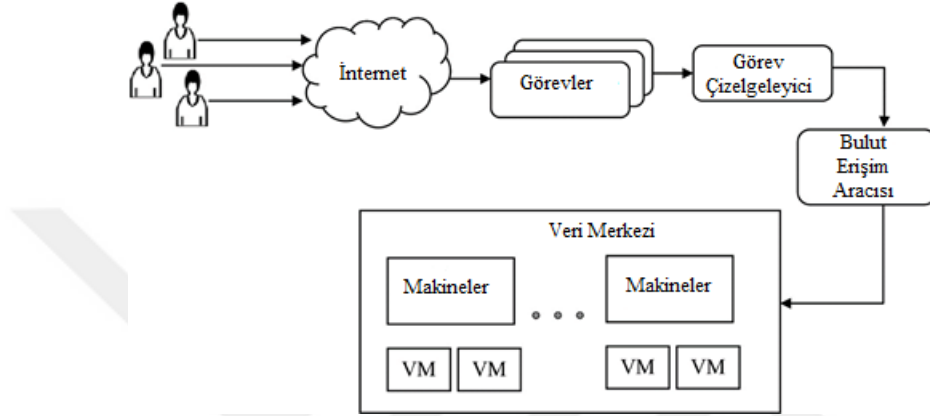
Bulut sistemler kullanıcıların ihtiyaçlarına göre sunmuş oldukları alt yapı ve yazılım hizmetleri farklılık göstermektedir. Temel hesaplama kaynaklarının sağlandığı bulut modeli olan altyapı hizmeti IaaS kullanıcılarını fiziksel donanımı yönetme gibi yüklerden kurtarır. Kullanıcıların ihtiyaç duydukları donanımları kiralayıp kullanmasına olanak sağlar. Bu model bulut sistemlerde kullanıcı sanal makineler, ağ ayarları ve diğer cihazların yönetiminde tam kontrole sahiptir. Google Bulut Platformu (Google Cloud Platform, GCP), Microsoft Azure, Amazon Web Hizmeti(Amazon Web Services, AWS) bu model bulut türleri olarak öne çıkmaktadır. Hizmet Olarak Platform (Platform as a Service, PaaS) bulut modeli ise yazılım geliştirme, test etme vb. gibi süreçler için gerekli olan tüm alt yapı ve araçları bir arada kullanıcılarına sunar. Bu modelde altyapı ile ilgili yönetimsel işlemler bulut sağlayıcı tarafından sağlanırken; kullanıcılarına uygulamalarını oluşturma ve yürütme için imkânlar sunar. Heroku, Google App Engine, Microsoft Azure App Service bu model bulut türlerine örnek olarak gösterilebilir. Hizmet Olarak Yazılım (Software as a Service, SaaS) bulut modeli kullanıcıların internet üzerinden yazılımlara erişmesine olanak sağlar. Bu modelde yazılımlar bulut üzerinde bulunur ve kullanıcılar internet aracılığıyla bu uygulamalara erişip kullanabilmektedir. Bu tür bulut modelinde yazılımların yüklenmesi, yazılımların güncelleme ve bakımlarının yapılması bulut hizmet sağlayıcısı tarafından yapılır. Google Workspace, Microsoft 365, Salesforce, Zoom bu tür bulut modellerine örnek olarak gösterilebilir.

2.2. Çizelgeleme Problemleri

Görev çizelgeleme bulutta bulunan kaynakların verimli kullanılması için kullanıcılardan gelen görev isteklerinin en uygun kaynağa atanmasını amaçlar [82]. Yapısı gereği dinamik olan bulut sistemlerin çok sayıda kullanıcıya hizmet vermesi görev çizelgeleme problemini daha zorlaştırmaktadır. Bulut sistemlerin alt yapılarında bulunan veri işleme veya depolama cihazları birbirinden farklı olabilmektedir. Büyük veri merkezlerinde bu cihazların sayısı yüzbinleri bulabilmektedir [83]. Google bulutta bir milyondan daha fazla sunucu olduğu; Microsoft'un Chicago'da veri merkezinden 300.000 den daha fazla sunucu olduğu tahmin edilmektedir. Görev çizelgeleme heterojen yapıya sahip bulut sistemlerde farklı kullanıcıların taleplerini en uygun kaynağa atama işlemidir. Görev çizelgeleme

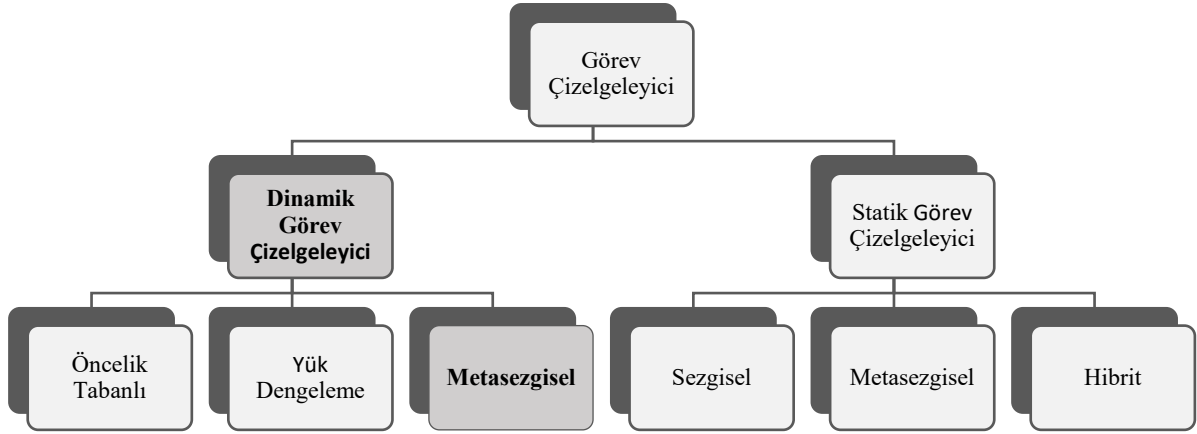
yapılırken bulut ortamında bulunan kaynakların maksimum oranda kullanılması, kullanıcıların taleplerine minimum sürede cevap verme, veri merkezinde harcanan enerji miktarını minimize etmek, kullanıcının bulut sağlayıcıya ödeyeceği hizmet bedeli veya bulut sağlayıcının işletim giderlerinin minimize edilmesi gibi çeşitli amaçlar hedeflenebilmektedir.

Bulut sistemlerde görev çizelgeleme aşamaları Şekil 2.3'te gösterilmiştir. Görev çizelgeleyici kullanıcılardan gelen görevleri bulut sisteminde aktif olan donanım arasındaki en uygun eşleştirmeyi yapar ve sonuçları bulut erişim aracısına gönderir. Bulut erişim aracısı çizelgeleyici tarafından gönderilen eşleştirmeleri dikkate alarak görev ve kaynaklar arasındaki ilişkiyi kurar.



Şekil 2.3. Bulut sistemlerde görev çizelgeleme aşamaları

Literatürde bulut sistemler için birçok görev çizelgeleme algoritması kullanılmıştır. Bu yöntemler Şekil 2.4'te gösterildiği gibi kategorize edilebilir. Dinamik görev çizelgelemede görevler sistemin mevcut durumuna göre atanır. Bulut ortamında kullanıcı talepleri ve mevcut kaynak durumu değişiklik göstermektedir. Dinamik görev çizelgeleme bulut sistemlerde yaygın bir şekilde kullanılmaktadır. Statik görev çizelgeleme de görevler kaynaklara en başta atanır ve bu atama daha sonra değiştirilemez. Bu görev çizelgeleme önceden kullanıcı talepleri ve mevcut kaynak durumu biliniyorsa kullanılır. Statik görev çizelgelemenin uygulaması dinamik çizelgelemeye göre daha basittir fakat değişen koşullara adapte olamaz. Dinamik görev çizelgeleme, bulut sistemlerde görevlerin anlık duruma göre esnek bir şekilde düzenlenmesini sağladığı için pek çok avantajı vardır. Dinamik çizelgeleme sistemin değişen farklı yük koşullarına adapte olmasına imkân verir. Görevler ve mevcut kaynaklar arasındaki eşleştirme en uygun şekilde gerçekleştirilir. Bu durum bulut sağlayıcılarının işletim giderlerinin azaltılmasına imkân verir. Ayrıca öncelikli görevlerin hızlı bir şekilde icra edilmesi, bulut ortamındaki iş yükünün dengeli bir şekilde dağıtılması gibi avantajları vardır. Bu çalışmada önerilen yöntemler dinamik görev çizelgeleyicidir.



Şekil 2.4. Bulut sistemlerdeki çizelgeleyici türleri

2.3. Optimizasyon

Optimizasyon kelimesi, soyut olarak bir kararın verilmesi ya da somut anlamda bir sistemin veya sistem parçalarının tasarımlarının yapılması süreçlerinde, elde edilecek somut ürünün mümkün olduğunca mükemmel olabilmesi için yapılan eylem, izlenen süreç veya yöntem anlamına gelmektedir [84]. Optimizasyondan kasıt bazı problemler için minimumu, bazı problemler için ise maksimum durumu ifade edebilir. Bu işlemi gerçekleştirmek için temel matematiksel yöntemler kullanılmaktadır. Bu yöntemler analitik, sayısal veya deneysel yöntemler içerebilir.

Mühendislik problemlerinin süreçlerinde problemin tespit edilmesi, problemin modellenmesi ve çözüm için uygun bir yöntemin seçilmesi gerekmektedir. Optimizasyon problemlerinde genellikle bir veya daha fazla amaç fonksiyonu çeşitli kısıtlar altında gerçekleştirilir. Standart bir optimizasyon problemini Denklem 2.1 'de gösterildiği gibi ifade edilebilir [84].

$$x \in R^n \min_{u(x) \in R^m} f(x, (u(x))) \quad (2.1)$$

Kısıt:

$$h(x, u(x)) = 0$$

$$g(x, u(x)) \leq 0$$

$$x_L \leq x \leq x_U$$

$$G(x, u(x)) = 0$$

Denklemde x tasarım yöneyini, $u(x)$ hal denklemlerini $f(x, (u(x)))$ amaç fonksiyonunu $h(x, u(x))$ eşitlik kısıtlarını $g(x, u(x))$ eşitsizlik kısıtlarını x_L problemdeki değişkenlerin alt sınırlarını x_U problemdeki değişkenlerin üst sınırlarını $G(x, u(x))$ hal denklemlerini içeren diğer eşitlik kısıtlarını temsil etmektedir. Optimizasyon problemin çözülmesi amacıyla en uygun tasarımın bulunmasıdır. Bu işlemler gerçekleştirilirken öncelikle bir değerlendirme ölçütü (amaç fonksiyonu) ve çeşitli kısıtların durumu dikkate alınmalıdır.

Amaç fonksiyonu optimizasyon probleminin sürecini belirleyen en önemli faktörlerden biridir. Optimizasyon sürecinde birçok çözüm elde edilebilir. Bu çözümlerden hangisinin daha iyi olduğunu belirlemek için nesnel kriterler gerekir. Çözümler arasında ayırt ediciliği sağlayan bu nesnel ölçütlere amaç fonksiyonu denir. Amaç fonksiyonu probleme göre tanımlanır. Optimizasyon sürecinde amaç fonksiyonu probleme bağlı olarak minimize veya maksimize edilmeye çalışılır.

Optimizasyon sürecinde sağlanması gereken koşullar kısıt olarak ifade edilir [84]. Başarılı bir optimizasyon sonucu eğer kısıtlara uymuyorsa; bu çözüm elde edilen çözümler arasına dahil edilmez. Bu kısıtlar optimizasyon sürecinde bir parametrenin alabileceği alt ve üst sınırlar olabileceği gibi diğer parametreler arasında sağlanması gereken bazı koşullar da olabilir. Optimizasyon sürecinde amaç fonksiyonunun farklı noktalarda değerleri olabilir. Amaç fonksiyonunun değeri belirli bir x^* değerinin çevresinde $f(x^*) < f(x)$ şartını sağlıyorsa x^* yerel minimum denir. Eğer amaç fonksiyonunun değeri belli bir x^* değerinin çevresinde $f(x^*) > f(x)$ şartını sağlıyorsa x^* yerel maksimum olarak ifade edilir. Eğer amaç fonksiyonunun değeri sadece belli bir bölgede değil de tüm x değerleri için $f(x^*) < f(x)$ x^* global minimum denir. Benzer bir şekilde amaç fonksiyonunun tüm x değerleri için $f(x^*) > f(x)$ şartını sağlıyorsa x^* global maksimum adı verilir.

Optimizasyon sürecinde, hedefe göre belirlenen yerel minimum veya yerel maksimum nokta, yerel optimal nokta olarak adlandırılır [84]. Benzer şekilde, genel optimum değer de amaca göre genel minimum veya genel maksimum noktası olabilir. Hem yerel hem de genel optimal noktalarda, amaç fonksiyonunun türevi sıfırdır, çünkü bu noktada fonksiyonun değeri azalma eğiliminden artma eğilimine veya artma eğiliminden azalma eğilimine geçer ve geçiş noktasında eğim sıfırlanır. Bu nedenle, türevin sıfır olduğu noktadan önce ve sonra türev değeri genellikle farklı işaret taşır. Dolayısıyla $x = x^*$ noktasında $f'(x^*) = 0$ eşitliğini yazmak mümkündür.

Optimizasyon teknikleri optimize edilecek amaç sayısına göre üç sınıfa ayrılmaktadır. Eğer optimizasyon sürecinde sadece bir amaç optimize ediliyorsa tek amaçlı optimizasyon olarak ifade edilir. İki veya daha fazla amacın aynı anda optimizasyonu çok amaçlı optimizasyon sınıfına girmektedir.

2.3.1. Tek Amaçlı Optimizasyon

Bir sistemin veya problemin bir amacının en iyi halinin gerçekleştirilmesinin hedeflendiği optimizasyon yöntemidir [84]. Probleme bağlı olarak amacın minimize edilmesi veya maksimize edilmesi hedeflenir. Bu optimizasyon türünde amacın durumunu belirlemek için bir amaç fonksiyonu kullanılır. Bu tür yöntemlerde tek bir amaç optimize edildiği için uygulanabilirlik açısından kolaydır ve hızlı sonuçlar üretirler. Bu yöntemler birden çok amacın optimize edilmesi problemlerinde etkili sonuçlar üretmekte iyi değildir. Bu tarz problemlerin genel ifadesi Denklem 2.2’de gösterilmiştir [84][84].

$$\min_{x \in X} f(x) \text{ veya } \max_{x \in X} f(x) \quad (2.2)$$

Burada $f(x)$ amaç fonksiyonunu temsil eder ve minimize veya maksimize edilmek istenen değeri verir. Karar değişkenlerinin tanımlı olduğu çözüm uzayı x 'tir (genellikle kısıtlarla tanımlanır). Eğer problemin kısıtları yoksa Denklem 2.3'te gösterildiği gibi; eğer varsa bu durumda Denklem 2.4'te yazılabilir [84].

$$\min_{x \in X} f(x) \quad (2.3)$$

Kısıtlar:

$$g_i(x) \leq 0, i = 1, \dots, m \quad (2.4)$$

$$h_j(x) = 0, j = 1, \dots, n$$

Denklem 2.4'te $g_i(x)$ eşitsizlik kısıtları olup; $h_j(x)$ eşitlik kısıtlarıdır.

2.3.2. Çok Amaçlı Optimizasyon

Bazı optimizasyon problemlerinde birden fazla amaç hedeflenebilir. Çok amaçlı optimizasyonda, tek amaçlı optimizasyon yöntemlerinden farklı olarak aynı anda birden fazla çelişen amaç dikkate alınır [85]. Gerçek dünya problemlerinde genellikle birden fazla hedef vardır ve bu hedeflerin tümünü aynı anda en iyi duruma getirmek her zaman mümkün olmaz. Bu tarz problemlerde farklı amaçları aynı çatı altında toplayarak genel tek bir amaç fonksiyonu oluşturabileceği, pareto tabanlı yöntemler ve değişken tabu yöntemleri yaygın olarak kullanılır. Çok amaçlı optimizasyon problemine ait genel yaklaşım Denklem 2.5'de gösterilmiştir [84].

$$\min\{f_1(x)f_2(x) \dots f_i(x) \dots f_n(x)\} \quad (2.5)$$

Optimizasyon sürecinde bazı amaçlarda iyileşme görülürken bazı amaçların değerlerinde gerilemeler gerçekleşebilir. Tüm amaçları kapsayacak genel bir amaç fonksiyonu Denklem 2.6'da gösterilmiştir [84].

$$\min F(x), F(x) = \sum_{i=1}^n w e_i f_i(x) \quad (2.6)$$

$w e_i$ ifadesi amacın ağırlık katsayısı olup; genellikle tüm amaçların ağırlık katsayıları arasındaki ilişki Denklem 2.7'de gösterildiği gibidir [84].

$$\sum_{i=1}^n w e_i = 1 \quad (2.7)$$

Amaçlar arasındaki önem ilişkisi ağırlık katsayılarına göre belirlenir. En önemli amacın ağırlık katsayısı en yüksek en az öneme sahip amacın ağırlık katsayısı ise en düşük değere sahiptir. Amacın dikkate alınması için $w e_i > 0$ şartının sağlanması gerekmektedir.

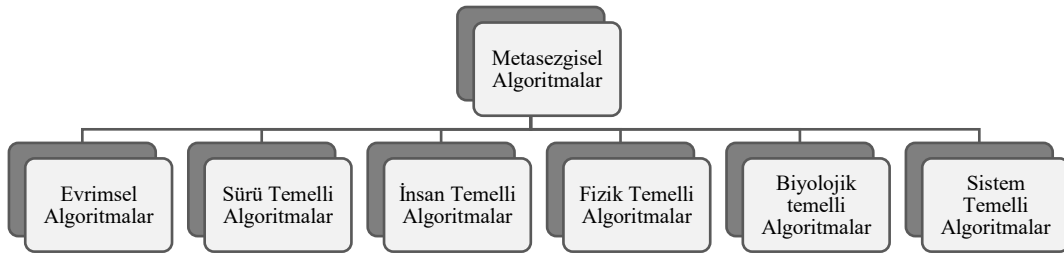
Pareto temelli yöntemlerde optimize edilecek amaçlar eşit öneme sahiptir ve ağırlık katsayıları eşittir. Bu yöntemde çözümler arasındaki değerlendirme dominasyon kontrolü ile sağlanır. Bir çözüm başka hiçbir çözüm tarafından domine edilmiyorsa, bu çözüm pareto optimal veya domine olmayan çözüm olarak kabul edilir. Dominasyon kontrolleri sonucunda oluşturulan çözümler kümesi, pareto sınırını oluşturur. Pareto sınırı, farklı çözümler arasında bir ödünleşim (trade-off) yaparak en iyi sonuçları temsil eder. Pareto temelli yöntemlerde genellikle tek bir optimum çözüm yerine bir dizi pareto

optimal çözüm elde edilir ve bu çözümler, hedef fonksiyonlar arasında bir denge sağlar. Bu tez çalışmasında önerilen çok amaçlı hibrit yöntem pareto temelli bir yaklaşımdır.

Dört veya daha fazla amaç fonksiyonunu optimize edilmesi birçok (many) amaçlı optimizasyon olarak isimlendirilir. Bu tür problemlerde, amaç fonksiyonları arasında denge kurmak ve çözümleri yönetmek daha karmaşık hale gelir. Birçok amaçlı optimizasyonda amaç sayısı arttığı için pareto ların karmaşıklığı artmaktadır. Çözümler arasındaki baskınlık ilişkisinin tespitinde karar vericinin sonuçlar arasında seçim yapmasını zorlaştırmaktadır. Amaç sayısı arttıkça hesaplama karmaşıklığı artmaktadır. Bu da çözüm süresinin uzaması ve hesaplama maliyetinin artması anlamına gelir.

2.4. Metasezgisel Yöntemler

Metasezgisel yöntemler karmaşık optimizasyon problemlerini çözmek için geliştirilmiş yöntemlerdir. Bu yöntemler özellikle çözüm alanının büyük olduğu karmaşık problemlerde başarılı olmaktadır. Mühendislik, endüstri vb. birçok alanda optimizasyon problemlerine başarılı çözümler getirmişlerdir. Metasezgisel yöntemler gradyan tabanlı yöntemlerden farklı olarak türev hesaplaması gerektirmezler [86]. Bu durum metasezgisel yöntemleri daha basit, esnek ve optimum noktalara yakalanmama noktasında avantaj sağlamaktadır. Metasezgisel yöntemler optimizasyon sürecinde rastgele çözümler üreterek arama sürecini başlatırlar. Bu durum yöntemlerin arama uzayını etkili kullanması ve erken yakınsamayı önler. Bu yöntemler en iyi çözümü garanti etmese de kabul edilebilir çözümlere hızlıca ulaşırlar. Metasezgisel yöntemlerin sınıflandırılması Şekil 2.5'te gösterilmiştir.



Şekil 2.5. Metasezgisel yöntemlerin sınıflandırılması

2.5. Bulut Sistemlerde Görev Çizelgeleme

Bulut sistemlerde görev çizelgelemede temel amaç, kuyruğa alınmış kullanıcı görevlerinin en uygun VM'e atanmasıdır. En uygun atamanın gerçekleşmesi, Sanal Makine Yöneticisi (Virtual Machine Manager, VMM) ve beraber çalıştığı görev planlayıcısının sorumluluğundadır. Görev kuyruğundaki kullanıcı görevleri birbirinden bağımsızdır ve farklı sayıda komut içerebilir. Her seferinde kuyruktan belirli sayıda görev alınır ve atanması planlanacak bu görevler, $T = \{t_1, t_2, \dots, t_i, \dots, t_n\}$ ile gösterilir. Burada, t_i i 'nci görevin içerdiği komut sayısını, n ise toplam görev sayısını temsil etmektedir.

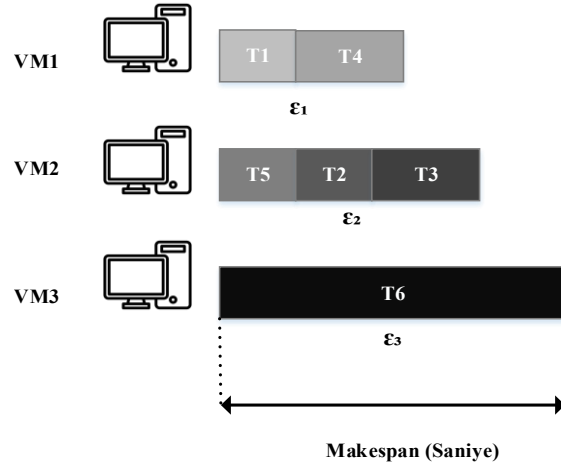
Görevlerin atanacağı sanal makineler farklı komut işleme kapasitelerine sahiptir ve sistemdeki toplam m adet aktif VM'ler $V = \{v_1, v_2, \dots, v_j, \dots, v_m\}$ ile ifade edilir. v_j , j 'nci sanal makinenin saniyede işlediği komut sayısını (Million Instructions Per Second, MIPS) temsil etmektedir. Buna göre, i 'nci görevin j 'nci sanal makinedeki yürütüm zamanı (ET_{ji}) Denklem 2.8 ile hesaplanır. Görev kuyruğundaki her görev farklı makinalara atanmak zorunda değildir. i 'nci görevin j 'nci sanal makineye atanıp atanmadığı bir mantıksal (Boolean) parametre olan e_{ji} ile ifade edilirse bu durumda j 'nci sanal makineye tahsis edilen tüm görevlerin toplam yürütüm süresi ($\mathcal{E}_j$) Denklem 2.9 ile bulunur.

$$ET_{ji} = \frac{t_i}{v_j} \quad (2.8)$$

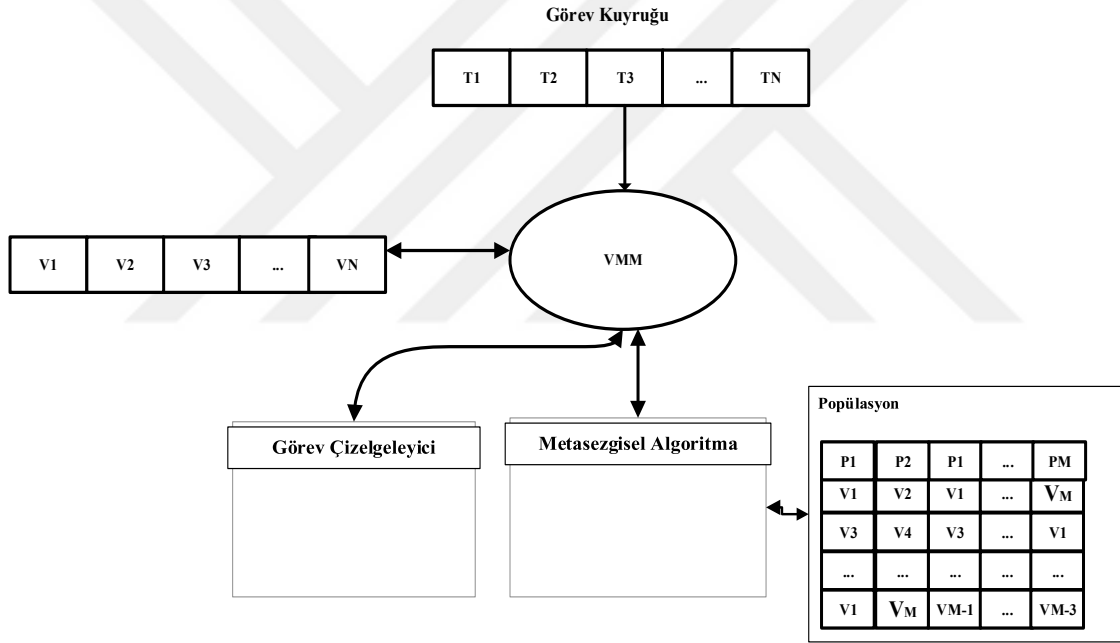
$$\mathcal{E}_j = \sum_{i=0}^N e_{ji} \times ET_{ji} \quad e_{ji} = \begin{cases} 1, & \text{eğer } i. \text{ görev } j. \text{ VM}'e \text{ atanmışsa} \\ 0, & \text{diğer durumlar} \end{cases} \quad (2.9)$$

Bulut sistemlerde görev çizelgeleme, makespan, maliyet ve enerji tüketimi gibi farklı amaçlar için gerçekleştirilebilir. Bulut sistemlerde makespan (MS), bir grup görevin tamamının bir grup sanal makinede en fazla ne kadar sürede yürütüleceğini gösterir ve Denklem 2.10 ile ifade edilir. Bu süreçlerin daha iyi anlaşılması için Şekil 2.6'da verilen temsili örnek incelenebilir. Bu örnekte, 6 farklı görevden oluşan bir görev grubunun 3 farklı sanal makineye nasıl atandığı ve bu atama planı için nasıl makespan süresinin bulunduğu görülmektedir. Metasezgisel görev çizelgelemede, başlangıçta genellikle rastgele görev dağılımı planlarına sahip belirli sayıda aday çözümler üretilir. Bu aday çözümler, kullanılan metasezgisel algoritmanın türüne ve kullandığı mekanizmalara bağlı olarak belirli sayıda iterasyon boyunca daha iyi çözüm önerileri sunmaya (exploration) ve komşu en iyi çözümünün önerdiği çizelgelemeye benzemeye (exploitation) çalışır. Aday çözümler önerdikleri çizelgelemeyi genellikle tabular formda sunarlar. Bu temsil formunda her bir sütun bir bağımsız görevi temsil ederken bunlara ait değerler ise atanacak olan sanal makineyi gösterir. İterasyonlardaki stokastik süreçler sonunda farklı bağımsız görevler ayrı VM'lere atanacağı gibi aynı VM'lere de atanabilir. Her aday çözümün başarımı, bir amaç fonksiyonu (örneğin makespan değeri gibi) ile ölçülür. İterasyonlar sonunda en iyi amaç fonksiyon değerine sahip olan çözümün görev çizelgelemesi kullanılır. Metasezgisel tabanlı görev çizelgeleme'in bu aşamaları Şekil. 2.7'deki gibi temsili olarak gösterilebilir.

$$MS = \max_{j=1}^m (\mathcal{E}_j) \quad (2.10)$$



Şekil 2.6. Makespan gösterimi



Şekil 2.7. Metasezgisel görev çizelgeleme

Makespan süresinin kısa olması kullanıcının almış olduğu hizmetin kalitesini arttırırken bu görevlerin bulut ortamında icra edilmesi esnasında tüketilen enerji miktarını arttırabilir. Bu durum bulut işletme maliyetlerini arttıracağı için dikkate alınması elzemdir. Herhangi bir görevin bir VM'de çalıştırılması için tüketilecek enerji miktarı Denklem 2.11'de gösterilen şekilde hesaplanır. Burada P_j , j'nin sanal makinenin bir komut işlemesi için harcadığı enerji miktarını göstermektedir. Böylece j'ninci sanal makinedeki toplam enerji tüketimi Denklem 2.12 ve ilgili görev çizelgeleme için toplam enerji tüketimi ise Denklem 2.13 ile bulunacaktır. Sonuç olarak bir görev çizelgeleme çözümü için bulunan makespan ve enerji metrikleri $O = [MS SE]$ vektörü ile temsil edilebilir.

$$E_{ji} = tc_i \times P_j \quad (2.11)$$

$$TE_j = \sum_{i=0}^N e_{ji} \times E_{ji} \quad e_{ji} = \begin{cases} 1, & \text{Eğer } i'ninci \text{ görev } j'nincisinal \text{ makineye atanmışsa} \\ 0, & \text{Diğerleri} \end{cases} \quad (2.12)$$

$$SE = \sum_{j=0}^M TE_j \quad (2.13)$$

2.6. Bulut Bilişimde Metasezgisel Tabanlı Görev Değiştirmenin Dezavantajları

Her ne kadar metasezgisel yaklaşımlar bulut sistemlerde görev çizelgeleme problemlerin çözümünde başarılı olsalar da bazı dezavantajlara da sahiptirler. Bu dezavantajların başında yürütüm zamanı gelir. İteratif random süreçler yürüten metasezgisel algoritmalar, arama uzayını keşfetmek ve optimum çözümler bulmak için zaman harcamak zorundadır. Özellikle çok sayıda görev ve kaynağa sahip büyük ölçekli bulut sistemler için bu yürütme süresi uzun olabilir. Gerçek zamanlı görev çizelgeleme, geleneksel metasezgisel algoritmaların sağlayabileceğinden daha hızlı yanıt süreleri gerektirebilir. Bu çalışmada önerilen yöntem, bu dezavantajı mümkün olduğu kadar minimize etmeye yöneliktir. Metasezgisel görev çizelgelemenin ikinci dezavantajı ise belirsiz bir yapıya sahip olmalarıdır. Metasezgisel algoritmalar doğası gereği stokastik olduklarından üretmiş oldukları çözümler her denemede değişiklik gösterebilir. Bu iki dezavantajın dışında, metasezgisel çözümlerin doğası gereği iki kronik dezavantajı daha vardır, parametre seçimi ve yerel optimal çözümlere yakınsama. Kullanılan metasezgisel yöntemin sahip olduğu parametrelerin doğru seçilip seçilmediği ile alakalı olup ayrıntılı bir parametre analizi gerektirir. Son dezavantaj ise yerel optimum çözümlere takılma problemidir ki bu da tüm metasezgisel yöntemlerin ortak sorunudur. Bu ise birbirinden bağımsız deneylerin gerçekleştirilmesi ve analizi ile değerlendirilebilir.

3. KURAL TABANLI AKILLI METASEZGİSEL GÖREV ÇİZELGELEME

Bu bölümde, önerilen IRMTS (Interpretable Rule-Based Metaheuristic Task Scheduling) tekniğinin temel aşamaları, alt mekanizmaları ve kullanılan metotlar açıklanacaktır.

3.1. Irmits'nin Temel Prensipleri

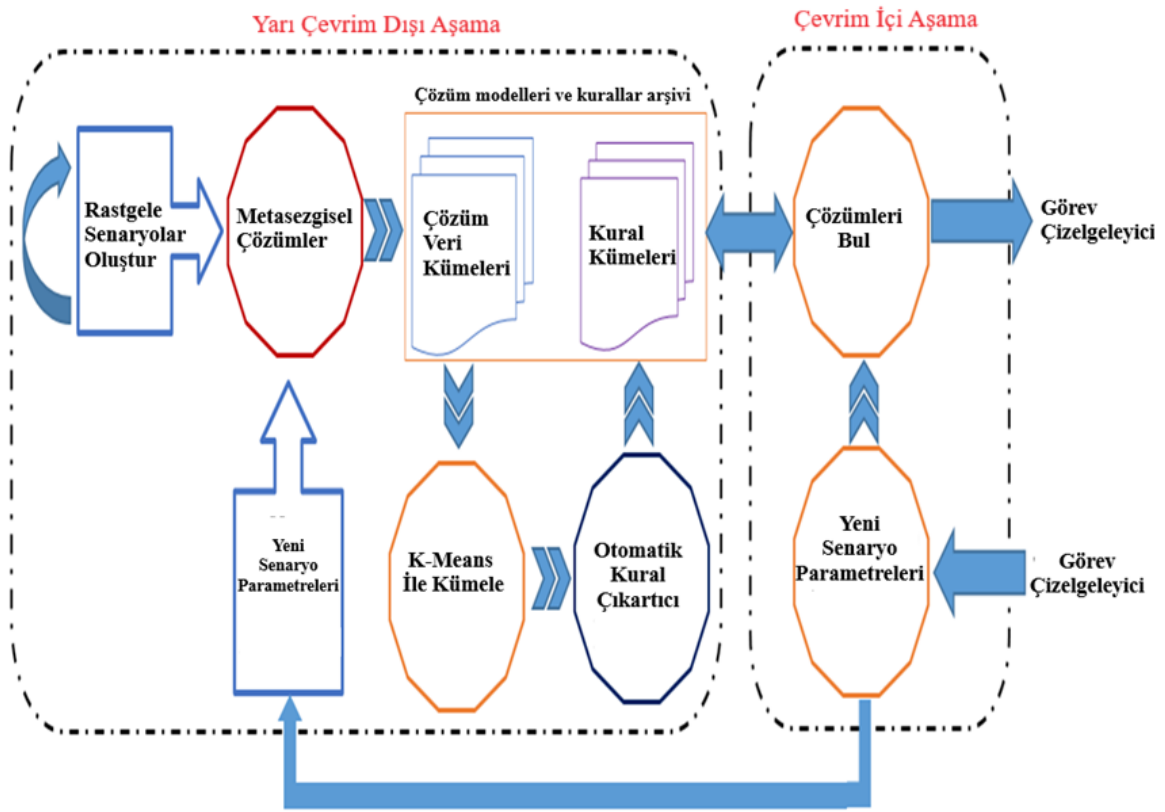
Tezin bu bölümünde bulut sistemlerde metasezgisel yaklaşımların sahip olduğu yürütüm zamanı dezavantajını büyük ölçüde gideren ve zaman kritik senaryolarda metasezgisel çözümlerin uygulanabilirliğe olanak tanıyan, kural tabanlı akıllı metasezgisel görev çizelgeleme (Interpretable Rule-Based Metaheuristic Task Scheduling, IRMTS) yöntemi tanıtılmıştır. Önerilen yöntemin temel prensibi, daha önce karşılaşılmış gerçek veya rastgele oluşturulmuş görev çizelgeleme senaryoları için bulunan metasezgisel çözümlerden modeller oluşturmak ve bu çözüm modelleri daha sonra karşılaşılan benzer problemlere uyarlamaktır. Bu yöntemde aşılması gereken başlıca zorluk ise elde edilen bu büyük çözüm havuzundan doğru modelin nasıl seçileceğidir.

Bilindiği kadarı ile literatürde ilk kez önerilen bu yöntem, birisi yarı çevrim dışı diğeri çevrim içi olan iki aşamadan oluşmaktadır (Şekil 3.1). Yarı çevrim dışı aşama arka arkaya gerçekleşen üç adımlı bir mekanizmaya sahiptir. Bu adımlar; çözüm uzayının (kural ve çözüm uzayı arşivi) oluşturulması, çözümlerin kümelenmesi (senaryo sınıflandırılması) ve otomatik kural çıkarımlarının yapılmasıdır. Yarı çevrim dışı aşamanın ilk adımı, bulut sistemde oluşabilecek olası görev çizelgeleme senaryolarını ve bu senaryolar için bulunan metasezgisel görev çizelgeleme çözümlerini içeren kapsamlı bir veri setinin (çözüm uzayı) oluşturulmasıdır. Bu adımda, çevrim dışı olarak önceden rastgele görev çizelgeleme problemleri oluşturulur ve bu problemler için PSO temelli metasezgisel çözümler üretilir. Rastgele üretilen problemlerin (senaryoların) istatistiksel parametreleri, görevlerin ve sanal makinelerin karakteristikleri ve bulunan metasezgisel atama planı bu veri setine eklenir. Bunun için iki ayrı veri seti (genel olarak çözüm veri kümesi olarak adlandırılırlar) kullanılmaktadır. Birincisi çevrim dışı görev çizelgeleme problemlerinin istatistiksel özelliklerini ve VM-görev eşleşmelerini tutarken diğeri ise her bir VM'in ve görevin karakteristiğini (MIPS, komut sayısı vb.) depolar. Doğal olarak, çözüm veri kümeleri basitten karmaşığa birçok farklı senaryoları içerecektir. Burada 3 adet veri türü vardır. VM'ler ve görevlere ilişkin istatistiksel sonuçları tutan, diğeri VM görev atamasını tutan diğeri VM'lere ait bilgileri tutan veri setidir.

Veri setindeki bu senaryolar, istatistiksel özellikleri bakımından bir birinden farklı olabileceği gibi benzer özellikler de gösterebilir. Yarı çevrim dışı aşamanın ikinci adımı bu senaryoların ve çözümlerinin gözetimsiz yöntemle kümelenmesidir. Bu kümelenme aslında tecrübe edilmiş senaryoların sınıflandırılmasını sağlamak için yapılmaktadır. Böylece, benzer senaryolar için kural setlerinin

indirgenmesi veya çözümlerin hızlı bulunabilmesi mümkün olabilmektedir. Bu adımdaki kümelendirme bir son teknoloji algoritma olan K-Means ile gerçekleştirilmektedir.

Yarı çevrim dışı safhanın bir diğer özgün aşamasında ise, her bir küme için oluşturulmuş çözüm kümelerinin bulunmasında yardımcı olacak yorumlanabilir kural setlerinin çıkarılması gerçekleştirilir. Veri setinin kapsamlı ve karmaşık olması insan gözü ile anlamlı kural çıkarımını mümkün kılmayacaktır. Bu nedenle makine öğrenmesi temelli otomatik kural çıkarım yöntemleri elzemdir. Literatürde sürekli veriler için kural çıkarımlarına olanak sağlayan metasezgisel yöntemler tercih edilebilir. Bu çalışmada, SFO temelli kural çıkarımı [87] bu problem için yeniden uyarlanmıştır. Her sınıf için çıkarılan yorumlanabilir kural setleri, çevrim içi safhada karşılaşılan senaryoya en yakın görev planlarının belirlenmesinde kullanılır. İlk safhanın “yarı çevrim dışı” olarak isimlendirilmesinin sebebi, çevrim dışı aşamada önceden öngörülmemiş bir senaryo ile karşılaşılması durumunda yeni bir metasezgisel çözümün arka planda bulunarak ilgili veri setine eklenebiliyor olmasındandır. Çevrim içi safhada, ilgili kural setlerini kullanarak en uygun çözümü bulma ve yeni problem parametrelerini yarı çevrim dışı safhaya iletme gerçekleşmektedir.



Şekil 3.1. IRMTS'nin iç adımı ve aşamaları

3.2. PSO Temelli Görev çizelgeleme

Yarı çevrim dışı aşamanın ilk adımı olan metasezgisel çözüm uzayının oluşturulması bir son teknoloji algoritma olan PSO ile gerçekleştirilmektedir. PSO 'da her bir aday çözüm parçacık olarak

adlandırılır ve bu parçacıklar popülasyonu oluşturur. Parçacıklar, iterasyonlar boyunca hem kendi en iyi değerini (P_{best}) hem de sürü en iyi değerini (G_{best}) dikkate alarak hareket ederler. Bu parçacık hareketleri Denklem 3.1-3.2 ile hesaplanır. Burada X_p ve V_p parçacığın pozisyon ve hız değerlerini verirken, $rand_1$ ve $rand_2$ rasgele katsayılarını göstermektedir. W parçacığın atalet katsayısıdır ve keşif ile sömürü (exploitation) arasındaki dengeyi belirler. c_1, c_2 ise parçacık hareketinin hızlanma katsayılarıdır. Kullanılan temsil formundaki vektörün her bir indisi bir görevi temsil ederken içindeki değer ise sanal makinenin tanımlama numarasını (Identification Number, ID) gösterir. Sanal makine ID numarasının tamsayı olmasından dolayı ondalıklı değerlerin Denklem 3.3 ile bulunan tamsayı karşılıkları dikkate alınır. Bu değerler aynı zamanda çizelgeleme vektörüdür (S_v). Bu çalışmada kullanılan ve görev çizelgeleme problemine uyarlanan PSO'nun sözde kodu Algoritma 3.1'de verilmiştir.

$$V_p = W \times V_p + c_1 \times rand_1 \times (P_{best,i} - X_i) + c_2 \times rand_2 \times (P_{gbest} - X_i) \quad (3.1)$$

$$X_p^{yeni} = X_p + V_p \quad (3.2)$$

$$S_v = TamsayıVektorDizisineYuvarla(X_p^{new}) \quad (3.3)$$

Eğer elde edilen S_v değerinin bileşenlerinden herhangi birisi alt ve üst sınır değerlerini aşıyorsa bu durumda ilgili bileşen alt ve üst sınır değerleri arasında rast gele üretilmiş bir tam sayı değeri alır.

Algoritma 3.1. Görev çizelgelemede kullanılan PSO'nun sözde kodu

-
1. Parametreleri oluştur
 2. Başlangıç popülasyonunu oluştur
 3. $Itr_{maks} \leftarrow$ Maksimum iterasyon
 4. $N_{pop} \leftarrow$ popülasyon boyutu
 5. **For** $i=1$ to N_{pop}
 6. $F_i \leftarrow P_i$ amaç değerini hesapla
 7. **Eğer** $F_i > P_{best}$
 8. $P_{best} = F_i$
 9. $G_{best} \leftarrow$ Tüm parçacıkların en iyi amaç değeri
 10. **For** $i=1$ to N_{pop}
 11. Denklem 3.1'e göre parçacıkların hızını hesapla
 12. Parçacıkların pozisyonunu Denklem 3.2'ye göre güncelle
 13. Parçacığın pozisyonunun tam sayı değerini al Denklem 3.3
 14. **Eğer** $(X_p < X_L)$ or $(X_p > X_U)$
 15. $X_p = YeniPozisyonDeğerleriAta(X_L, X_U)$
-

Çözüm uzayı veri setinin oluşturulmasında beklenen senaryoya ait istatistiksel veriler toplanır. Toplanan istatistiksel parametreler arasında PSO çözümünün makespan değeri, VM sayısı, görev sayısı, VM'lerin kapasitelerinin ve görevlerin komut sayılarının ayrı ayrı ortalaması, medyan ve standart sapma gibi değerleri bulunmaktadır. Bu istatistiki bilgiler sonraki aşamada karşılaşılabilecek olan senaryoların benzerliklerini/sınıflarını bulmak için kullanılmaktadır. Buna ek olarak her bir beklenen senaryo için bulunan PSO çözümündeki sanal makine başına her sanal makinenin o senaryo içindeki görev yükü de kaydedilmektedir. Bu görev yükleri aynı zamanda güncel senaryonun görev dağılım modelini de

göstermektedir. Bu modeller daha sonra karşılaşılabacak benzer görev çizelgeleme problemlerinin çözümünde kullanılacaktır.

Bu aşamalarının sonuçlarının daha iyi anlaşılabilmesi için Tablo 3.1 ve Tablo 3.2 incelenebilir. Bu tablolar, bu çalışma kapsamında yapılan, 20 VM içeren bağımsız bir deney sonucunda elde edilen kümelendirilmemiş örnek bir veri setini içeriğini göstermektedir. Daha rahat görülebilmesi için veri seti iki ayrı tablo şeklinde sunulmuştur. Tablo 3.1 üretilen 2000 senaryonun istatistiksel özelliklerini, Tablo 3.2 ise her senaryo için görev çizelgeleme modelini göstermektedir.



Tablo 3.1. Ham veri setinde beklenen yarı-çevrimdışı görevlendirme senaryoları

Deney	VM Sayısı	Ort. CPU	Ort. Görev	Makespan	Görev Medyan	CPU Medyan	...	CPU Mod	Görev Std.	CPU Std.
1	17	9 000.0	5 027.32	41.56	5 041.5	9 000.0	...	1 000	564.86	5 049.75
2	9	5 000.0	5 061.40	16.73	5 008.5	5 000.0	...	1 000	570.76	2 738.61
3	12	6 500.0	5 018.98	34.87	5 025.0	6 500.0	...	1 000	586.45	3 605.55
4	7	4 000.0	5 000.23	155.62	4 998.0	4 000.0	...	1 000	586.41	2 160.25
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
2 000	3	2 000.0	4 968.29	349.26	4 933.0	2 000.0	...	1 000	551.12	1 000.0

Tablo 3.2. Her senaryo için PSO tabanlı görev çizelgeleme çözümleri

Deney	VM_1	VM_2	...	...	VM_{20}	$\mathcal{E}_1$	$\mathcal{E}_2$	...	...	$\mathcal{E}_{20}$
1	T_2	T_4	...	...	T_{n-1}	40.10	37.5	...	...	41.56
2	T_1	T_2	...	...	T_{n-3}	10.25	16.73	...	...	14.20
3	T_5	T_1	...	...	T_n	34.87	30.24	...	...	33.55
4	T_n	T_8	...	...	T_1	155.5	155.62	...	...	150.7
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
2000	T_6	T_7	...	...	T_{n-1}	342.10	320.15	...	...	349.26

3.3. K-Means

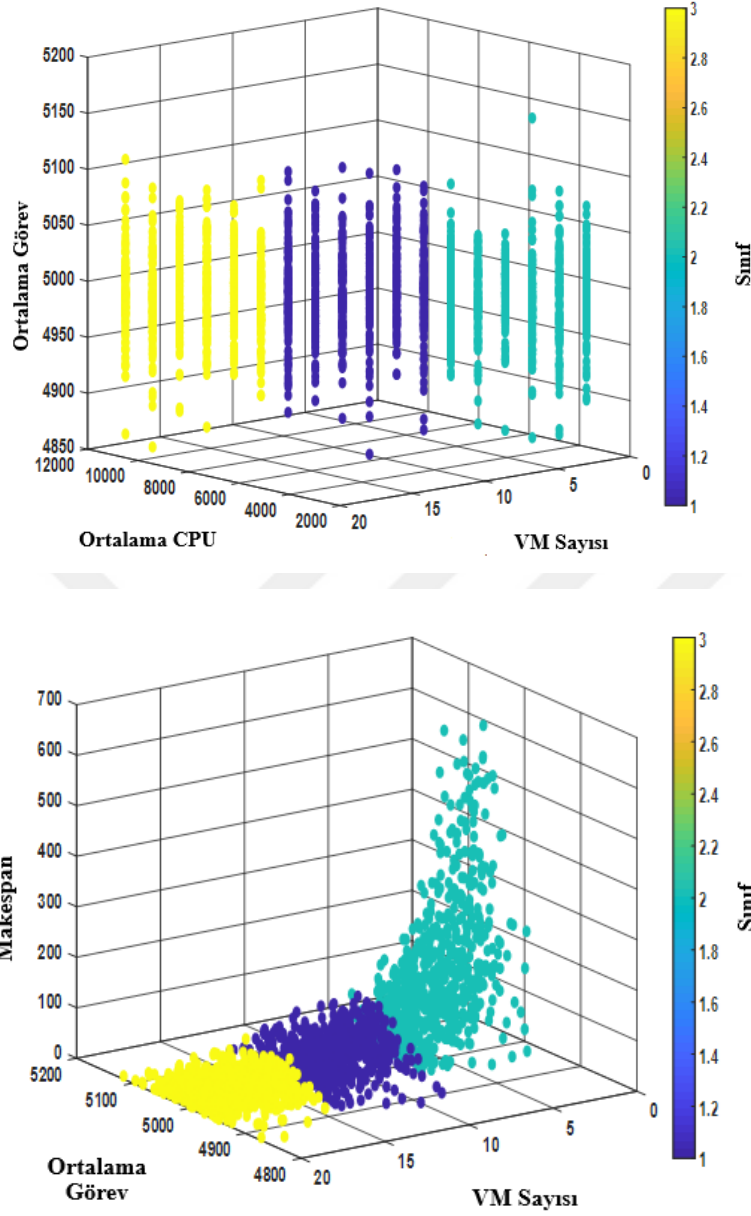
Önerilen yöntemde bulut görev planlayıcı, bir kural tabanlı gözetimli makine yöntemi ile eğitilmiştir. Eğitim süreci bir önceki adımda elde edilen ve içinde binlerce farklı senaryo ve PSO çözümü içeren veri seti ile yapılmaktadır. Bu kapsamlı veri seti, deneysel ve rastgele parametreler ile oluşturulduğu için benzer özelliklere sahip senaryolar içerebilir. Dahası pratikte görev çizelgeleme senaryoları görev sayısı, VM sayısı ve kapasitelerine göre düşükten yükseğe farklı yoğunluk kategorilerde değerlendirilebilir. Bu çalışmada hem görev çizelgeleyicisinin eğitilmesi hem de bulunan kurala göre atama sürecinin hızlı yapılabilmesi için elde edilen veri setinde anlamlı kümelendirme/etiketleme süreci gerçekleştirilmiştir. Bu amaçla, veri setindeki senaryolar için üç farklı etiket (daha sonra sınıf olarak kullanılacak) kullanılmıştır. Bu etiketler sırası ile görev sayılarının az ve sanal makinelerin yeterli olduğu LD (Düşük yoğunluk), nispeten daha fazla görevin ve yeterli sanal makinenin bulunduğu MD (Orta yoğunluk) ve görev sayısının fazla ve VM sayısının kısıtlı olduğu HD (Yüksek yoğunluk) sınıflarını temsil etmektedir. Bu etiket sayısı opsiyonel olup tasarımcı ve mühendis bakış açısına göre değişebilir. Kümelendirme literatürde iyi bilinen K-Means algoritması ile yapılmıştır. Sözde kodu Algoritma 3.2’de verilen K-Means algoritması, ön tanımlı küme sayısına ve elemanların bu küme merkezlerine olan uzaklıklarını dikkate alır. K-Means iterasyonları küme merkezlerinin stabil konumu sağlanıncaya kadar devam ettirilir. Tablo 3.3 bir önceki adımda elde edilen veri setinin kümelendirme sonrası son halini gösterirken Şekil 3.2 ise ortalama görev ve makespan gibi farklı parametrelere göre kümeleme sonrası çözüm uzayını göstermektedir.

Algoritma 3.2. Kümeleme için kullanılan K- Means algoritmasının sözde kodu

1.	Arşiv senaryo verilerini al
2.	İstenen küme sayısını belirt
3.	Veri alanında küme merkez noktasını rastgele seç
4.	Yakınsama kriterleri karşılanana kadar
5.	Her öğeyi en yakın merkez noktasına sahip kümeye ata
6.	Her küme için yeni ortalama hesapla

Tablo 3.3. Senaryoların K-Means ile sınıflandırılması

VM Sayısı	Ort. CPU	Ort. Görev	Makespan	Görev Medyan	CPU Medyan	...	CPU Mod	Görev Std. S.	CPU Std. S.	Sınıf (Etiket)
17	9 000.0	5 027.32	41.56	5 041.5	9 000.0	...	1 000	564.86	5 049.75	HD
9	5 000.0	5 061.40	16.73	5 008.5	5 000.0	...	2 000	570.76	2 738.61	LD
12	6 500.0	5 018.98	34.87	5 025.0	6 500.0	...	1 000	586.45	3 605.55	LD
7	4 000.0	5 000.23	155.62	4 998.0	4 000.0	...	3 000	586.41	2 160.25	MD
...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
3	2 000.0	4 968.29	349.26	4 933.0	2 000.0	...	2 000	551.12	1 000.0	MD



Şekil 3.2. Kümeleme sonrası çözüm alanı (Sarı: HD, Mor: LD, Yeşil: MD)

3.4. Ayçiçeği Optimizasyonuna Dayalı Kural Çıkarımı

Binlerce farklı senaryondan ve metasezgisel çözümlerden oluşan kümelendirilmiş veri seti daha sonraki görev çizelgeleme senaryoları için belirleyici ve yol gösterici olacaktır. Görev planlayıcısı bu kapsamlı veri seti ile eğitilerek anlaşılabilir/yorumlanabilir kurallar çıkarabilir ve yeni görev atamalarını bu kurallara göre yapabilir. Yorumlanabilir kural tabanlı makine öğrenmesinin seçilmesinin en önemli amacı, operatör ve tasarımcılar için kontrol edilebilir yönetim modelinin mümkün kılınmasıdır. Bu kadar kapsamlı bir veri setinden insan gözü ile anlamlı kural setleri çıkarmak imkânsızdır. Bu nedenle bu çalışmada otomatik olarak yorumlanabilir kural setleri çıkaran metasezgisel yöntemler tercih

edilmiştir. Metasezgisel yöntemler, sahip oldukları esneklik sayesinde otomatik kural çıkarım motorları olarak kullanılabilir. En önemli avantajları ise sürekli veri setlerinde herhangi bir ayrıklaştırmaya ihtiyaç duymadan anlamlı kurallar çıkarabilmeleridir. Metasezgisel kural çıkarım yönteminde kullanılacak algoritma için doğru temsil biçiminin kullanılması gerekmektedir.

Metasezgisel tabanlı kural çıkarımlarında genel olarak kullanılan temsil biçiminde her aday çözüm üç alt veri içermektedir. Buna göre S_i çözümü S_i^b, S_i^a ve S_i^u ($i=1, 2, \dots, n$) olmak üzere üç adet vektöre sahiptir ve bu vektörler her iterasyonda güncellenir. Bunlardan $S_i^b = (s_1^b, s_2^b, \dots, s_j^b, \dots, s_n^b)$, ilgili çözümde hangi niteliklerin kurala dahil edeceğini gösteren binary vektördür. Her aday çözümde başlangıç s_j^b , Denklem 3.4’de gösterildiği gibi bir $t(.)$ rastgele üreteç fonksiyonu ve ön tanımlı “ β ” eşik değeri ile belirlenir.

$$s_j = \begin{cases} 1 & (\text{eğer } s_j \text{ kuralda kullanılmışsa}) \\ 0 & \text{diğer} \end{cases} \quad \text{eğer } s_j^b > \beta, \beta \in [0,1] \quad (3.4)$$

$S_i^l = (s_1^l, s_2^l, \dots, s_j^l, \dots, s_n^l)$ ve $S_i^u = (s_1^u, s_2^u, \dots, s_j^u, \dots, s_n^u)$ ise adayın iterasyonlar boyunca bulunduğu/güncellediği alt ve üst limitleridir ($s_d^l, s_d^u \in R$). İterasyon süresince bir aday çözümün, “ j ” niteliği için bulunduğu alt-üst değerler, o niteliğin veri setindeki minimum (L^l) ve maksimum (U^u) değerlerini aşamaz. Bu nedenle “ $L^l \leq s_j^l < s_j^u \leq U^u$ ” koşulu her iterasyon adımı için geçerlidir. Bir iterasyonda $s_j^b > \beta$ şartını sağlarsa “ j ” niteliği “ S_i ” kuralına eklenir. Adayın iterasyonlardaki alt vektörlerinin değerleri aynı zamanda adayın oluşturduğu kuralı da ifade edecektir. Örneğin, “ A ” sınıfı için yapılan aramada, i ’ninci iterasyonda 2, 5 ve 7 niteliklerinin kurala dahil edildiğini varsayalım (s_2^b, s_5^b , ve $s_7^b > \beta$). Bu durumda veri seti nitelikleri ($D_1, D_2, \dots, D_n$) için adayın kural ifadesi Denklem 3.5’teki gibi olacaktır.

$$\text{eğer } s_2^l \leq D_2 \leq s_2^u \text{ ve } s_5^l \leq D_5 \leq s_5^u \text{ ve } s_7^l \leq D_7 \leq s_7^u = A \quad (3.5)$$

Bir aday çözümün çıkarmış olduğu kural, her iterasyonda, arama uzayındaki tüm veriler için kontrol edilir ve o aday için uygunluk değeri hesaplanır. Uygunluk değeri, bulunan kuralın doğruluk metriği ile belirlenir. S_i çözümünün doğruluk değeri, çıkarmış olduğu kuralın “if” ve “then” kısımlarının her bir veri için birlikte değerlendirmesi hesaplanır. Bu değerlendirmeler Tablo 3.4’e göre yapılır ve kural için doğru pozitif (TP), doğru negatif (TN), yanlış pozitif (FP) ve yanlış negatif (FN) değerleri elde edilir. Elde edilen bu değerler ve Denklem 3.6 ile adayın doğruluk değeri hesaplanır.

Tablo 3.4. TP, TN, FP ve FN değerlerinin hesaplanması

“if” kısmı	“then” kısmı	işlem
Doğru	Doğru	TP’yi 1 arttır
Yanlış	Yanlış	TN’yi 1 arttır
Doğru	Yanlış	FP’yi 1 arttır
Yanlış	Doğru	FN’yi 1 arttır

$$Doğruluk = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.6)$$

Bu çalışmada metasezgisel kural çıkarım algoritması olarak SFO kullanılmıştır. SFO, ayçiçeklerinin güneşe doğru hareketlerinden esinlenilmiş bitki zekası temelli optimizasyon algoritmasıdır [101]. SFO’da temel prensip, ayçiçeklerinin (aday çözümler) güneşe (iterasyondaki en iyi çözüm) doğru yönelmelerini sağlamaktır. Güneşin pozisyonuna göre komşu bireyler, kendi aralarında tozlaşır ve yeni bireyler üretir. Yeni oluşturulan bireyler popülasyonu büyüteceğinden performans kaybına yol açar. Bu nedenle belli oranda ayçiçekleri elenmek zorundadır. Tüm bu süreçler, sonlandırma kriteri sağlanana kadar iteratif şekilde devam eder. İterasyondaki güneş (X^*), diğer ayçiçeklerinin (X_i) yönelimleri belirlemede referans olur. n sayıda birey içeren popülasyonun güneşe yönelimi $\vec{O}_i$ Denklem 3.7 ile hesaplanırken diğer adayların hareket adımı (d_i) Denklem 3.8 ile bulunur. Burada λ atalet katsayısı olup $P_i(\|X_i + X_{i-1}\|)$ i’nci aday için tozlaşma olasılığıdır.

$$\vec{O}_i = \frac{X^* - X_i}{\|X^* - X_i\|} \quad i = 1, 2, \dots, n_p \quad (3.7)$$

$$d_i = \lambda \times P_i(\|X_i + X_{i-1}\|) \|X_i - X_{i-1}\| \quad (3.8)$$

Böylece, güneşe yakın bireylerin adımları diğerlerine göre daha küçük olacaktır. Bununla birlikte keşif safhasında adayların istenmeyen sıçramaların yapmaması için Denklem 3.9 ile hesaplanan maksimum adım değerini aşmamaları da gereklidir. Bu değer, ilgili niteliklerin üst (X_{max}) ve alt (X_{min}) değerlerine ve popülasyon sayısına (N_{pop}) bağlıdır. Son olarak adayın güneşe yönelimi Denklem 3.10 ile bulunur.

$$d_{max} = \frac{\|X_{max} - X_{min}\|}{2 \times N_{pop}} \quad (3.9)$$

$$\vec{X}_{i+1} = \vec{X}_i + d_i \times \vec{S}_i \quad (3.10)$$

Algoritma 3.3, SFO’nun kural çıkarım için uyarlanmış çalışma prensibini ve ilgili süreçlerin işleyişini göstermektedir. Kural çıkarım denetimli olduğu için iki aşamada gerçekleşmektedir; eğitim ve test. Bu nedenle veri seti eğitim ve test alt setlerine ayrılır. Eğitim safhasında kurallar oluşturulur ve en

başarılı uygun değerlerine sahip adayların kuralları test verileri ile denenir. Test verilerinde en yüksek başarıyı gösteren kurallar görev çizelgeleme safhasında kullanılır.

Algoritma 3.3. Uyarlanmış kural tabanlı SFO

-
1. Eğitim Veri Setini Al
 2. *Popülasyonu başlat (herbir S_i^b, S_i^l and S_i^u başlat)*
 3. $Itr_{maks} \leftarrow$ Maksimum Iterasyon
 4. $N_{pop} \leftarrow$ Popülasyon Boyutu
 5. $E \leftarrow$ Eliminasyon Oranı (%)
 6. **for** $i=1$ to N_{pop}
 7. $S_i =$ Ayçiçeği Üretme Fonksiyonu (S_i^b, S_i^a, S_i^u)
 8. Popülasyona Ekle(S_i)
 9. Varsayılan Değerleri Ata (TP, TN, FP, FN)
 10. Amaç Değerini Hesapla(populasyon)
 11. $S^* \leftarrow$ Güneşi Bul()
 12. **while**($i < Itr_{maks}$)
 13. Güneşe Dön()
 14. **for** $j=1$ 'den N_{pop}
 15. Bireysel Hesaplamalar ()
 16. Bireyleri Elemine Et (%E, i)
 17. Yeni Bireyleri Değerlendir (X^*)
 18. Varsayılan Değerleri Ata (TP, TN, FP, FN)
 19. Amaç Değerini Hesapla(populasyon)
 20. $S^* \leftarrow$ Güneşi Bul ()
 21. Test Veri Kümesi İçin En İyi Kuralları Test Et()
 22. En İyi Kuralları Göster()
-

Kural çıkarımları bu kümelenendirilmiş veri setleri üzerinden gerçekleştirilmiştir. SFO temelli kural çıkarımı denetimli bir yöntem olduğu için kümelenendirilmiş veri seti %80-%20 oranında eğitim ve test verisi olarak ayrılmıştır. Kural tabanlı SFO, her bir etiket/sınıf (LD, MD ve HD) için kural setleri elde etmiştir. Bunun için öncelikle eğitim veri seti üzerinden kurallar elde edilmiş ve daha sonra bu kurallar test veri seti üzerinde denenmiştir. Tablo 3.5 test veri seti deneylerinden elde edilmiş örnek kuralları göstermektedir. Bu kurallar, daha önceden denenmiş olası senaryoların hangi tip karakteristiğe sahip olduklarını göstermektedir. Örnek olarak; çevrim içi safhada bir görev çizelgeleme senaryosunun geldiği ve bu senaryonun parametrelerinin Tablo 3.5'teki 0.98 doğruluk değerine sahip ilk kuralla eşleştiği kabul edilsin. Bu da, mevcut senaryonun HD tipinde olduğu ve uygun çizelgeleme planının bu sınıf için daha önce elde edilmiş çözüm modellerinde aranması gerektiği anlamına gelir. Bundan sonraki aşama, IRMTS'nin mevcut HD çözüm modelleri içinde en yakın benzerliğe sahip çözümü bulmasıdır.

Tablo 3.5. Kural tabanlı SFO'dan elde edilen örnek kurallar

Sınıf	Doğruluk	Kural
HD	0.980	if((4995 <= AVGTask <=5087) and (4939 <= TaskMedyan <=5127) and (5348 <= TaskMode <=5998)):
MD	0.820	if((4979 <= AvgTask <=5048) and (5893 <= TaskMode <=5948) and (574 <= TaskSD <=610)):
LD	0.80	if((5 <= VmNumber <=5) and (3000 <= CPUMedyan <=3000) and (4632 <= TaskMode <=5579) and (577 <= TaskSD <=601)):
LD	0.750	if((3000 <= AVGCPU <=3000) and (5043 <= AVGTask <=5050) and (565 <= TaskSD <=597) and (1581 <= CPUSD <=1581)):
HD	0.745	if((5 <= VmNumber <=5) and (4992 <= AVGTask <=4994) and (4942 <= TaskMedyan <=5074) and (4140 <= TaskMode <=5031) and (547 <= TaskSD <=619) and (1581 <= CPUSD <=1581)):
HD	0.740	if((5 <= VmNumber <=5) and (5015 <= AVGTask <=5042) and (3000 <= CPUMedyan <=3000) and (4610 <= TaskMode <=5213) and (1000 <= CPUMode <=1000) and (586 <= TaskSD <=598) and (1581 <= CPUSD <=1581)):
HD	0.737	if((4813 <= TaskMode <=5567) and (580 <= TaskSD <=597)):
MD	0.730	if((10 <= VmNumber <=10) and (5500 <= AVGCPU <=5500) and (4923 <= AVGTask <=4970) and (550 <= TaskSD <=578)):

3.5. Çevrimiçi Aşama

Çevrimiçi safhada, yeni gelen görev çizelgeleme problemi için önceden hesaplanmış en iyi VM-görev modeli bulunur ve görev atama bu modele benzer şekilde gerçekleştirilir. Çevrim içi safhada gerçekleşen tüm adımlar, Algoritma 3.4'te verilen sözde kodda görülebilir. Bu safhada ilk olarak yeni gelen problemin görev ve VM karakteristiklerine en yakın kural ve bu kurala denk gelen sınıf bulunur. Bulunan sınıf içerisindeki tüm senaryolar ile mevcut senaryo arasında sanal makine sayısı açısından bir filtrasyon yapılır. Filtrasyon sonucunda elde edilen senaryolar ile mevcut senaryo arasındaki benzerlik oranının tespit edilmesi sonraki aşamada gerçekleştirilir. Hangi senaryonun modelinin seçimi iki şekilde belirlenebilir. Birincisi, eşleşen kurallardan en düşük makespan değerine sahip kural modellerinin kullanılmasıdır. İkincisi ise gelen senaryonun nitelikleri ile eşleşen kuralların nitelikleri arasındaki öklit mesafesinin Denklem 3.11 ile dikkate alınmasıdır. Bu teknikte en kısa öklit mesafesine sahip senaryonun modeli kullanılır. Öklit mesafesi aynı zamanda gelen senaryonun her hangi bir kurala uymaması durumunda en yakın modeli bulmada da kullanılmaktadır. Nadir de olsa her hangi bir kurala uymayan senaryonun gelmesi durumunda (benzerlik oranının çok düşük olması), yeni karakteristiğe en yakın çizelgeleme modeli kullanılırken aynı zamanda bu senaryo için arka planda tekrar birinci safha yürütülür ve kural seti yeni durum için güncellenir.

$$d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \quad (3.11)$$

En uygun senaryo belirlendikten sonra, atama modelinin oluşturulması, belirlenen modeldeki VM yük dağılım oranlarına göre (Denklem 2.9) gerçekleştirilir. Bunun için öncelikle dağıtımı yapılacak olan görevler komut sayılarına göre büyükten küçüğe sıralanır. En büyük komut sayısına sahip görev sırasıyla VM’lerde çalıştırılması durumunda oluşacak yükün ne olacağı Denklem 2.8 kullanılarak hesaplanır. Eğer VM’in mevcut yükü bu değerden büyük ise görev bu VM’e atanır, VM’in mevcut yük değeri güncellenir ve bir sonraki görev için işlemler tekrarlanır. Herhangi bir görevin tüm sunucuların mevcut yükünden fazla olması durumunda ise bu görev makespan değerinin minimize olacağı sanal sunucuya atanır. Bu durum nadir görülen bir durum olup mevcut modelin dışına çıkılması durumunda kullanılır.

Algoritma 3.4. Çevrim içi sahnenin sözde kodu

```

1.  T ← Yeni senaryonun parametrelerini al ( VM’ler,Gorevler)
2.  C ← Kurallardan yeni senaryonun sınıfını bul
3.  D ← Bulunan sınıfa ait tüm senaryoların verilerini al
4.  S ← Boyut(D)
5.  Ts=Boyut(Gorevler)
6.  Vs= Boyut(VM’ler)
7.  Sim ← ∞
8.  Index ← -1
9.  for i=1 to S
10.   P ← Öklit mesafesini Hesapla(Di, T)
11.   Eğer (P<Sim)
12.     Sim=P
13.     Index=i
14.  VmYuk ← Bulunan senaryoya göre VM’lerin yükünü hesapla
15.  for i=1 to Ts
16.   for j=1 to Vs
17.    Yuk=YukHesapla(Gorevler[i], VM[j])
18.    Eğer (VmYuk[j]>Yuk)
19.      i’ninci görevi j’ninci sanal makineye ata
20.    Eğer i’ninci görev bir sanal makineye atanmadıysa
21.      Makespan değerinin minimum olduğu sanal makineye i’ninci görev ata

```

3.6. Deney Sonuçları ve Tartışma

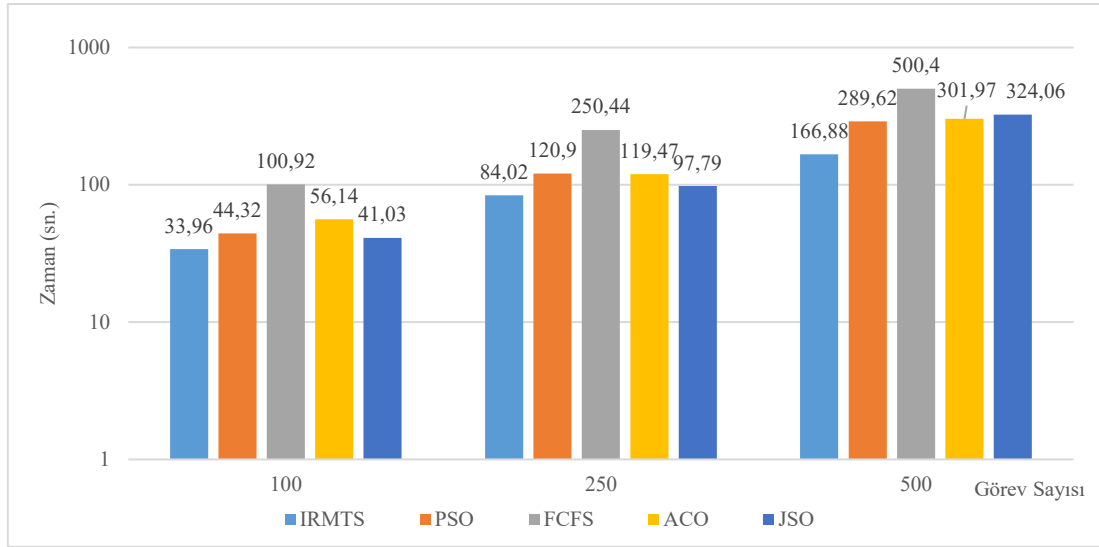
Önerilen IRMTS yönteminin farklı görev çizelgeleme senaryoları için uygulanabilirliği, başarılı bir bulut simülasyon ortamı olan CloudSim’de test edilmiştir. Deneylerde IRMTS’nin performansı, sezgisel FCFS algoritması, klasik ACO, PSO ve JSO algoritmalarının performansları ile karşılaştırmalı olarak sunulmuştur. Karşılaştırma için kullanılan bu üç klasik metasezgisel algoritma, iki nedenle deneylere eklenmiştir. Birincisi, stokastik süreçler yürüten bu belirsiz olmayan algoritmaların ürettiği anlık çözüm çeşitliliğini görmektir. İkincisi ise IRMTS’nin toplam yürütüm süresinin, klasik metasezgisel algoritmalarla göre ne kadar avantaj sağlayabildiğini ölçmektir. Deneyler, değişken VM ve görev sayılarına sahip farklı senaryolar için denenmiştir. Her deneyde, görevlerin içerdiği komut sayıları rast gele bir şekilde belirlenmiştir. VM’lerin MIPS kapasiteleri ise her senaryoda kullanılan VM sayısı ile orantılı bir şekilde belirlenmiştir. Kullanılan yöntemlerin performansları, makespan değerleri, ortalama yürütüm süreleri, CPU ve hafıza kullanımlarına göre değerlendirilmiştir. Simülasyonlar

Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz 2.90 GHz 8,00 GB RAM özellikli bilgisayarda yapılmıştır. Bu çalışmada çevrim dışı safhada veri setinin oluşturulması için 2.000 senaryo deneyi gerçekleştirilirken simülasyon ortamında her bir deney için elde edilen değerler toplamda 10 adet bağımsız deneyin ortalaması olacak şekilde alınmıştır. Yapılan deneylere ve metotlara ait genel parametreler Tablo 3.6’da sunulmuştur.

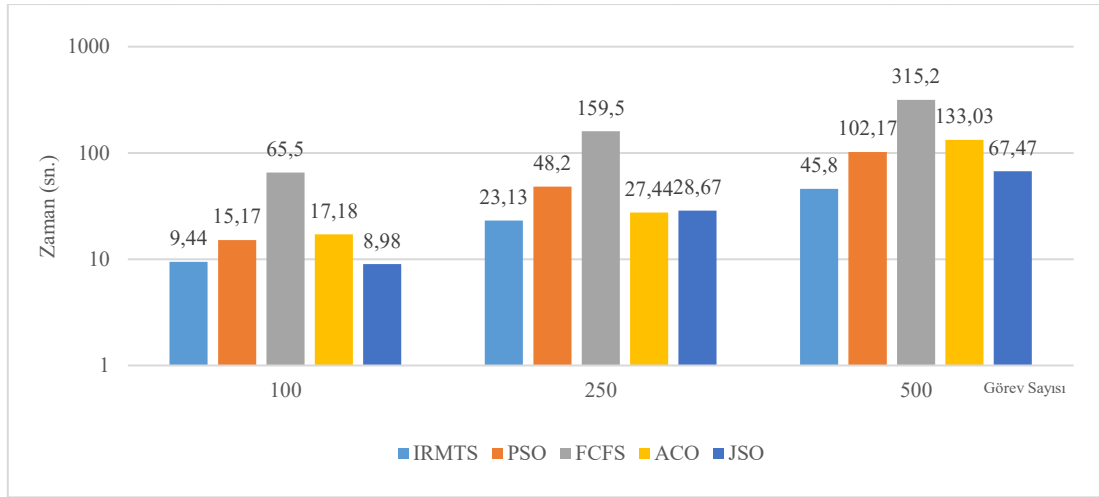
Tablo 3.6. Deney parametreleri

CloudSim	Görev Sayısı	100, 250, 500
	Görev Komut Sayısı	4.000-6.000
	VM Sayısı	5, 10, 20
	MIPS	1 000-20 000
PSO	c_1	1
	c_2	2
	w	0.5
	Maksimum İterasyon	100
K-Means	Küme Sayısı	3
SFO	Popülasyon Boyutu	20
	Tozlaşma	0.25
	Eliminasyon Oranı	0.20
	İterasyonlar	3x20
	Eğitim-Test Yüzdesi Dağılımı	%80-%20

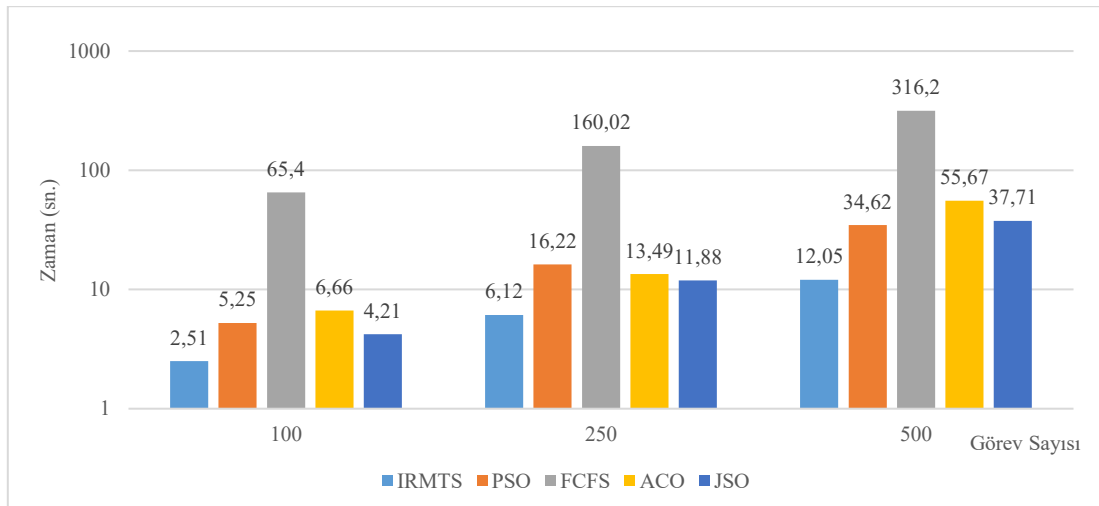
Deneylerde ilk performans değerlendirmesi makespan için yapılmıştır. Şekil 3.3, bağımsız deneylerden elde edilen ortalama makespan sonuçlarını, Tablo 3.7 ise bu deney sonuçlarının istatistiksel özetlerini göstermektedir. IRMTS tüm deneylerde en iyi makespan değerini elde etmeyi başarmıştır. Bunun en önemli nedeni, klasik metasezgisel algoritmalar, her yeni senaryo için değişken anlık çözümler bulurken, IRMTS en iyi çözüm modeline göre görev dağılımını yapmasıdır. Sezgisel FCFS, tüm deneylerde metasezgisel algoritmaların gerisinde kalmıştır. Ortalama deney sonuçlarına göre IRMTS, en iyi ikinci çözüm üreten JSO’ya göre yaklaşık %60 daha iyi makespan değeri elde ederken; JSO ise PSO’ya göre yaklaşık %10 başarılı olmuştur. Deneylerde, metasezgisel algoritmalar içinde makespan performansı en düşük olan ACO’dur. Buna rağmen tüm metasezgisel yöntemler sezgisel FCFS’den çok daha başarılı makespan değerleri elde etmeyi başarmıştır. Genel olarak FCFS, IRMTS’den yaklaşık 5 kat, diğer çözümlerden ise yaklaşık 3 kat daha yavaş çizelgeleme planlaması yapmıştır.



a) 5 VM'li senaryolar için ortalama makespan süresi sonuçları (Logaritmik ölçek)



b) 10 VM'li senaryolar için ortalama makespan süresi sonuçları (Logaritmik ölçek)



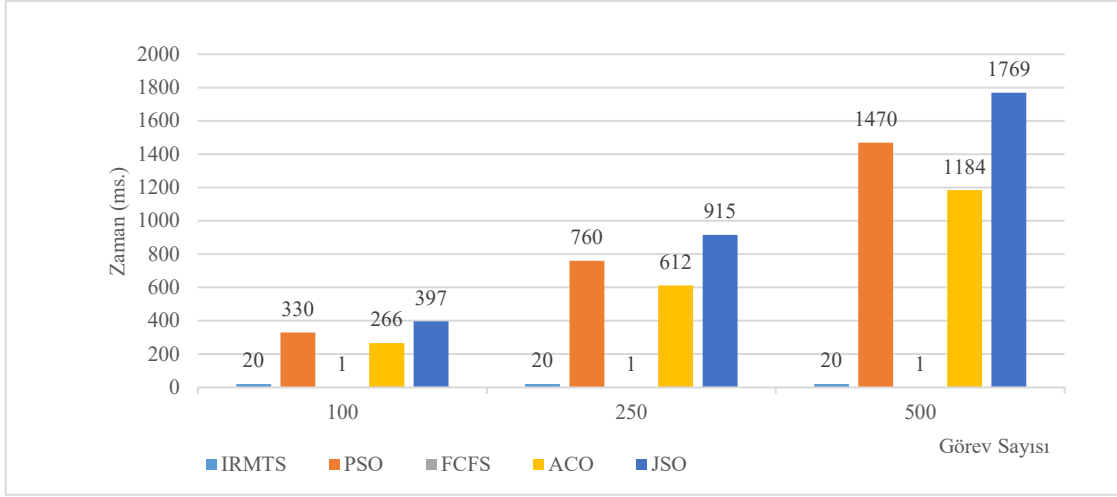
c) 20 VM'li senaryolar için ortalama makespan süresi sonuçları (Logaritmik ölçek)

Şekil 3.3. Kullanılan tüm yöntemlerin ortalama üretim süresi sonuçları

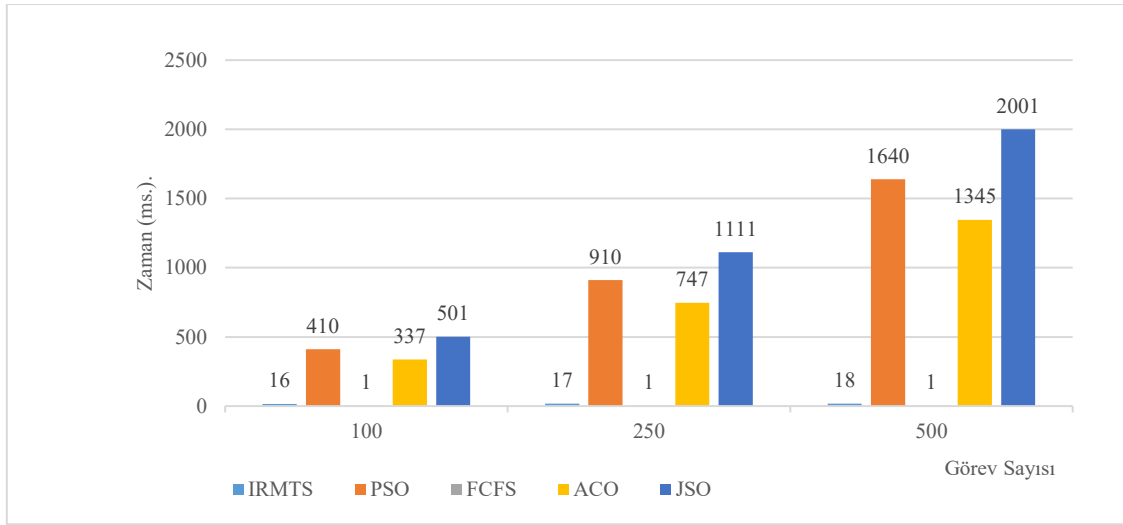
Tablo 3.7. IRMTS'nin makespan deneylerinin istatistiksel sonuçları

VM Sayısı	Görev Sayısı	Makespan (Saniye)			
		Ortalama	Medyan	Minimum	Maksimum
5	100	45.29	45.5	33.10	52.86
	250	131.0	134.0	83.57	160.75
	500	283.4	284.0	166.82	370.83
10	100	16.32	16.3	11.57	20.33
	250	47.75	47.4	29.2	66.57
	500	106.0	103.0	62.33	155.79
20	100	5.83	5.83	2.51	8.19
	250	17.53	17.78	5.98	25.35
	500	40.15	39.9	12.02	63.9

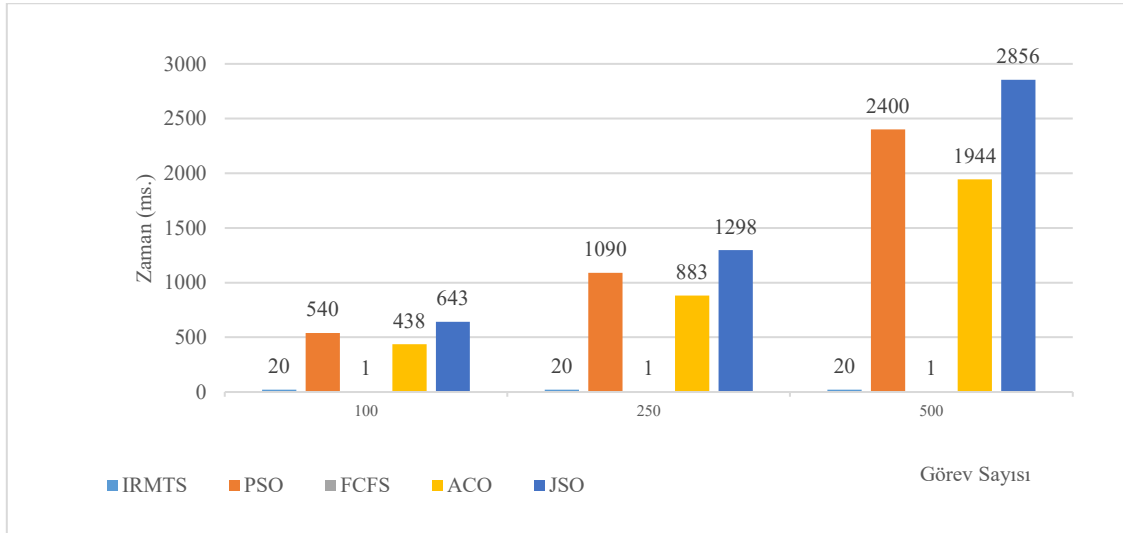
Önerilen yöntemin en önemli başarımı aslında karar verme sürecinde klasik metasezgisel algoritmalarla göre ne kadar zaman tasarrufu sağladığıdır. Bulut sistemler doğası gereği dinamik yapıdadır ve VM sayıları ve görev karakteristikleri değişken olabilmektedir. Klasik metasezgisel algoritmalar, her yeni senaryo için kendi iteratif stokastik süreçlerini tekrardan yürütmek zorundadır. Doğal olarak, bu da ciddi bir yürütüm zamanı maliyeti anlamına gelmektedir. Öte yandan IRMTS, otomatik çıkarmış olduğu kural setleri ve arşiv bilgileri ile en uygun çözüm modelini çok daha hızlı elde edebilmektedir. Şekil 3.4, farklı problemler için yöntemlerin harcamış olduğu ortalama karar verme sürelerini göstermektedir. IRMTS'nin klasik metasezgisel algoritmalarla göre sağlamış olduğu ortalama zaman kazancı bariz bir şekilde görülebilmektedir. Özellikle görev sayısı artarken diğer metasezgisel yöntemlerin IRMTS'nin karar verme sürecinin neredeyse sabit kaldığına dikkat ediniz. Tüm senaryolar dikkate alındığında, IRMTS, görev çizelgeleme çözüme, en yakın metasezgisel yönteme göre, ortalama olarak en az 17, en fazla 20 kat daha hızlı ulaşabilmiştir. Her ne kadar en kötü makespan performansını FCFS göstermiş ise de doğası gereği en hızlı atama planı yine FCFS'e aittir. Algoritmik süreci gereği bu beklenen bir durumdur ve atama süreci 1ms'nin altında gerçekleşmiştir. Belirsiz yapılarından dolayı metasezgisel algoritmaların makespan performansları bir birlerine ve IRMTS'e benzerlik gösterebilir. Ancak karar verme için toplam yürütüm süresi açısından hiçbir metasezgisel algoritma IRMTS'e yetişememektedir. Bu deney sonuçları IRMTS'nin özgün yönünün başarımını net bir şekilde ispatlamaktadır.



a) 5 VM'li senaryolar için ortalama karar süresi



b) 10 VM'li senaryolar için ortalama karar süresi

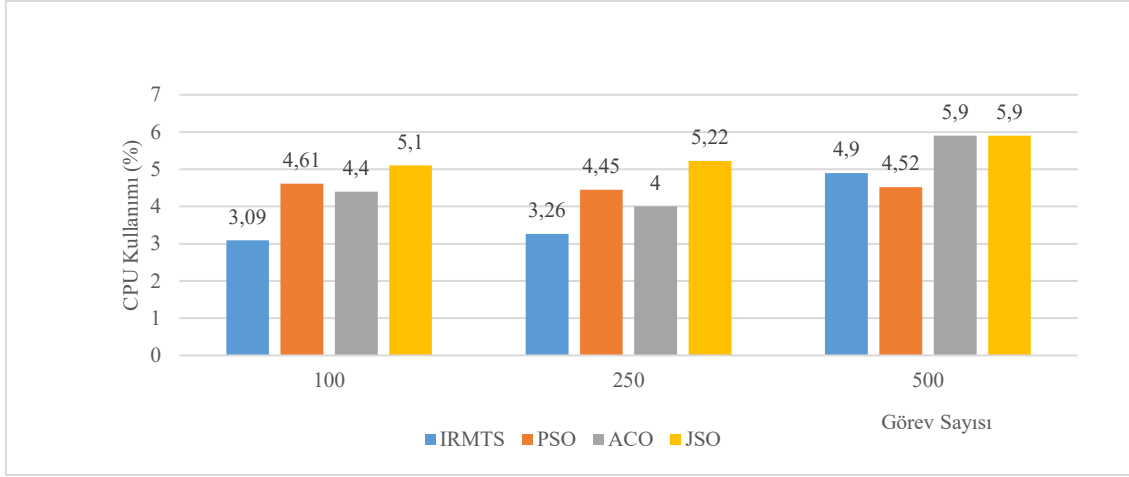


c) 20 VM'li senaryolar için ortalama karar süresi

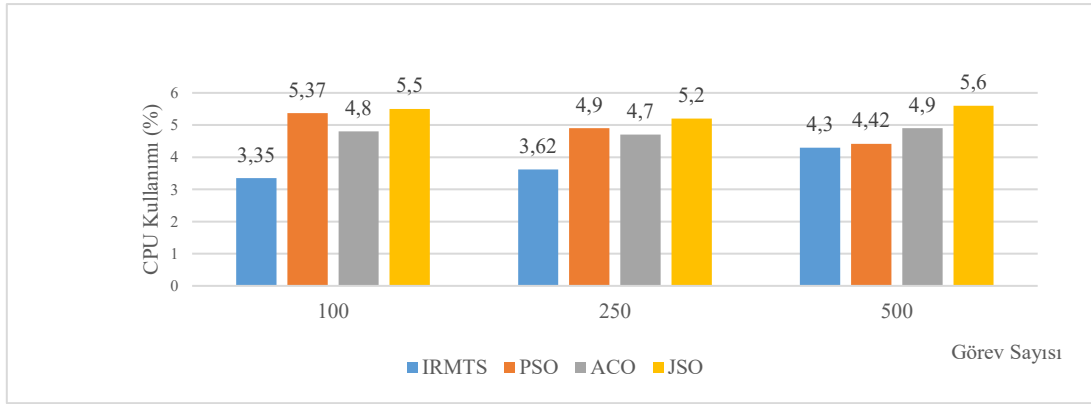
Şekil 3.4. Ortalama karar alma süreleri

IRMTS'nin kural tabanlı çalışması, çevrim içi safhada verimli bir kaynak kullanımını da beraberinde getirmiştir. IRMTS ve diğer metasezgisel algoritmaların deneyler süresince ortalama CPU ve hafıza kullanımları Şekil 3.5'te verilmiştir. IRMTS, ikinci en iyi CPU kullanım oranına sahip PSO'ya göre yaklaşık %35 daha iyi performans sağlamayı başarmıştır. Öte yandan sadece metasezgisel algoritmalar dikkate alındığında, PSO, ACO 'a göre yaklaşık %8 CPU' daha verimli kullanırken JSO en fazla CPU kaynağını kullanan yöntem olmuştur. Bunun sebebi JSO'nun iç döngü kullanımı ve dinamik popülasyon sirkülasyonudur.

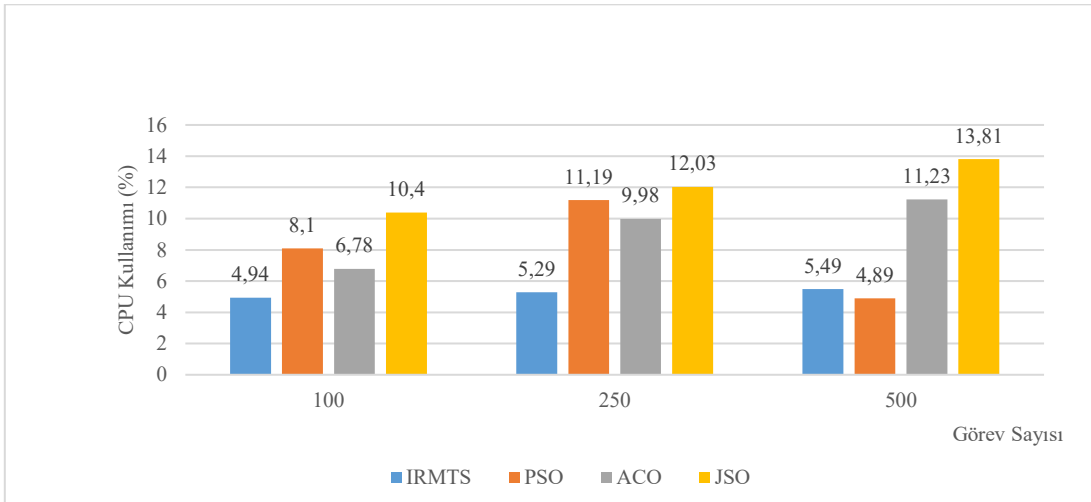




a) 5 VM ile ortalama CPU kullanımı



b) 10 VM ile ortalama CPU kullanımı

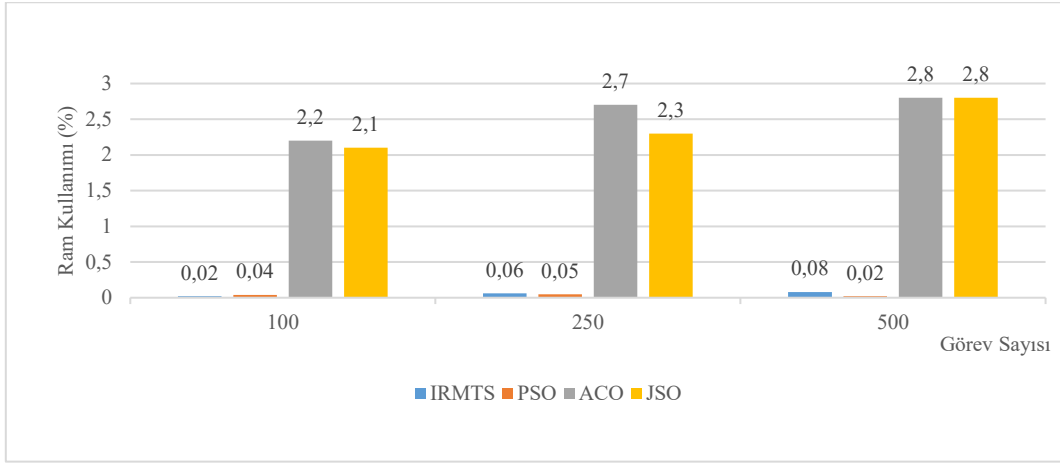


c) 20 VM ile ortalama CPU kullanımı

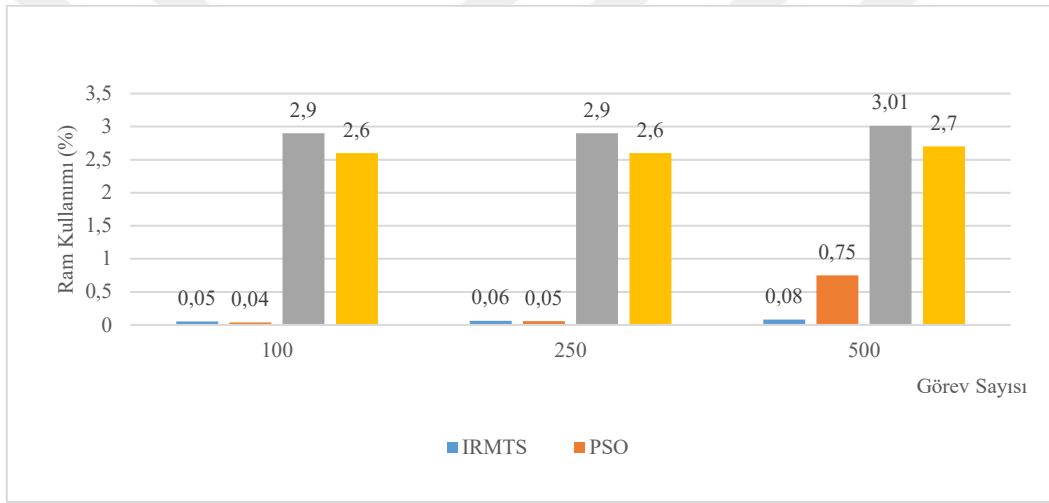
Şekil 3.5. CPU kullanımı yönünden yöntemlerin karşılaştırılması

Yöntemlerin çalışması esnasında kullanılan hafıza miktarı yönünden karşılaştırılması Şekil 3.6'da gösterilmiştir. Şekil 3.6'daki sonuçlar incelendiğinde hafıza kullanımları açısından neredeyse tüm yöntemler benzer kaynak kullanımı gerçekleştirmiştir. Problem boyutunun değişmesi IRMTS'nin kaynak kullanımını neredeyse hiç değiştirmemiştir. Özetle, IRMTS

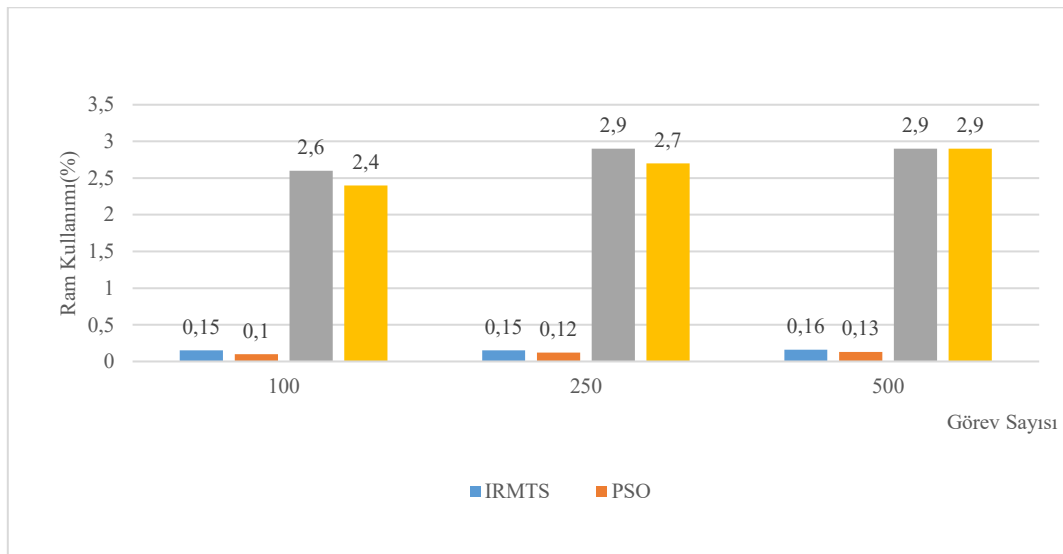
kaynak kullanımı açısından sisteme ciddi bir yük getirmeyip metasezgisel yöntemlere göre nispeten daha verimlidir.



a) 5 VM ile ortalama RAM kullanımı



b) 10 VM ile ortalama RAM kullanımı



c) 20 VM ile ortalama RAM kullanımı

Şekil 3.6. RAM kullanımı yönünden yöntemlerin karşılaştırılması

3.7. Bölüm Sonu Değerlendirmesi

Bu tez çalışmasında önerilen IRMTS yöntemi ile metasezgisel algoritmaların sağlamış olduğu başarılı bulut görev çizelgeleme çözümleri çok daha kısa sürede uygulanabilmektedir. IRMTS, bunu detaylı bir çözüm modeli havuzu kullanarak başarmaktadır. Yarı çevrim dışı aşamada, rastgele ve güncel senaryolar için metasezgisel çözümler ile oluşturulan bu model havuzu, güncellenebilir bir yapıdadır. Bu model havuzundan en uygun çözüm modelinin seçilebilmesi için yine yarı çevrim dışı aşamada kümeleme ve kural çıkarım adımları gerçekleştirilmektedir. Kümeleme aşaması senaryo tiplerini sınıflandırmada, çıkarılan kurallar ise atama planlarının dökümünde kullanılmaktadır. Kurallar ise metasezgisel makine öğrenmesi teknikleri ile otomatik olarak elde edilmektedir. Yapılan bağımsız deney sonuçlarına göre, IRMTS, metasezgisel yöntemle göre oldukça başarılı makespan değerleri elde ederken, belirsiz metasezgisel yöntemlerin makespan değerlerini yakalamış ve hatta çoğu zaman geçmiştir. En önemli başarımı ise, neredeyse aynı kaynak tüketimi ile klasik metasezgisel algoritmalara göre ortalama 18 kat daha hızlı çözüme ulaşabilesidir. Genel özellikleri ile IRMTS'nin metodolojisi, sadece bulut sistemler için değil aynı zamanda benzer görev çizelgeleme mekanizması kullanılan farklı sistemler içinde kolaylıkla uygulanabilir yapıdadır.

4. ÇOK AMAÇLI HİBRİT YÖNTEM

İki veya daha fazla çelişen amacın optimizasyonu ile ilgilenen çok amaçlı optimizasyonda (Multi Objective Optimization, MOO) temel prensip, karar vericiye ilgili amaçlar arasındaki dengeleri gözeterek bir veya birden fazla çözüm sunmaktır. Optimize edilecek amaç sayısı, 2 veya 3 tane ise problem çok amaçlı, daha fazla için ise birçok amaçlı olarak adlandırılır. MOO problemler için literatürde farklı yaklaşımlar önerilmiştir [88,89]. Bunlardan ilki ağırlıklandırılmış toplam (weighted sum), sözlüksel düzen (lexicographic) ve ϵ -Kısıt (ϵ -Constraint) gibi klasik priori çözüm yöntemleridir. Karar vericilerin çözüm sürecine başlangıçta dâhil olduğu bu tür yöntemlerde, problemin amaçları belirli ağırlıklara veya sıralamaya göre değerlendirir. Bu yöntemlerin en önemli dezavantajları, karar vericilerin öznel tercihlerinden dolayı sınırlı keşif ile ön yargılı çözümlere ulaşılmasıdır. Bu nedenle, tüm amaçların eşit muamele gördüğü ikinci MOO türü olan pareto temelli yaklaşımlar daha sık tercih edilebilmektedir. Posterior yaklaşımlarda amaçlar arasındaki mutlak denge gözetilir ve süreç sonunda karar vericiye (veya problem sahibine) bir birine mutlak üstünlüğü olmayan bir veya daha fazla pareto temelli sonuçlar verilir. Pareto temelli yöntemler, amaçlar hakkında bir ön bilgiye sahip olmanın gerekli olmadığı ve amaçların eşit öneme sahip olduğu problemler için uygundur. Pareto temelli yöntemlerde ana kriter dominasyondur ve bu kriter adayların birbirlerine olan çözüm üstünlüklerini temsil eder. Örneğin, bir maksimizasyon probleminde, domination kriteri Denklem 3.1 ile ifade edilir. Burada, $\vec{s}_1$ ve $\vec{s}_2$ çözümleri, k amaç sayısını, i çözüm vektöründeki ilgili amaç indeksini ve $f(\cdot)$ ise adayın amaç fonksiyonunu temsil etmektedir. Buna göre, Denklem 4.1'deki kriterin sağlanması durumunda $\vec{s}_1$ çözümü $\vec{s}_2$ 'nü domine etmiş demektir.

$$\forall i \in \{1,2,3, \dots, k\}, [F(s_1^i) \geq f(s_2^i)] \cap [\exists i \in \{1,2,3, \dots, k\}: f(s_1^i) > f(s_2^i)] \quad (4.1)$$

Diğer iki MOO yöntemi ise ilerici(progresive) ve hibrittir [90]. İlerici yöntemlerde karar verici her hangi bir iterasyonda çözüm sürecine müdahil olabilir ve belirleyici bir faktördür. Hibrit yöntemler ise tüm bu yaklaşımların birlikte kullanılması ilkesini esas alır. Hangi MOO yönteminin kullanılacağı, karar vericiye ve problem türüne bağlı olup, bu çalışmada öznellik kısıtına takılmamak için hibrit metasezgisel mekanizma kullanan pareto temelli çözüm yöntemini önerilmiştir. Bu yöntem, farklı metasezgisel algoritmaların aynı amaçlar için farklı süreçlerde eş zamanlı rastgele arama yapmalarına imkân sağlamaktadır. Bu tez çalışmasında metasezgisel algoritmalar olarak literatürde iyi bilinen ve görev çizelgeleme problemlerine uyarlanan NSGA-2 [91] ve SPEA-2 [92] kullanılmıştır.

4.1. NSGA-2 (Nondominated Sorting Genetic Algorithm-2)

Pareto optimal çözümlere ulaşmada etkili bir evrimsel temelli çok amaçlı metasezgisel optimizasyon yöntemidir. Özellikle, kullanmış olduğu başarılı kalabalık mesafesi (Crowding Distance, CD) mekanizması ile erken yakınsamayı önler ve popülasyonun keşif sürecini daha esnek hale getirir. NSGA-2'nin ana mekanizması ise domine olmayan sıralamadır. Bu mekanizma, her iterasyonda aday

çözümler arasında Denklem 4.1'e göre dominasyon kontrolü yapar ve adayları domine edilme sayılarına göre cephe (front) olarak adlandırdığı gruplara yerleştirir. Bir cephe F_n ($n \in N^+$) ile temsil edilir ve bu cephedeki aday çözümlerin domine edilme sayıları $n - 1$ 'dir. Yani, ilk cephe (F_1), hiçbir şekilde domine edilmemiş en başarılı adayları temsil ederken F_2 cephesindeki adaylar 1 kez domine edilmiştir. Bu cepheler sonraki nesillerin (yeni aday çözümlerin) oluşturulmasında başlıca rolü oynarlar. Bir sonraki neslin oluşturulması için seçilen cephe bireyleri arasındaki çeşitlilik güçlü bir keşif için önemlidir. Bu nedenle NSGA-2, M amaç için bir cephedeki her bir adayın seyreklik tahminini, Denklem 4.2'de verilen CD ile yapar. CD değeri, seçme veya eliminasyonda güçlü bir parametre olup, CD'si yüksek olan bir adayın bir sonraki neslin oluşumu için seçilme olasılığı daha yüksektir.

$$CD_i = \sum_{k=1}^M \frac{|F_{i+1}^k - F_{i-1}^k|}{F_{max}^k - F_{min}^k} \quad (4.2)$$

Temel aşamaları Algoritma 4.1'de verilen NSGA-2, diğer genetik algoritmalar gibi başlangıçta potansiyel aday çözümleri içeren bir populasyon havuzuna sahiptir. Başlangıç popülasyonundaki her bir adayın temsil ettiği çözüm önerisi probleme göre kodlanmıştır. Her iterasyonda ilk olarak aday çözümlerin amaç fonksiyon değerlerine göre cepheleri ve cephedeki lokasyonlarına göre CD değerleri hesaplanır. Turnuva usulü seçilen aday çözümler için genetik operasyonlar uygulanarak yeni aday çözümler üretilir ve yeni aday havuzuna eklenir. Yeni adayların ana popülasyona eklenmesi sonucunda sabit olması gereken boyut (N) artacaktır. Bu nedenle, front ve CD değerlerine göre en iyi N aday ana popülasyonda kalacak şekilde bir eleme yapılır. Nihai iterasyona ulaşıldıktan sonra karar vericiye domine olmayan çözümler sunulur.

Algoritma 4.1. NSGA-2'nin sözde kodu

-
1. $P \leftarrow$ Popülasyon oluştur
 2. $Q \leftarrow \emptyset$
 $N \leftarrow \text{Boyut}(P)$
 3. P içerisindeki her bir çözüm için onun amaç değerini hesapla
 4. P üzerinde baskın olmayan sıralama, cepheleri oluştur; $F_1, F_2, \dots, F_n$
 5. Tüm cephelerde bulunan çözümler için, CD hesapla
 6. Mevcut popülasyon P ile yavru havuzul Q ile birleştir
 7. **While** (İterasyon Sayısı < Maksimum İterasyon Sayısı)
 8. **While** Q 'nin boyutu < N :
 9. P 'den CD tabanlı seçim operatörü kullanarak iki ebeveyn çözüm seç
 10. Genetik Operasyonlar Uygula (Çaprazlama ve Mutasyon)
 11. Yavru çözümler için amaç fonksiyon değerlerini hesapla
 12. Yavru çözümleri Q 'ya ekle
 13. Birleştirilmiş popülasyonda domine olmayan sıralama gerçekleştir ($P \cup Q$)
 15. F_1 'den başlayarak frontlardan N adet çözüm seç
 16. **Eğer** F_1 cephesi ihtiyaçtan daha fazla çözüm varsa en iyi çözümleri seçmek için CD kullan
 17. Seçilen N adet çözüm ile P 'yi güncelle
 18. **Return** Domine olmayan çözümler
-

Çaprazlama ve mutasyon işlemleri evrimsel algoritmalarda temel genetik operasyonlar tasarımcıya göre farklılık gösterebilir. Bu çalışmada çaprazlama için Simüle Edilmiş İkili Çaprazlama (Simulated Binary Crossover, SBX) tercih edilmiştir. SBX'in en önemli avantajı popülasyondaki çeşitliliği koruması ve ebeveynlerin genetik bilgilerini birleştirerek yeni bireylere aktarabilmesidir. SBX ile ebeveynlerin değerlerine bağlı olarak iki yeni birey oluşur ve Denklem 4.3 $P(\delta)$ ile ifade edilen yayılma faktörü kullanılır. Buradaki δ değeri, çocuk çözümlerin ebeveynlerin genetik alanına ne kadar yayıldığını veya odaklandığını analiz etmek için kullanılır ve Denklem 4.4'te gösterildiği gibi hesaplanır. Yeni bireylerin ebeveynlerine ϕ benzerliği ($\delta \in \mathbb{R}^+$) ile belirlenir. Daha yüksek ϕ daha yüksek benzerlik oranı anlamına gelir. Çaprazlama işlemi sonunda üretilen yeni bireyler Denklem 4.5 ve Denklem 4.6 kullanılarak bulunur.

$$P(\delta) = \begin{cases} 0.5 \times (\phi + 1) \times \delta^\phi & \text{eğer } \delta \leq 1. \\ 0.5 \times (\phi + 1) \times \frac{1}{\delta^{\phi+2}} & \text{diğer durumlarda} \end{cases} \quad (4.3)$$

$$\delta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right| \quad (4.4)$$

$$of_1 = 0.5 \times (1 + \delta_h) \times eb_1 + (1 - \delta_h) \times eb_2 \quad (4.5)$$

$$of_2 = 0.5 \times (1 - \delta_h) \times eb_1 + (1 + \delta_h) \times eb_2 \quad (4.6)$$

NSGA-2’de mutasyon sonucu oluşan birey, asıl çözüme yakın değerlere sahiptir ve özellikle lokal minima (veya maksima)’dan kaçınmak için ciddi katkıda bulunur. Mutasyon işlemi Denklem 4.7 ve 4.8 ile gerçekleştirilir. Burada $s_n^{(m)}$ m ’inci popülasyona ait olan n ’inci bireyi, $s_n^{(U)}$ ve $s_n^{(L)}$ üst ve alt sınırları, γ karışıklık parametresini temsil eder ve $k \in [0,1]$ ’dir.

$$s_n^{(m+1)} = s_n^{(m)} + (s_n^{(U)} - s_n^{(L)}) \times \alpha_n \quad (4.7)$$

$$\alpha_n = \begin{cases} (2 \times k)^{\frac{1}{\gamma+1}} - 1 & \text{eğer } k < 0.5 \\ 1 - (2 \times (1 - k))^{\frac{1}{\gamma+1}} & \text{diğer durumlarda} \end{cases} \quad (4.8)$$

4.2. SPEA-2 (Strength Pareto Evolutionary Algorithm 2)

Bir diğer iyi bilinen evrimsel MOO algoritması SPEA2 [92], NSGA-2’den farklı olarak, aday çözümleri değerlendirmek için güç (strength) adı verilen ve popülasyon içindeki baskınlık seviyesini gösteren bir metrik kullanır. Bir aday çözüm, popülasyondaki diğer adayları ne kadar domine ederse o kadar yüksek bir güce sahiptir. Çeşitlilik için bir yoğunluk değerlendirme mekanizması kullanan SPEA2, klasik evrimsel mekanizmalardan farklı olarak popülasyonla (P) birlikte bir de sabit boyutlu arşiv ($\bar{P}$) yönetimi gerçekleştirir. Genel adımları Algorithma 4.2’de verilen SPEA-2, ilk iterasyonda ($t = 0$) ilk popülasyonun (P_0) domine olmayan çözümlerini boş arşive ($\bar{P}_0$) yükler. Her iterasyonda, güncel popülasyon (P_t) ve arşiv ($\bar{P}_t$) üyelerinin güçleri hesaplanır. i .nisi bireyin domine ettiği çözüm sayısı, onun güç değeridir ve Denklem 4.9 ile hesaplanır. Bu denklemde, $|\cdot|$ kümenin eleman sayısı, $+$ çoklu küme birleşimi ve $>$ ise pareto dominasyonunu temsil etmektedir. $S(i)$, ilgili aday çözümün popülasyon içindeki dominasyon durumunu gösteren ve Denklem 4.10 ile hesaplanan ham amaç ($R(i)$) bulmada kullanılır. Bir adayın diğer adaylar tarafından domine edilmesi onun $R(i)$ değerini arttırırken, $R(i) = 0$ olması ise o adayın domine olmadığını gösterir.

$$S(i) = |\{j | j \in P_t + \bar{P}_t \wedge i > j\}| \quad (4.9)$$

$$R(i) = \sum_{j \in P_t + \bar{P}_t, j > i} S(j) \quad (4.10)$$

Algoritma 4.2. SPEA2'nin sözde kodu

-
1. $P_t, \bar{P}_t, t = 0$
 2. Polulasyonu oluřtur
 3. $T_{maks} \leftarrow$ Maksimum iterasyon sayısı
 4. **While** ($t < T_{maks}$)
 5. Bireylerin ama deęerlerini hesapla (Denklem 6.10-6.14 ile)
 6. P_t ve $\bar{P}_t$ 'de bulunan domine olmayan bireyleri $\bar{P}_{t+1}$ 'e kopyala
 7. **Eęer** $\bar{P}_{t+1}$ boyutu $\bar{N}$ deęerini ařarsa then $\bar{P}_{t+1}$ 'in boyutunu azalt
 8. **Deęilse Eęer** $\bar{P}_{t+1}$ domine olmuř mlerle doldur
 9. Evrimsel iřlemleri gerekleřtir (Denklem 6.4- 6.8)
 10. Bireyleri gncelle
 11. $t = t + 1$
 12. **Return** Domine Olmayan mler
-

Domine olmayan adayların fazla olması durumunda $R(i)$ deęerlendirme iin yetersiz kalacaęı iin adaylara ait yoęunluk bilgisi de kullanılır. Bunun iin k 'nıncı en yakın komřu teknięi kullanılır. Bu teknikte i 'ninci birey iin dięer aday j 'e olan mesafeleri hem arřiv hem de poplasyon iin hesaplanarak listelenir. Artan sıraya gre dzenlenen listede, k 'nıncı deęer istenilen mesafeyi verir ve σ_i^k ile gsterilir. k deęeri, poplasyon ve arřiv boyutuna baęlı olup $k = \sqrt{|P_t| + |\bar{P}_t|}$ ile bulunur. Denklem 4.11 'de grleceęi zere, adayın yoęunluk deęeri σ_i^k ile ters orantılıdır. Burada payda deęerinin pozitif olması iin 2 deęeri kullanılmıřtır. Sonu olarak bir adayın fitness deęeri, ham ama ve yoęunluk deęerlerine gre Denklem 4.12 ile belirlenir.

$$D(i) = \frac{1}{\sigma_i^k + 2} \quad (4.11)$$

$$F(i) = R(i) + D(i) \quad (4.12)$$

Sonraki jenerasyonların oluřturulmasında adayların $F(i)$ deęerleri kullanılır. Bunun iin bařlangıta, Denklem 4.13'te gsterildięi zere, ama deęeri 1'den kk olan domine olmayan adaylar arřive alınır. Ancak burada bir sonraki jenerasyonun arřiv boyutu nemlidir. Eęer $\bar{P}_{t+1} = \bar{N}$ ise sre tamamlanır. $\bar{P}_{t+1} < \bar{N}$ olması durumunda, bir nceki arřiv ve poplasyondaki en iyi $\bar{N} - |\bar{P}_{t+1}|$ tane domine olmayan m bir sonraki jenerasyonun arřivine alınır. Eęer $\bar{P}_{t+1} > \bar{N}$ ise, $\bar{P}_{t+1} = \bar{N}$ řartı saęlanıncaya kadar aday eliminasyonu yapılır. Aday eliminasyonunda, σ_i^k belirleyicidir ve Denklem 4.14'te gsterildięi gibi aynı uzaklıęa sahip birden fazla aday mün olması durumunda, ikinci en kk uzaklıklar dikkate alınır. Sonraki jenerasyonun yeni aday mlerin elde edilmesinde yine genetik operasyonlar kullanılır. Bu alıřmada, SPEA2, NSGA-2'deki aynı genetik operasyonları kullanmıřtır.

$$\bar{P}_{t+1} = \{i | i \in P_t + \bar{P}_t \wedge F(i) < 1\} \quad (4.13)$$

$$i \leq dj \Leftrightarrow \forall 0 < k < |\bar{P}_{t+1}|: \sigma_i^k = \sigma_j^k \vee \exists 0 < k < |\bar{P}_{t+1}|: [(\forall 0 < l < k: \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^k] \quad (4.14)$$

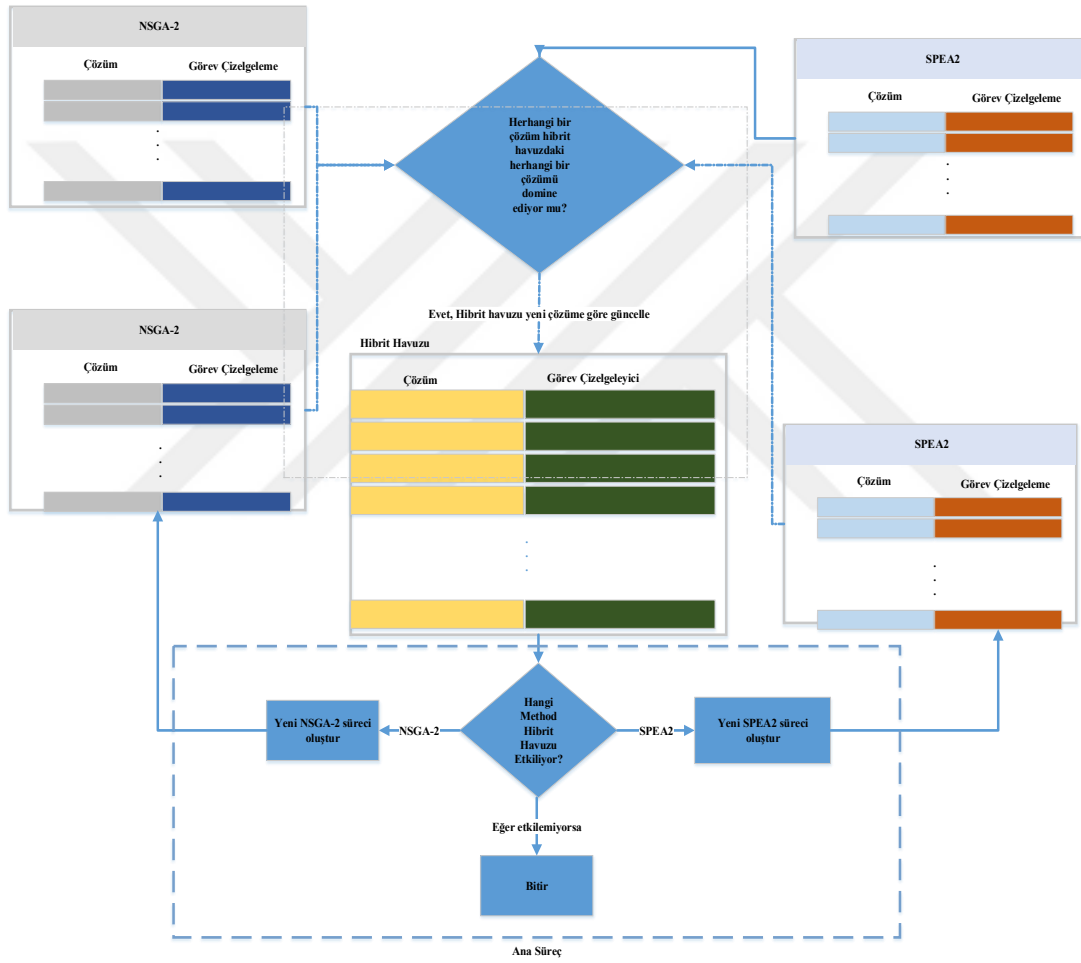
4.3. Önerilen Hibrit Paralel MOO Görev Planlama Yaklaşımı

Metasezgisel temelli optimizasyon, rastgele arama temelli belirsiz olmayan bir çözüm yöntemidir. Her rastgele aramanın başarı ile sonuçlanacağı garanti edilemez. Ayrıca, bu yöntemlerin iteratif süreçlerinden kaynaklanan zaman ve sistem kaynağı maliyetleri de yüksek olabilmektedir. Optimum çözümün aranması için algoritma parametrelerinin zorlanması (örneğin yüksek iterasyon sayıları gibi) veya tekrarlama seçenekleri bu maliyetleri daha da arttırmaktadır. Bir başka mesele ise farklı arama algoritmalarına sahip metasezgisel yöntemlerin, problem türüne göre farklı sonuçlar üretebilmeleridir. Aynı anda birden fazla amaca yönelik yapılacak optimizasyonda bu hususlar daha belirginleşebilmektedir. Bu nedenle, bu çalışma, günümüzün paralel yazılım geliştirme teknolojilerinden faydalanarak, farklı MOO algoritmalarını, yarışma temelli çalıştıran bir yaklaşımı önermektedir. Diğer paralel yöntemlerden en önemli farkı, anlık başarımlar ve kaynak kullanımına göre paralel arama yapan MOO örneklerinin sayısını kontrol altında tutabilmesidir.

Hibrit paralel rastgele aramanın en büyük avantajı, tek bir algoritmanın çalışma süresi içerisinde, birden fazla farklı algoritma ve popülasyon dağılımları ile daha esnek optimizasyonun yapılabilmesidir. Ancak, sabit sayıda paralel aktif arama yapan MOO örnekleri kullanmanın bazı dezavantajları oluşabilmektedir. Örneğin benzer keşif süreçlerini yürütmeye başlayan örneklerin iterasyon başarımlarını etkilemediği gibi gereksiz sistem kaynağı da tüketirler. Örnek sayısının fazla olması hem kaynak israfını hem de görevler arasında işlemcinin geçiş yapma (context switching) maliyetini artırır. Düşük sayıda örnek kullanılması da büyük boyutlu problemlerde arama uzayının yeterli taranmamasına veya yerel minimalara takılmalara sebep olabilir. Öte yandan, iterasyon başarımlarının anlık değerlendirilmesi ile aktif paralel örneklerin sayısının kontrollü bir şekilde artırılıp azaltılması bu dezavantajın önüne geçebilir. Bu çalışma, iki farklı süreç türü kullanan bir mekanizmaya sahiptir. Bunlardan ilki, NSGA-2 ve SPEA2 algoritmalarının yürütüldüğü MOO işçi süreçlerdir. İkincisi ise anlık başarımları değerlendirerek aktif çalışan MOO türünü ve örnek sayısını belirleyen ana süreçtir.

Temel çalışma prensipleri Algoritma 4.3 ve Şekil 4.1'de gösterilen önerilen yaklaşım, belirli sayıda NSGA-2 ve SPEA2 süreçleri ile paralel optimizasyon sürecine başlar. Her bir süreç, kendi popülasyonuna sahiptir ve iterasyonlarında kendi algoritma adımlarını yürütür. Ancak her iterasyon sonunda, NSGA-2 süreçleri kendi F_1 çözümlerini, SPEA-2 süreçleri ise kendi $\bar{P}_{t+1}$ çözümlerini paylaşımlı bir hafıza alanında bulunan ve ana süreç tarafından yönetilen bir hibrit havuzda yarışır. İterasyonlar sonunda, sadece bu havuzda önceden bulunan çözümler tarafından domine edilemeyen çözümler havuza dahil edilir. Hibrit havuzu boyutu, uyumluluk nedeni ile SPEA-2 arşiv boyutunu aşamaz. Boyutun aşması durumunda, hızlı olmasından dolayı NSGA-2'de kullanılan CD mekanizması

benimsenir ve havuzdaki her bir çözüm için yeni CD değerleri hesaplanır. Düşük CD değerli hibrit havuz adayları ana süreç tarafından havuzdan çıkarılır. Yeni iterasyona başlayan her süreç belirli sayıda yeni çözümü hibrit havuz çözümleri ile diğer kısmını da kendi iç mekanizması ile elde eder. Bu işleyiş, her algoritmaya hızlıca en iyi çözüme ulaşma imkanı sağlarken ayrıca her iterasyonda yeni ve esnek keşif süreçlerinin de oluşmasına imkan verir. Ana süreç, ön tanımlı bir hibrit havuz güncelleme denemesi ($U = 10$) kadar hibrit havuzda değişiklik olmadığını gördüğünde daha önce en çok domine olmayan aday üretmiş algoritma türünden yeni süreçleri aramaya dâhil eder. Yine ön tanımlı iterasyon sonucunda hibrit havuza katkıda bulunmayan süreçleri elimine ederek süreç yönetimini aktif şekilde sürdürür. Tüm iterasyonlar tamamlandıktan sonra hibrit havuz çözümleri karar vericiye sunulur.



Şekil 4.1. Önerilen hibrit yöntemin genel aşamaları

Algoritma 4.3. Önerilen hibrit-paralel yöntemin sözde kodu

1. Ana işlemi ve paylaşılan bellek alanını (Hibrit Havuz) başlat
2. NSGA-2 ve SPEA2 Algoritma 4.1 ve 4.2'ye göre oluşturup çalıştır
3. **While** Çalışan bir işlem varsa
4. **Eğer** Herhangi bir NSGA-2 veya SPEA2 süreci tamamlanırsa
5. F_1 veya $\bar{P}_{t+1}$ çözümlerini hibrit havuza taşı
6. Hibrit havuz için bir dominasyon kontrolü yap ve yalnızca domine edilmemiş çözümler ekle (U 'yu arttır)
7. Hibrit havuzdaki her çözümün CD değerlerini hesapla
8. Hibrit havuzu dolduysa CD değerlerine göre gerekli çözümü çıkar
9. Sonraki nesli (her örnek için) oluşturmak amacıyla hibrit havuzdan ana çözümleri seç
10. Hibrit havuza katkıda bulunan süreci etiketle ve katkı puanını artır
11. **Eğer** $U == 10$ ise sıfır katkı puanına sahip tüm süreçleri sonlandırın ve yeni bir süreç başlat
14. Hibrit havuzdaki baskın olmayan çözümleri gönder

4.4. Deneyler ve Tartışma

Önerilen yöntemin başarımlı testleri, CloudSim üzerinde üç farklı görev çizelgeleme senaryosu için yapılmıştır. Ana parametreleri Tablo 4.1'de verilen senaryoların her biri için 10 farklı bağımsız deney gerçekleştirilmiştir. Her bağımsız deneyde, ilgili senaryodaki VM'lere ait işlem kapasiteleri ve görev komut sayıları rastgele olarak tekrar oluşturularak bulut sistemin dinamik yapısı simüle edilmiştir. Makespan ve enerji tüketimi için yapılan optimizasyon deneylerinin sonuçları, hem klasik NSGA-2 ve SPEA2 hem de sezgisel FCFS ve SJF algoritmaları sonuçları ile karşılaştırmalı olarak verilmiştir. NSGA-2 ve SPEA2 algoritmalarında kullanılan parametreler Tablo 4.1 ve Tablo 4.2'deki gibidir. Simülasyonlar, i5-10400 2.90GHz CPU ve 8GB RAM bellekli bilgisayar üzerinde yapılmıştır.

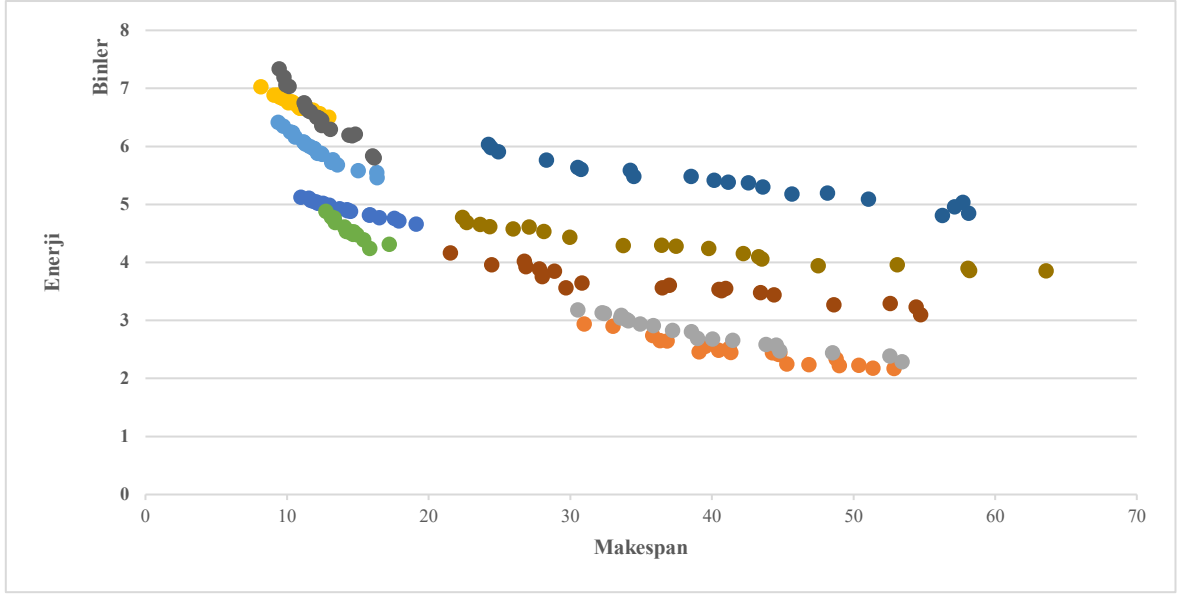
Tablo 4.1. Deneylerde kullanılan senaryolara ait parametreler

Senaryolar	VM Sayısı	Görev Sayısı	Deney Sayısı	Görev Komut Sayısı (MIPS)	CPU (MIPS)
Senaryo 1	5	100	10	4 000-6 000	1 000-20 000
Senaryo 2	10	250	10	4 000-6 000	1 000-20 000
Senaryo 3	20	500	10	4 000-6 000	1 000-20 000

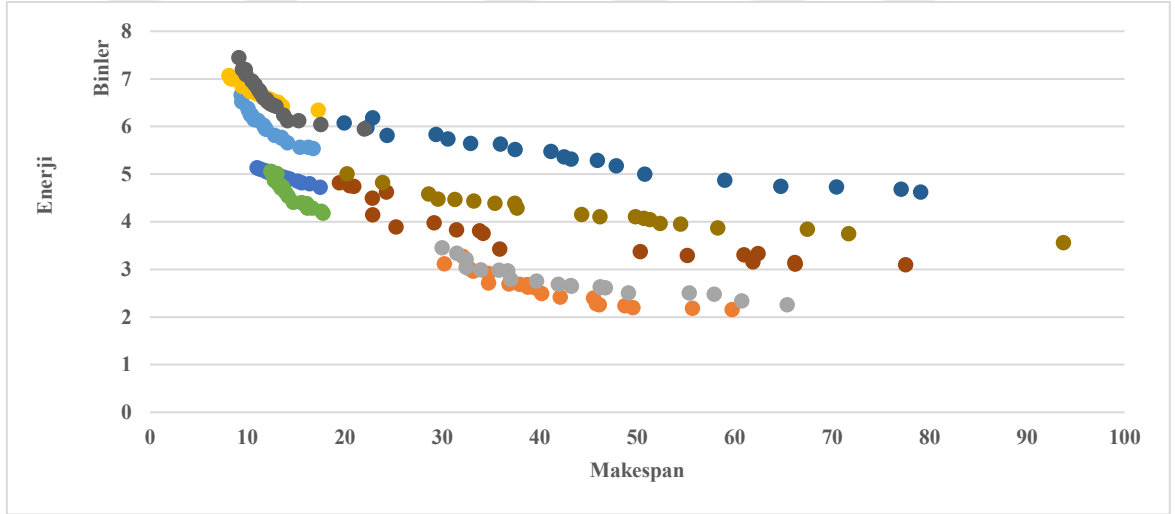
Tablo 4.2. NSGA-2 ve SPEA2 parametreleri

Parametre	Değer
P	20
$\bar{P}$	20
λ	0.5
γ	5
ϕ	5
δ	$[0,2] \in R$
Çaprazlama Olasılığı	0.9
Mutasyon Olasılığı	0.1
Maksimum iterasyon	100

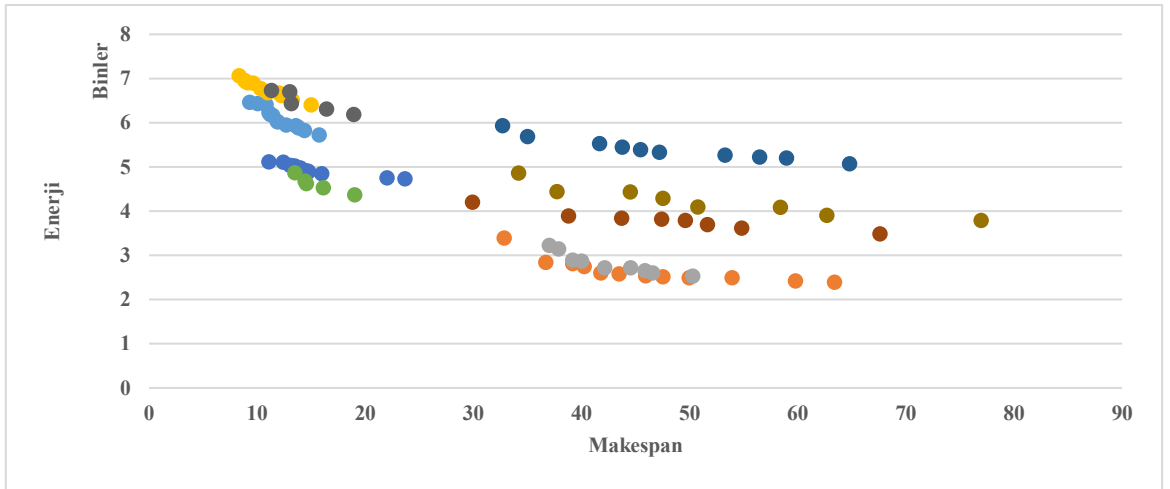
Genel karşılaştırmalara geçmeden önce tüm metasezgisel çözümlerin elde ettiği pareto çözümler incelenmiştir. Şekil 4.2’de, ilk senaryo için tüm metasezgisel algoritmalar tarafından elde edilen pareto sonuçlarını göstermektedir. Grafiklerde, her bağımsız deneye ait pareto sonuçlar farklı bir renkte gösterilmiştir. İlk senaryoda, SPEA2 ve hibrit yöntem hem pareto çözüm yoğunluğu hem de sonuç karakteristiklerin açısından birbirine yakın iken NSGA-2 özellikle enerji metriği açısından biraz geride kalmıştır. Deneylerde elde edilen maksimum enerji tüketimi ve makespan değerleri göz önünde bulundurulduğunda Hibrit yöntemin enerji açısından az da olsa bir iyileştirme yaptığı, makespan değerinde oldukça başarılı olduğu gözlemlenmiştir. NSGA-2 ile elde edilen çözümlerin sayısı diğer yöntemlere göre daha az olup CD değerleri oldukça yüksektir. Her bir yöntem için Senaryo 2’e göre 10 bağımsız deney sonucunun pareto cephe grafikleri Şekil 4.3’te gösterilmiştir. Sonuçlar Hibrit yöntemin SPEA2’ye göre hem enerji hem de Makespan değeri açısından daha iyi sonuçlar verdiğini göstermektedir. Hibrit yöntemin makespan aralığının NSGA-2’ye oranla daha düşük olduğu; enerji tüketimi açısından ise benzerlik gösterdikleri görülmüştür. Son olarak her bir yöntem için Senaryo 3’e göre 10 bağımsız deney sonuçlarının pareto cephe grafikleri Şekil 4.4’te gösterilmiştir. Sonuçlar diğer senaryolara göre yapılan sonuçlar ile benzerlik göstermektedir.



a) Hibrit Sonuçları

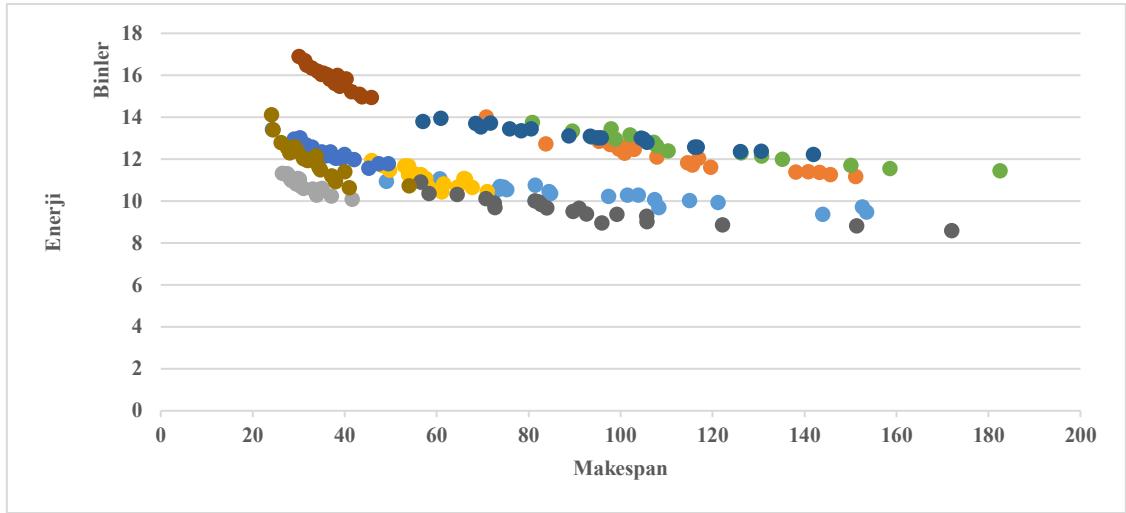


b) SPEA2 Sonuçları

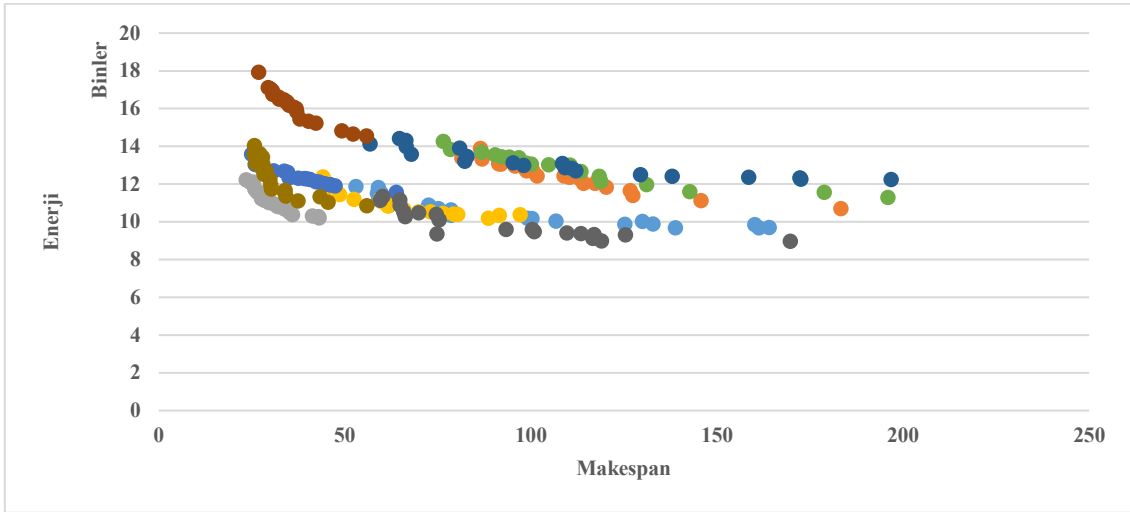


c) NSGA-2 Sonuçları

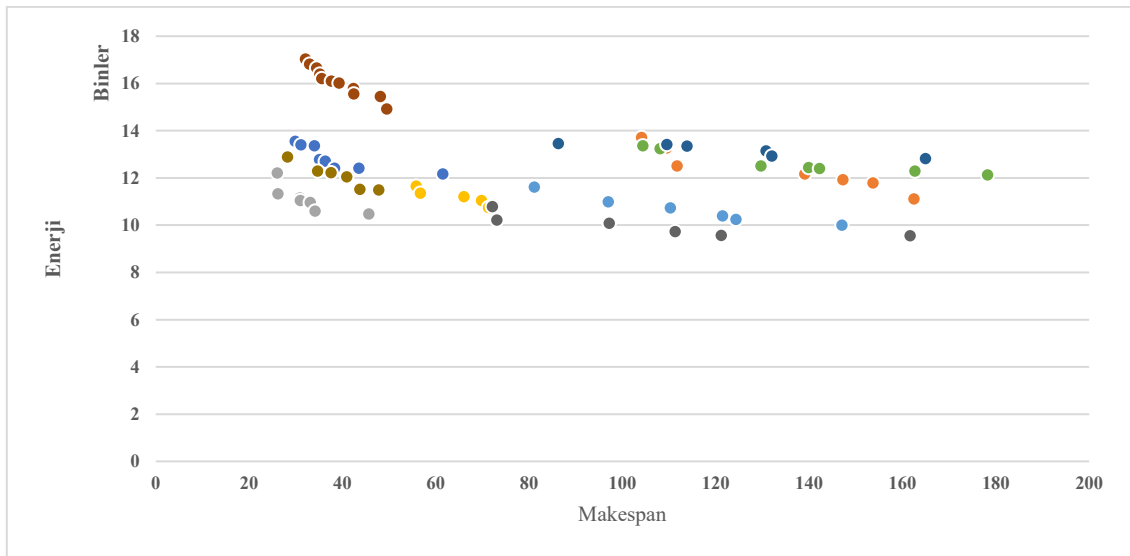
Şekil 4.2. Senaryo 1 için pareto grafikleri



a) Hibrit Sonuçları

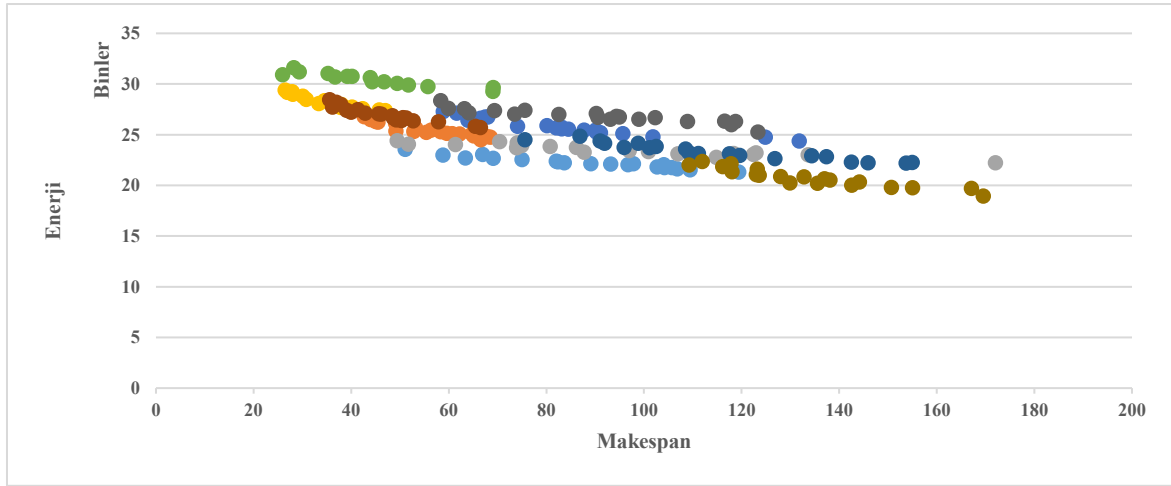


b) SPEA2 Sonuçları

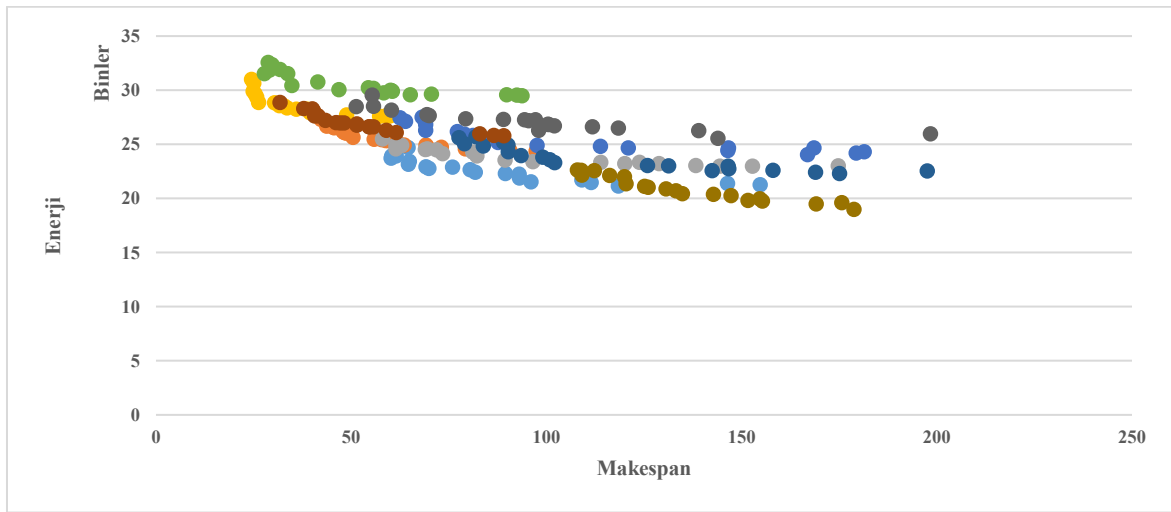


c) NSGA-2 Sonuçları

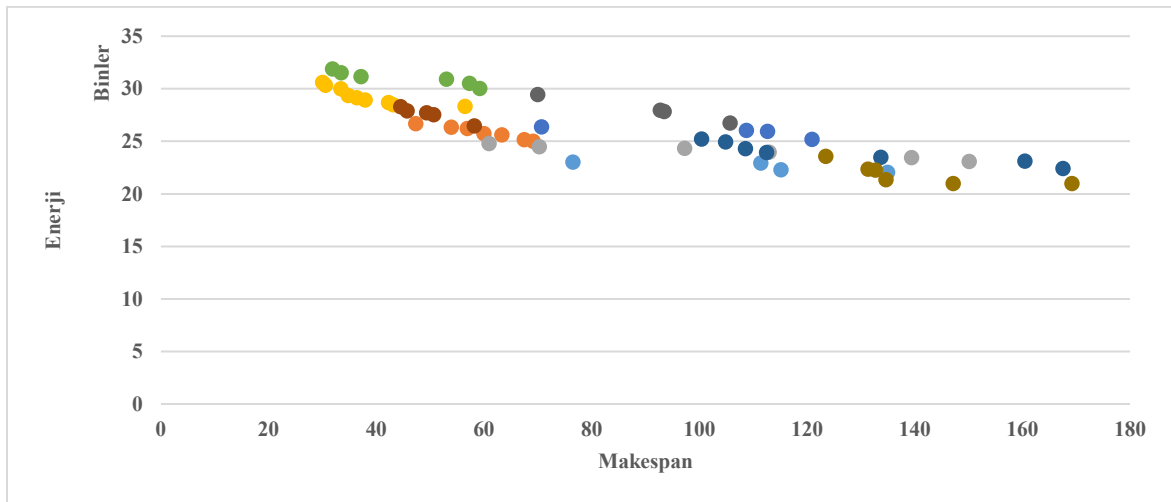
Şekil 4.3. Senaryo 2 için pareto grafikleri



a) Hibrit Sonuçları



b) SPEA2 Sonuçları



c) NSGA-2 Sonuçları

Şekil 4.4. Senaryo 3 için pareto grafikleri

Tablo 4.3'te önerilen yöntem, SPEA2 NSGA-2, FCFS ve SJF yöntemlerine göre her 3 senaryo için elde edilmiş bazı makespan ve enerji değerleri verilmiştir. Senaryo 1'de önerilen yöntemin elde ettiği [8.12, 7 027.65] ve [9.36, 6 419.58] sonuçları SPEA2 ve NSGA-2 yöntemlerinin elde ettiği sonuçlara baskın olduğu gözlemlenmiştir. Diğer deney sonuçlarında ise önerilen hibrit yöntemin sonuçları SPEA2 ve NSGA-2 yöntemleri tarafından domine edilmemiştir. Önerilen hibrit yöntem sonuçları FCFS ve SJF algoritmalarının sonuçları ile kıyaslandığında önerilen yöntemin makespan değerinin önemli ölçüde diğer yöntemlere göre iyi olduğu buna karşın enerji tüketimi açısından daha yüksek tüketim olduğu gözlemlenmiştir. Senaryo 2'de hibrit yöntemin [24.3, 13 423.16], [30.04, 16 901.33] elde ettiği sonuçlar SPEA2 ve NSGA-2'nin sonuçlarını domine etmiştir. Diğer 3 adet deney sonuçlarında ise SPEA2 ve NSGA-2 yöntemlerinin sonuçlarına domine olmamıştır. FCFS ve SJF yöntemlerinin sonuçları ile hibrit yöntem sonuçları kıyaslandığında sonuçların Senaryo 1'deki sonuçlarla benzerlik göstermektedir. Senaryo 3'teki sonuçlar göz önünde bulundurulduğunda hibrit yöntemin [25.9, 30 918.26] ve [49.38, 24 409.22] elde ettiği sonuçların SPEA2 ve NSGA-2'nin sonuçlarını domine etmiştir. Hibrit yöntemin elde ettiği diğer sonuçlar SPEA2 ve NSGA-2 yöntemlerinin sonuçlarına domine olmamıştır. 3 senaryo dikkate alındığında önerilen hibrit yöntemin her bir senaryoda 5 çözümden 2 adet deneyde domine çözümler ürettiği buna karşın 3 çözümde domine olmadığı gözlemlenmiştir.

Tablo 4.3. Tüm yöntemler için karşılaştırmalı sonuçlar

Senaryo	Önerilen Yöntem		SPEA2		NSGA-2		FCFS		SJF	
	Makespan	Enerji	Makespan	Enerji	Makespan	Enerji	Makespan	Enerji	Makespan	Enerji
Senaryo 1	8.12	7 027.65	8.13	7 070.48	8.33	7 064.10	11.55	6 787.97	100.03	4 611.23
	9.36	6 419.58	9.41	6 523.61	9.42	6 466.77	12.09	6 124.09	16.16	5 381.79
	9.44	7 341.07	9.48	7 202.23	11.33	6 732.51	14.22	6 737.27	24.75	5 753.82
	10.98	5 126.30	10.99	5 138.87	11.09	5 117.27	14.24	4 971.06	101.34	4 008.35
	12.74	4 882.01	12.41	5 062.26	13.49	4 873.29	16.12	4 762.85	101.97	4 879.24
Senaryo 2	24.04	14 121.98	25.7	13 575.51	28.22	12 888.30	42.14	12462.28	41.85	12 457.59
	24.30	13 423.16	24.94	13 574.90	29.90	13 557.64	42.36	12 813.2	42.37	12 888.72
	26.42	11 336.65	25.87	11 731.25	30.98	11 047.19	31.57	11289.27	31.55	11 296.39
	30.04	16 901.33	30.08	17 035.08	32.07	17 040.97	40.69	16323.37	42.18	16 193.35
	45.84	11 946.94	44.16	12 382.87	55.78	11 661.13	63.8	11796.76	62.69	11 775.31
Senaryo 3	25.90	30 918.26	27.69	31 546.44	33.43	31 540.4	62.07	31275.67	62.94	31 239.31
	26.42	29 414.10	24.45	31 008.64	30.01	30 622.07	42.59	29233.15	41.60	29 260.77
	35.54	28 456.00	31.72	28 870.43	44.50	28 298.99	62.07	31275.67	62.26	27 832.66
	40.02	27 538.53	42.23	27 300.82	47.31	26 701.00	65.22	25794.86	63.03	25 843.70
	49.38	24 409.22	57.94	25 494.34	60.91	24 789.54	128.57	24459.39	125.54	24 543.77

Tablo 4.4'te önerilen yöntem, SPEA2 ve NSGA-2 yöntemleri her 3 senaryo için (her senaryo 10 adet bağımsız deney) en iyi, en kötü, ortalama, medyan ve standart sapma metrik değerleri hesaplanarak verilmiştir. Tablo 4.4 incelendiğinde önerilen yöntemin makespan değerinin en iyi, en kötü, ortalama ve standart sapma değerlerinin SPEA2 ve NSGA-2 yöntemlerine göre daha iyi olduğu görülmektedir. Bu durum Şekil 4.2'de verilen sonuçlar ile benzerlik olduğunu göstermektedir. Senaryo 2 ve Senaryo 3'teki sonuçlar birlikte değerlendirildiğinde önerilen hibrit yöntemin enerji değerlerinin SPEA2 ve NSGA-2 yöntemlerine göre daha iyi olduğu görülmektedir. Makespan açısından bakıldığında ise en iyi ve en kötü değerlerinin diğer yöntemlere çok yakın olduğu buna karşın ortalama, ortanca ve standart sapma değerlerinin daha iyi olduğu gözlemlenmiştir. Senaryo 2 ve Senaryo 3 görev sayısı ve VM sayısı yönünden Senaryo 1'e göre daha yoğun kaynak ve görev sayısı içermektedir. Tablo 4.4'teki sonuçlar daha karmaşık problemlerin çözümünde önerilen yöntemin enerji tüketimi açısından diğer yöntemlere göre çok daha başarılı olduğunu göstermektedir.

Tablo 4.4. Önerilen yöntem, SPEA2 ve NSGA-2 için deney sonuçları performans metrikleri

Senaryo	Metrikler	Önerilen Yöntem		SPEA2		NSGA-2	
		Makespan	Enerji	Makespan	Enerji	Makespan	Enerji
Senaryo 1	En İyi	8.14	2 171.59	8.17	2 156.04	8.33	2 393.75
	En Kötü	67.71	7 341.07	93.76	7 445.48	76.96	7 064.10
	Ortalama	26.19	4 727.19	27.89	4 773.38	29.62	4 811.97
	Medyan	20.33	4 776.13	19.68	4 839.68	22.01	4 928.07
	Standart Sapma	15.31	14 27.34	18.64	1 426.01	18.44	1 415.39
Senaryo 2	En İyi	24.04	85 95.55	23.56	8 970.15	26.06	9 557.07
	En Kötü	182.48	16 901.33	196.84	17923.04	178.26	17040.97
	Ortalama	70.07	12 008.78	72.65	12161.16	79.23	12 523.99
	Medyan	60.68	11 902.51	64.78	12 106.18	66.03	12 290.13
	Standart Sapma	38.73	1 775.75	42.12	1 873.12	45.89	1 886.91
Senaryo 3	En İyi	25.90	18 945	24.45	18 987.72	30.01	20 983.08
	En Kötü	171.97	31 640.53	198.34	32 585.00	169.21	31 892.47
	Ortalama	78.24	25397.48	84.90	25 554.66	82.38	26 190.95
	Medyan	69.36	25 467.02	77.58	25 566.43	69.41	26 215.08
	Standart Sapma	36.29	2 923.46	42.29	3 000.55	41.88	2 995.13

4.5. Bölüm Sonu Değerlendirmesi

Bu çalışmada, bulut sistemlerde minimize edilmesi önemli olan ve bir birleri ile çelişen makespan ve enerji tüketimi amaçlarını dikkate alarak görev çizelgeleme yapan bir paralel hibrit metasezgisel yaklaşım önerilmiştir. Önerilen yaklaşımın başarısı, farklı senaryolar için yapılan bağımsız simülasyon deneyleri ile ispatlanmıştır. Genel olarak, tüm senaryolar için, önerilen yaklaşım, deneylerin % 40'ında diğer yöntemlere ait çözümleri, her iki amaç için domine etmiş, geri kalanlarında ise domine olmayan çözümler üretmeyi başarmıştır. Özellikle VM sayısı ve görev sayısı arttıkça önerilen yöntemin diğer yöntemlere üstünlüğü daha iyi görülmüştür. En yüksek başarımlar genellikle makespan amacıyla olmuş, özellikle domine olmayan çözümlerde ise enerji metriğinde yaklaşık çözümler yakalanmıştır. Önerilen yöntemin başarısında, birbirleri ile farklı çalışma mekanizmalarına sahip iki son teknoloji MOO algoritmasının yarışma temelli paralel çalıştırılması yatmaktadır. Algoritma süreçleri ile yine paralel çalışan bir ana süreç, çözüm sürecine katkıda bulunmayan süreçleri elimine ederek ve gerekli olması durumunda yeni süreçler üreterek daha esnek ve verimli keşif süreci yönetmiştir. Önerilen bu yaklaşımın uygulanabilirliği ve başarısı yazarları kenar (edge) ve sis (fog) sistemlerde benzer çözümler aramaya yöneltmiştir.

5. TARTIŞMA

Bulut sistemler, verilerin ve uygulamaların internet üzerinden erişilebilen bir altyapıda depolanmasına ve yönetilmesine olanak tanıyan teknolojilerdir. Bu sistemler, işletmelere ve bireylere yüksek kapasiteli veri depolama, hızlı işleme, esneklik ve maliyet tasarrufu gibi avantajlar sunar. Hızlı bir büyüme sağlayan bulut bilişim pazarının 2028 yılına kadar bir trilyon doları aşması beklenmektedir. Büyük bir ekonomik değere sahip olan bulut sistemler yapı olarak karmaşıktır. Yüz binlerce hesaplama veya depolama kaynağına sahip olabilen bu sistemlerin kullanıcılarının taleplerine optimum bir şekilde cevap vermesi zor bir problemidir. Bulut sistemlerde görev çizelgeleme problemi olarak da ifade edilen ve kullanıcıların taleplerinin optimum bir şekilde bulut kaynaklarına dağıtılması NP zor sınıfına girmektedir.

Literatürde çokça çalışılan bu konu ile ilgili çeşitli yöntemler önerilmiştir. Bu tez çalışmasında önerilen tek amaçlı kural tabanlı IRMTS yönteminde literatürde önerilen yöntemlerden farklı olarak farklı senaryolar dikkate alınarak hazırlanan sentetik veri setinden üretilmiş kurallar dikkate alınmıştır. Bu kuralları kullanarak eldeki senaryonun sınıfı tespit edilmiştir. Daha sonra senaryonun tespit edilen sınıf içerisindeki senaryolar ile benzerliği kontrol edilmiştir. Eldeki senaryoya en çok benzeyen senaryo referans alınarak görev çizelgeleme gerçekleştirilir. Optimizasyon doğası gereği oldukça maliyetli bir iştir. Bulutun dinamik yapısı ve kullanıcı taleplerinin de değişkenlik göstermesi gibi etkenler dikkate alınarak her bir görev çizelgeleme için optimizasyon sürecinin gerçekleştirilmesi hem kaynakların verimli kullanılmamasına hem de gecikmelere neden olabilmektedir. Bu tez çalışmasında önerilen IRMTS yönteminin en yenilikçi tarafı optimizasyon maliyetini minimize etmesidir. Önerilen yöntem PSO, ACO ve JSO yöntemleriyle karar verme süreci yönünden kıyaslandığında yaklaşık 18 kat daha hızlı olduğu tespit edilmiştir. Önerilen yöntem görev çizelgeleme süresi yönünden de PSO, ACO ve JSO'ya önemli bir üstünlük göstermiştir. Senaryoların karmaşıklığı arttıkça (görev sayısı ve bulutta mevcut olan kaynak miktarının artması durumunda) önerilen yöntemin diğer yöntemlere göre başarısı artmaktadır. Karmaşık senaryolar metasezgisel yöntemlerin arama uzayının artmasına sebep olurken aynı zamanda yöntemlerin optimum çözüme ulaşma olasılığını düşürmektedir. Senaryoların karmaşıklık düzeyinin artmış olması IRMTS'nin karar verme ve görev çizelgeleme süresini diğer yöntemlere göre oldukça az etkilemektedir.

Metasezgisel yöntemler doğası gereği gereği optimum çözüm üretmezler. Bunu yerine optimuma yakın çözümler üretirler. Arama uzayının daha etkin kullanılması ve farklı sonuçların tespit edilebilmesi amacıyla bu tez çalışmasında önerilen ikinci yöntem ise çok amaçlı paralel hibrit yöntemdir. Bu yöntem NSGA2 ve SPEA2 yöntemlerinin birbirlerine olan üstünlüklerini dikkate alarak farklı süreçler halinde paralel olarak çalıştırır. Yöntemlerin elde etmiş olduğu en iyi sonuçlar ortak bir hibrit havuzda toplanmaktadır. Ayrıca ana süreç olarak ifade edilen süreç NSGA2 ve

SPEA2 süreçlerinin ortak havuza yapmış olduđu katkıları dikkate almaktadır. Ana süreç ve bu yöntemlerin katkı oranlarına göre yeniden süreçlerin optimizasyon sürecine dahil edilmesi kararını verir. Deneysel sonuçlar önerilen hibrit yöntemin yaklaşık %40 oranında NSGA2 ve SPEA2'nin çözümlerini domine ettiğini göstermiştir. Hibrit yöntemin NSGA2 ve SPEA2 yöntemlerinden farklı sayıda süreçler oluşturması ve bunları paralel çalıştırması daha yoğun işlemler yapmasını gerektirmektedir. Bu durum hibrit yöntemin NSGA2 ve SPEA2'ye göre daha fazla zaman ve daha fazla enerji tüketmesine sebep olmaktadır. Zaman gecikmesinin azaltılması amacıyla önerilen hibrit yöntem süreçleri paralel olarak yürütmektedir.



6. SONUÇLAR

1. Bulut sistemlerde görev çizelgeleme için önerilen tek amaçlı kural tabanlı IRMTS yöntemi metasezgisel optimizasyon yöntemleri olan PSO ACO ve JSO ile kıyaslanmıştır. Deneysel sonuçlar önerilen yöntemin makespan açısından oldukça başarılı olduğunu göstermiştir.
2. Görev-VM arasındaki eşleştirmenin tespit edilmesi için geçen süre dikkate alındığında önerilen IRMTS'nin diğer yöntemlere göre en az 18 kat daha hızlı olduğu tespit edilmiştir.
3. Problemin karmaşıklığı arttıkça diğer yöntemlere göre IRMTS'nin başarı yüzdesi artmaktadır.
4. Önerilen çok amaçlı hibrit paralel yöntemin diğer yöntemlere göre %40 oranında sonuçları iyileştirdiği buna karşı hiçbir senaryoda domine olmadığı gözlemlenmiştir.
5. Önerilen hibrit yöntemin çok amaçlı problemlerin çözümünde arama uzayının daha etkin araştırılması amacıyla kullanılabileceği gözlemlenmiştir.
6. Önerilen Hibrit yöntemin hesaplama maliyeti yönünden NSGA2 ve SPEA2'ye göre dezavantajlı olduğu gözlemlenmiştir.

ÖNERİLER

1. Önerilen IRMTS'nin özellikle bulut sistemlerde etkin olduğu gözlemlendikten sonra bu yöntemin Mobil Kenar (Mobil Edge), Kenar (Edge) ve Sis (Fog) sistemlerde kullanılmasının performans artışı sağlayacağı düşünülmektedir.
2. Önerilen hibrit yöntemin başarısının artması için başka yöntemlerin de sürece dahil edilmesi elde edilen sonuçların çeşitliliğini arttırabilir.



KAYNAKLAR

- [1] F.Z. and M.N. aeem S. Rehman, N. Javaid, S. Rasheed, K. Hassan, Min-min scheduling algorithm for efficient resource distribution using Cloud and fog in smart buildings, (2018) 15–27.
- [2] N.A. Mehdi, A. Mamat, A. Amer, Z.T. Abdul-Mehdi, Minimum Completion Time for Power-Aware Scheduling in Cloud Computing, in: 2011 Dev. E-Systems Eng., IEEE, 2011: pp. 484–489. <https://doi.org/10.1109/DeSE.2011.30>.
- [3] P. Pradhan, P.K. Behera, B.N.B. Ray, Modified Round Robin Algorithm for Resource Allocation in Cloud Computing, *Procedia Comput. Sci.* 85 (2016) 878–890. <https://doi.org/10.1016/j.procs.2016.05.278>.
- [4] M.A. Alworafi, A. Dhari, A.A. Al-Hashmi, A.B. Darem, Suresha, An improved SJF scheduling algorithm in cloud computing environment, 2016 Int. Conf. Electr. Electron. Commun. Comput. Optim. Tech. ICEECCOT 2016 (2017) 208–212. <https://doi.org/10.1109/ICEECCOT.2016.7955216>.
- [5] Y. Pachipala, K.S. Sureddy, A.B.S. Sriya Kaitepalli, N. Pagadala, S.S. Nalabothu, M. Iniganti, Optimizing Task Scheduling in Cloud Computing: An Enhanced Shortest Job First Algorithm, *Procedia Comput. Sci.* 233 (2024) 604–613. <https://doi.org/10.1016/j.procs.2024.03.250>.
- [6] D.A. Agarwal, S. Jain, Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment, *Int. J. Comput. Trends Technol.* 9 (2014) 344–349. <https://doi.org/10.14445/22312803/ijctt-v9p163>.
- [7] J. Yang, B. Jiang, Z. Lv, K.-K.R. Choo, A task scheduling algorithm considering game theory designed for energy management in cloud computing, *Futur. Gener. Comput. Syst.* 105 (2020) 985–992. <https://doi.org/10.1016/j.future.2017.03.024>.
- [8] R. Duan, R. Prodan, X. Li, Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds, *IEEE Trans. Cloud Comput.* 2 (2014) 29–42. <https://doi.org/10.1109/TCC.2014.2303077>.
- [9] K.K.R. Yang, J., Jiang, B., Lv, Z., & Choo, A task scheduling algorithm considering game theory designed for energy management in cloud computing, *Futur. Gener. Comput. Syst.* (2020) 985–992.
- [10] I. Attiya, M. Abd Elaziz, I. Issawi, An improved hunger game search optimizer based IoT task scheduling in cloud-fog computing, *Internet of Things (Netherlands)* 26 (2024) 101196. <https://doi.org/10.1016/j.iot.2024.101196>.
- [11] S. Sheng, P. Chen, Z. Chen, L. Wu, Y. Yao, Deep Reinforcement Learning-Based Task Scheduling in IoT Edge Computing, *Sensors* 21 (2021) 1666. <https://doi.org/10.3390/s21051666>.
- [12] P. Song, C. Chi, K. Ji, Z. Liu, F. Zhang, S. Zhang, D. Qiu, X. Wan, A Deep Reinforcement Learning-based Task Scheduling Algorithm for Energy Efficiency in Data Centers, *Proc. - Int. Conf. Comput. Commun. Networks, ICCCN* 2021-July (2021) 1–9. <https://doi.org/10.1109/ICCCN52240.2021.9522309>.
- [13] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, J. Zeng, Q-learning based dynamic task scheduling for energy-efficient cloud computing, *Futur. Gener. Comput. Syst.* 108 (2020) 361–371. <https://doi.org/10.1016/j.future.2020.02.018>.
- [14] X. Chen, D. Long, Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm, *Cluster Comput.* 22 (2019) 2761–2769. <https://doi.org/10.1007/s10586-017-1479-y>.
- [15] C.W. Tsai, W.C. Huang, M.H. Chiang, M.C. Chiang, C.S. Yang, A hyper-heuristic scheduling algorithm for cloud, *IEEE Trans. Cloud Comput.* 2 (2014) 236–250. <https://doi.org/10.1109/TCC.2014.2315797>.
- [16] S.E. Shukri, R. Al-Sayyed, A. Hudaib, S. Mirjalili, Enhanced multi-verse optimizer for task scheduling in cloud computing environments, *Expert Syst. Appl.* 168 (2021) 114230. <https://doi.org/10.1016/j.eswa.2020.114230>.

- [17] G. Agarwal, S. Gupta, R. Ahuja, A.K. Rai, Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog–cloud computing, *Knowledge-Based Syst.* 272 (2023) 110563. <https://doi.org/10.1016/j.knosys.2023.110563>.
- [18] P. Gupta, P. Singh Rawat, D. kumar Saini, A. Vidyarthi, M. Alharbi, Neural network inspired differential evolution based task scheduling for cloud infrastructure, *Alexandria Eng. J.* 73 (2023) 217–230. <https://doi.org/10.1016/j.aej.2023.04.032>.
- [19] P. Krishnadoss, V.K. Poornachary, P. Krishnamoorthy, L. Shanmugam, Improvised Seagull Optimization Algorithm for Scheduling Tasks in Heterogeneous Cloud Environment, *Comput. Mater. Contin.* 74 (2023) 2461–2478. <https://doi.org/10.32604/cmc.2023.031614>.
- [20] A. Talha, A. Bouayad, M.O.C. Malki, An improved pathfinder algorithm using opposition-based learning for tasks scheduling in cloud environment, *J. Comput. Sci.* 64 (2022) 101873. <https://doi.org/10.1016/j.jocs.2022.101873>.
- [21] S.M. Hussain, G.R. Begh, Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment, *J. Comput. Sci.* 64 (2022) 101828. <https://doi.org/10.1016/j.jocs.2022.101828>.
- [22] N. Mansouri, B. Mohammad Hasani Zade, M.M. Javidi, Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory, *Comput. Ind. Eng.* 130 (2019) 597–633. <https://doi.org/10.1016/j.cie.2019.03.006>.
- [23] S. Velliangiri, P. Karthikeyan, V.M. Arul Xavier, D. Baswaraj, Hybrid electro search with genetic algorithm for task scheduling in cloud computing, *Ain Shams Eng. J.* 12 (2021) 631–639. <https://doi.org/10.1016/j.asej.2020.07.003>.
- [24] L. Abualigah, A.M.A. Hussein, M.H. Almomani, R.A. Zitar, H. Migdady, A.I. Alzahrani, A. Alwadain, Improved synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing, *Sustain. Comput. Informatics Syst.* 43 (2024) 101012. <https://doi.org/10.1016/j.suscom.2024.101012>.
- [25] P. Sathishkumar, N. Kumar, S.H. Raju, D. Rosy Salomi Victoria, An Intelligent Task Scheduling Approach for the Enhancement of Collaborative Learning in Cloud Computing, *Sustain. Comput. Informatics Syst.* 43 (2024) 101024. <https://doi.org/10.1016/j.suscom.2024.101024>.
- [26] A.Y. Hamed, M.K. Elnahary, F.S. Alsubaei, H.H. El-Sayed, Optimization Task Scheduling Using Cooperation Search Algorithm for Heterogeneous Cloud Computing Systems, *Comput. Mater. Contin.* 74 (2023) 2133–2148. <https://doi.org/10.32604/cmc.2023.032215>.
- [27] P. Gupta, A. Anand, P. Agarwal, G. McArdle, Neural network inspired efficient scalable task scheduling for cloud infrastructure, *Internet Things Cyber-Physical Syst.* 4 (2024) 268–279. <https://doi.org/10.1016/j.iotcps.2024.02.002>.
- [28] C.-Y. Liu, C.-M. Zou, P. Wu, A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing, in: 2014 13th Int. Symp. Distrib. Comput. Appl. to Business, Eng. Sci., IEEE, 2014: pp. 68–72. <https://doi.org/10.1109/DCABES.2014.18>.
- [29] X. Huang, Y. Lin, Z. Zhang, X. Guo, S. Su, A gradient-based optimization approach for task scheduling problem in cloud computing, *Cluster Comput.* 25 (2022) 3481–3497. <https://doi.org/10.1007/s10586-022-03580-9>.
- [30] S. Nabi, M. Ahmad, M. Ibrahim, H. Hamam, AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing, *Sensors* 22 (2022) 1–22. <https://doi.org/10.3390/s22030920>.
- [31] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, M. Zivkovic, Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer, 27th Telecommun. Forum, TELFOR 2019 (2019). <https://doi.org/10.1109/TELFOR48224.2019.8971223>.
- [32] S. Mangalampalli, G.R. Karri, A.A. Elngar, An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization, *Sensors* 23 (2023). <https://doi.org/10.3390/s23031384>.
- [33] S. Asghari, N.J. Navimipour, Cloud service composition using an inverted ant colony optimisation algorithm, *Int. J. Bio-Inspired Comput.* 13 (2019) 257. <https://doi.org/10.1504/IJBIC.2019.100139>.

- [34] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, J. Murphy, A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems, *IEEE Syst. J.* 14 (2020) 3117–3128. <https://doi.org/10.1109/JSYST.2019.2960088>.
- [35] M. Bürkük, G. Yıldırım, Cloneable Jellyfish Search Optimizer Based Task Scheduling in Cloud Environments, *Türk Doğa ve Fen Derg.* 11 (2022) 35–43. <https://doi.org/10.46810/tdfd.1123962>.
- [36] C. Chandrashekar, P. Krishnadoss, V.K. Poornachary, B. Ananthakrishnan, MCWOA Scheduler: Modified Chimp-Whale Optimization Algorithm for Task Scheduling in Cloud Computing, *Comput. Mater. Contin.* 78 (2024) 2593–2616. <https://doi.org/10.32604/cmc.2024.046304>.
- [37] M.S. Sanaj, P.M. Joe Prathap, Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere, *Eng. Sci. Technol. an Int. J.* 23 (2020) 891–902. <https://doi.org/10.1016/j.jestch.2019.11.002>.
- [38] S.K. Panda, S.S. Nanda, S.K. Bhoi, A pair-based task scheduling algorithm for cloud computing environment, *J. King Saud Univ. - Comput. Inf. Sci.* 34 (2022) 1434–1445. <https://doi.org/10.1016/j.jksuci.2018.10.001>.
- [39] D. Wu, Cloud computing task scheduling policy based on improved particle swarm optimization, *Proc. - 2018 Int. Conf. Virtual Real. Intell. Syst. ICVRIS 2018* (2018) 99–101. <https://doi.org/10.1109/ICVRIS.2018.00032>.
- [40] S. Zhan, H. Huo, Improved PSO-based task scheduling algorithm in cloud computing, *J. Inf. Comput. Sci.* 9 (2012) 3821–3829.
- [41] A. Al-Maamari, F.A. Omara, Task scheduling using PSO algorithm in cloud computing environments, *Int. J. Grid Distrib. Comput.* 8 (2015) 245–256. <https://doi.org/10.14257/ijgdc.2015.8.5.24>.
- [42] C. Chandrashekar, P. Krishnadoss, Opposition based sunflower optimization algorithm using cloud computing environments, *Mater. Today Proc.* 62 (2022) 4896–4902. <https://doi.org/10.1016/j.matpr.2022.03.534>.
- [43] X. Tang, C. Shi, T. Deng, Z. Wu, L. Yang, Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems, *Appl. Soft Comput.* 113 (2021) 107914. <https://doi.org/10.1016/j.asoc.2021.107914>.
- [44] H. Liu, Research on cloud computing adaptive task scheduling based on ant colony algorithm, *Optik (Stuttg.)* 258 (2022) 168677. <https://doi.org/10.1016/j.ijleo.2022.168677>.
- [45] H. Emami, Cloud task scheduling using enhanced sunflower optimization algorithm, *ICT Express* 8 (2022) 97–100. <https://doi.org/10.1016/j.ict.2021.08.001>.
- [46] T. Jena, J.R. Mohanty, GA-Based Customer-Conscious Resource Allocation and Task Scheduling in Multi-cloud Computing, *Arab. J. Sci. Eng.* 43 (2018) 4115–4130. <https://doi.org/10.1007/s13369-017-2766-x>.
- [47] Y. Fang, X. Xiao, J. Ge, Cloud computing task scheduling algorithm based on improved genetic algorithm, *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019* (2019) 852–856. <https://doi.org/10.1109/ITNEC.2019.8728996>.
- [48] M. Sharma, R. Garg, HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers, *Eng. Sci. Technol. an Int. J.* 23 (2020) 211–224. <https://doi.org/10.1016/j.jestch.2019.03.009>.
- [49] M. Agarwal, S. Gupta, An Adaptive Genetic Algorithm-Based Load Balancing-Aware Task Scheduling Technique for Cloud Computing, *Comput. Mater. Contin.* 73 (2022) 6103–6119. <https://doi.org/10.32604/cmc.2022.030778>.
- [50] M.A. Abu-Hashem, M. Shehab, M.K.Y. Shambour, M.S. Daoud, L. Abualigah, Improved Black Widow Optimization: An investigation into enhancing cloud task scheduling efficiency, *Sustain. Comput. Informatics Syst.* 41 (2024) 100949. <https://doi.org/10.1016/j.suscom.2023.100949>.
- [51] T. Saravanan, K. Ramesh, A Bio-inspired Energy Efficient Dynamic Task Scheduling (BEDTS) scheme and classification for virtualization CDC, *J. Eng. Res.* (2024). <https://doi.org/10.1016/j.jer.2023.08.026>.

- [52] S. Kaur, A. Verma, An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment, *Int. J. Inf. Technol. Comput. Sci.* 4 (2012) 74–79. <https://doi.org/10.5815/ijitcs.2012.10.09>.
- [53] H. Materwala, L. Ismail, Performance and energy-aware bi-objective tasks scheduling for cloud data centers, *Procedia Comput. Sci.* 197 (2021) 238–246. <https://doi.org/10.1016/j.procs.2021.12.137>.
- [54] Z. Zhang, M. Zhao, H. Wang, Z. Cui, W. Zhang, An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty, *Inf. Sci. (Ny)*. 583 (2022) 56–72. <https://doi.org/10.1016/j.ins.2021.11.027>.
- [55] X. Wang, H. Lou, Z. Dong, C. Yu, R. Lu, Decomposition-based multi-objective evolutionary algorithm for virtual machine and task joint scheduling of cloud computing in data space, *Swarm Evol. Comput.* 77 (2023) 101230. <https://doi.org/10.1016/j.swevo.2023.101230>.
- [56] N. Sharma, Sonal, P. Garg, Ant colony based optimization model for QoS-based task scheduling in cloud computing environment, *Meas. Sensors* 24 (2022). <https://doi.org/10.1016/j.measen.2022.100531>.
- [57] P. Tamilarasu, G. Singaravel, Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment, *J. Eng. Res.* (2024). <https://doi.org/10.1016/j.jer.2023.09.024>.
- [58] L. Imene, S. Sihem, K. Okba, B. Mohamed, A third generation genetic algorithm NSGAIII for task scheduling in cloud computing, *J. King Saud Univ. - Comput. Inf. Sci.* 34 (2022) 7515–7529. <https://doi.org/10.1016/j.jksuci.2022.03.017>.
- [59] S.M. Hosseini, M. Hosseini Shirvani, H. Motameni, Multi-objective discrete Cuckoo search algorithm for optimization of bag-of-tasks scheduling in fog computing environment, *Comput. Electr. Eng.* 119 (2024) 109480. <https://doi.org/10.1016/j.compeleceng.2024.109480>.
- [60] M. Abdullahi, M.A. Ngadi, S.I. Dishing, S.M. Abdulhamid, B.I. eel Ahmad, An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment, *J. Netw. Comput. Appl.* 133 (2019) 60–74. <https://doi.org/10.1016/j.jnca.2019.02.005>.
- [61] F. Ramezani, J. Lu, F. Hussain, Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization, *Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* 8274 LNCS (2013) 237–251. https://doi.org/10.1007/978-3-642-45005-1_17.
- [62] E.S. Alkayal, N.R. Jennings, M.F. Abulkhair, Efficient Task Scheduling Multi-Objective Particle Swarm Optimization in Cloud Computing, *Proc. - Conf. Local Comput. Networks, LCN* (2016) 17–24. <https://doi.org/10.1109/LCN.2016.024>.
- [63] S. Mangalampalli, G.R. Karri, U. Kose, Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization, *J. King Saud Univ. - Comput. Inf. Sci.* 35 (2023) 791–809. <https://doi.org/10.1016/j.jksuci.2023.01.016>.
- [64] M.A. Khan, R. ur Rasool, A multi-objective grey-wolf optimization based approach for scheduling on cloud platforms, *J. Parallel Distrib. Comput.* 187 (2024) 104847. <https://doi.org/10.1016/j.jpdc.2024.104847>.
- [65] L. Zuo, L. Shu, S. Dong, C. Zhu, T. Hara, A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, *IEEE Access* 3 (2015) 2687–2699. <https://doi.org/10.1109/ACCESS.2015.2508940>.
- [66] A. Got, A. Moussaoui, D. Zouache, A guided population archive whale optimization algorithm for solving multiobjective optimization problems, *Expert Syst. Appl.* 141 (2020) 112972. <https://doi.org/10.1016/j.eswa.2019.112972>.
- [67] B. Chen, W. Zeng, Y. Lin, D. Zhang, A new local search-based multiobjective optimization algorithm, *IEEE Trans. Evol. Comput.* 19 (2015) 50–73. <https://doi.org/10.1109/TEVC.2014.2301794>.
- [68] P. Pabitha, K. Nivitha, C. Gunavathi, B. Panjavarnam, A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing, *Sustain. Comput.*

- Informatics Syst. 41 (2024) 100944. <https://doi.org/10.1016/j.suscom.2023.100944>.
- [69] I. Behera, S. Sobhanayak, Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach, *J. Parallel Distrib. Comput.* 183 (2024) 104766. <https://doi.org/10.1016/j.jpdc.2023.104766>.
 - [70] K. Dubey, S.C. Sharma, A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing, *Sustain. Comput. Informatics Syst.* 32 (2021). <https://doi.org/10.1016/j.suscom.2021.100605>.
 - [71] P. Suresh, P. Keerthika, R. Manjula Devi, G.K. Kamalam, K. Logeswaran, K.K. Sadasivuni, K. Devendran, Optimized task scheduling approach with fault tolerant load balancing using multi-objective cat swarm optimization for multi-cloud environment, *Appl. Soft Comput.* 165 (2024) 112129. <https://doi.org/10.1016/j.asoc.2024.112129>.
 - [72] M. Zhao, Z. Zhang, T. Fan, W. Guo, Z. Cui, Many-Objective Optimization-Based Task Scheduling in Hybrid Cloud Environments, *C. - Comput. Model. Eng. Sci.* 136 (2023) 2425–2450. <https://doi.org/10.32604/cmes.2023.026671>.
 - [73] S. Midya, A. Roy, K. Majumder, S. Phadikar, Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach, *J. Netw. Comput. Appl.* 103 (2018) 58–84. <https://doi.org/10.1016/j.jnca.2017.11.016>.
 - [74] Y.J. Zheng, H.F. Ling, J.Y. Xue, Disaster rescue task scheduling: An evolutionary multiobjective optimization approach, *IEEE Trans. Emerg. Top. Comput.* 6 (2018) 288–300. <https://doi.org/10.1109/TETC.2014.2369957>.
 - [75] F. Ramezani, J. Lu, J. Taheri, F.K. Hussain, Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments, *World Wide Web* 18 (2015) 1737–1757. <https://doi.org/10.1007/s11280-015-0335-3>.
 - [76] X. Guo, Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm, *Alexandria Eng. J.* 60 (2021) 5603–5609. <https://doi.org/10.1016/j.aej.2021.04.051>.
 - [77] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, G. Fortino, Agent-based Internet of Things: State-of-the-art and research challenges, *Futur. Gener. Comput. Syst.* 102 (2020) 1038–1053. <https://doi.org/10.1016/j.future.2019.09.016>.
 - [78] F. De la Prieta, S. Rodríguez-González, P. Chamoso, J.M. Corchado, J. Bajo, Survey of agent-based cloud computing applications, *Futur. Gener. Comput. Syst.* 100 (2019) 223–236. <https://doi.org/10.1016/j.future.2019.04.037>.
 - [79] R. Kumar, M.F. Hassan, M.H.M. Adnan, S. Shukla, S. Safdar, M.A. Qureshi, A.H. Abdel-Aty, A user-priorities-based strategy for three-phase intelligent recommendation and negotiating agents for cloud services, *IEEE Access* 11 (2023) 26932–26944. <https://doi.org/10.1109/ACCESS.2023.3254552>.
 - [80] M. Rimol, Gartner Says Four Trends Are Shaping the Future of Public Cloud, Gartner (2021). <https://www.gartner.com/en/newsroom/press-releases/2021-08-02-gartner-says-four-trends-are-shaping-the-future-of-public-cloud>.
 - [81] M.I. Alam, M. Pandey, S.S. Rautaray, A Comprehensive Survey on Cloud Computing, *Int. J. Inf. Technol. Comput. Sci.* 7 (2015) 68–79. <https://doi.org/10.5815/ijitcs.2015.02.09>.
 - [82] R.M. Singh, S. Paul, A. Kumar, Task Scheduling in Cloud Computing : Review, 5 (2014) 7940–7944.
 - [83] S. Bharany, S. Sharma, O.I. Khalaf, G.M. Abdulsahib, A.S. Al Humaimeedy, T.H.H. Aldhyani, M. Maashi, H. Alkahtani, A Systematic Survey on Energy-Efficient Techniques in Sustainable Cloud Computing, *Sustain.* 14 (2022) 1–89. <https://doi.org/10.3390/su14106256>.
 - [84] □ Optimizasyon, T. Kavramlar, Y. Doç, Y. Volkan, Matlab UygulamalarMühendislikÇözümleri, (n.d.).
 - [85] N. Gunantara, A review of multi-objective optimization: Methods and its applications, *Cogent Eng.* 5 (2018) 1–16. <https://doi.org/10.1080/23311916.2018.1502242>.
 - [86] V. Tomar, M. Bansal, P. Singh, Metaheuristic Algorithms for Optimization: A Brief Review, *Eng.*

- Proc. 59 (2023) 1–16. <https://doi.org/10.3390/engproc2023059238>.
- [87] M. Ulucan, G. Yildirim, B. Alatas, K.E. Alyamac, A new intelligent sunflower optimization based explainable artificial intelligence approach for early-age concrete compressive strength classification and mixture design of RAC, *Struct. Concr.* (2023) 1–19. <https://doi.org/10.1002/suco.202300138>.
- [88] R.H. Stewart, T.S. Palmer, B. DuPont, A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers, *Prog. Nucl. Energy* 138 (2021) 103830. <https://doi.org/10.1016/j.pnucene.2021.103830>.
- [89] K. Taha, Methods that optimize multi-objective problems: A survey and experimental evaluation, *IEEE Access* 8 (2020) 80855–80878. <https://doi.org/10.1109/ACCESS.2020.2989219>.
- [90] Q. Luo, G. Wu, B. Ji, L. Wang, P.N. Suganthan, Hybrid Multi-Objective Optimization Approach With Pareto Local Search for Collaborative Truck-Drone Routing Problems Considering Flexible Time Windows, *IEEE Trans. Intell. Transp. Syst.* 23 (2022) 13011–13025. <https://doi.org/10.1109/TITS.2021.3119080>.
- [91] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.* 6 (2002) 182–197. <https://doi.org/10.1109/4235.996017>.
- [92] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach, *IEEE Trans. Evol. Comput.* 3 (1999) 257–271. <https://doi.org/10.1109/4235.797969>.

ÖZGEÇMİŞ

Cebrail BARUT

[illegible]

AKADEMİK FAALİYETLER

A) Makaleler

1. C. Barut, G. Yildirim, and Y. Tatar, "An intelligent and interpretable rule-based metaheuristic approach to task scheduling in cloud systems," *KNOWLEDGE-BASED Syst.*, vol. 284, 2024, doi: 10.1016/j.knosys.2023.111241

B) Uluslararası Toplantıda Sunularak Tam Metin Olarak Yayınlanan Bildiriler

1. Barut Cebirail, Yıldırım Güngör, Tatar Yetkin (2021) Sis Hesaplamada Görev Dağıtımının Arşimet Optimizasyon Algoritması Ile Optimizasyonu, International Siirt Scientific Research Congress, Türkiye, (Kasım 2021)