

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

BULUT BİLİŞİMDE METASEZGİSEL GÖREV ÇİZELGELEMESİ

Merve Nur AKTAN

Danışman : Doç. Dr. Hasan BULUT

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Yüksek Lisans Programı

İzmir
2019

Merve Nur AKTAN tarafından YÜKSEK LİSANS tezi olarak sunulan “ **Bulut Bilişimde Metasezgisel Görev Çizelgelemesi**” başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve **30.07.2019** tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:

Jüri Başkanı : Doç. Dr. Hasan Bulut
Raportör Üye : Prof. Dr. Serdar Korukoğlu
Üye : Dr. Öğr. Üyesi Hüseyin Abacı

İmza

.....
.....
.....



EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca **Yüksek Lisans Tezi** olarak sunduğum “**Bulut Bilişimde Metasezgisel Görev Çizelgelemesi**” başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

30 / 07 / 2019

İmzası

Adı-Soyadı

Merve Nur AKTAN

ÖZET**BULUT BİLİŞİMDE METASEZGİSEL
GÖREV ÇİZELGELEMESİ**

AKTAN, Merve Nur

Yüksek Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Doç. Dr. Hasan BULUT

Temmuz 2019, 86 sayfa

Son yıllarda yüksek ölçeklenebilirliği, güvenilirliği, bilgi paylaşımı ve düşük maliyetinden dolayı bulut bilişim dikkat çekmektedir. Bulut bilişim esnek, yüksek performanslı, kullanılan kadar ödemeli, isteğe bağlı bir hizmet sunma avantajına sahiptir. Kullanıcıların memnuniyeti için, bulut hizmet sağlayıcısı Servis Seviyesi Anlaşmasına sadık kalmalıdır. İş yayılımı çok uzun olmamalı ve kaynak kullanımını artırmak için sunucular arasında yükün dinamik olarak dengelemesi gereklidir. Bu nedenle, görevleri dinamik ve etkin bir şekilde planlamak çözülmesi gereken kritik bir sorundur. Bu tez çalışmasında, bulut ortamında çalışan görevler için meta-sezgisel algoritmalara dayalı görev çizelgeleme algoritmaları geliştirilmiştir. Geliştirilen yaklaşımlar ile bulut ortamındaki görevler için dinamik ve statik görev çizelgeleme teknikleri kullanılarak, makineler üzerindeki bağımsız görevlerin yük dengelemesinin ve işlemin tamamlanma zamanının iyileştirilmesi amaçlanmıştır. Bu yaklaşımlar, literatürde bilinen yöntemlerle karşılaştırılmıştır. Çeşitli durumlarda, geliştirilen yaklaşımların karşılaştırılan yöntemlerden daha iyi sonuç verdiği görülmüştür.

Anahtar sözcükler: Bulut bilişim, Cloudsim, statik görev çizelgeleme, dinamik görev çizelgeleme, meta-sezgisel algoritmalar, genetik algoritma, tavlama benzetimi, açgözlü yaklaşım, diferansiyel gelişim algoritması.

ABSTRACT**METAHEURISTIC TASK SCHEDULING
IN CLOUD COMPUTING**

AKTAN, Merve Nur

MSc in Computer Eng.

Supervisor: Assoc. Prof. Dr. Hasan BULUT

July 2019, 86 pages

In recent years, cloud computing has attracted attention due to its high scalability, reliability, information sharing and low cost. Cloud computing has the advantage of providing a flexible, high-performance, pay as you use and on-demand service. The cloud service provider must adhere to the Service Level Agreement to provide user satisfaction. Makespan should not be too long and dynamic load balancing between servers is required to increase resource usage. Therefore, scheduling tasks dynamically and effectively is a critical issue that needs to be solved. In this thesis, task scheduling algorithms based on meta-heuristic algorithms, have been developed for tasks running in cloud environment. In this thesis, it is aimed to improve the load balancing of the independent tasks on the machines and to reduce the completion time of the process by using dynamic and static task scheduling techniques for the tasks in cloud. These approaches have been compared with the methods known in the literature. In various cases, it has been found that the developed approaches outperformed than the compared methods.

Keywords: Cloud computing, Cloudsim, static task scheduling, dynamic task scheduling, meta heuristic algorithms, genetic algorithm, simulated annealing, greedy approach, differential evolution algorithm.

ÖNSÖZ

Bu tez çalışmasında yaygın olarak bilinen görevlerin kaynaklara atanması problemi, günümüzde kullanımı günden güne artan bulut ortamındaki görevler için ele alınmıştır. Tez danışmanım Doç. Dr. Hasan BULUT' un bilgi birikimi ve yol göstericiliği ile yapılan literatür araştırmalarında ve tez çalışmasında hem kullanıcı için hem de hizmet sağlayıcı için iyileştirilmesi gereken ölçütlere karar verilerek bu ölçütleri iyileştirecek yaklaşımlar önerilmiştir. Tez konusunun güncel teknolojiyle ilgili olması teze olan ilgimin artmasını ve merakımın devamlı olmasını sağlamıştır.

Akademik bir çalışmanın nasıl yapılması gerektiğini deneyimlediğim, tez konusunun belirlenmesinden tezin tamamlanmasına kadar geçen bu süreç boyunca her aşamada desteklerini esirgemeyen tez danışmanım Doç. Dr. Hasan BULUT'a sonsuz teşekkürlerimi sunarım.

İZMİR

Merve Nur AKTAN

30.07.2019

İÇİNDEKİLERSayfa

İÇ KAPAK	i
KABUL ONAY SAYFASI	iii
ETİK KURALLARA UYGUNLUK BEYANI.....	v
ÖZET	vii
ABSTRACT.....	ix
ÖNSÖZ	xi
İÇİNDEKİLER DİZİNİ.....	xiii
ŞEKİLLER DİZİNİ	xvii
ÇİZELGELER DİZİNİ.....	xxiii
SİMGELER VE KISALTMALAR DİZİNİ	xxv
1. GİRİŞ.....	1
2. TEMEL KAVRAMLAR VE İLGİLİ ÇALIŞMALAR	6
2.1 Bulut Bilişim.....	6
2.2 Sanallaştırma	12
2.2.1 Sanallaştırma uygulama seviyeleri	13
2.3 Bulut Mimarisi.....	16

İÇİNDEKİLER (devam)

Sayfa

2.4 Görev Çizelgeleme.....	17
2.4.1 Problem tanımı.....	18
2.4.2 Çizelgeleme algoritmaları	19
2.5 Simülasyon Ortamı: Cloudsim.....	20
2.5.1 Cloudsim çalışma mimarisi.....	20
2.5.2 Birimler arası iletişim.....	22
2.6 İlgili Çalışmalar.....	22
3. GELİŞTİRİLEN YAKLAŞIMLAR.....	27
3.1 Kromozom Yapısı.....	27
3.2 Kayan Pencere Tekniği	28
3.3 Maxspan, Ortalama Kaynak Kullanımı, Kabul Edilebilir Kuyruk Sayısı ve Uygunluk Fonksiyonu.....	29
3.4 Geliştirilen Algoritmalar	32
3.4.1 Genetik algoritma.....	32
3.4.2 Tavlama benzetimi	36
3.4.3 Açgözlü yaklaşım.....	37
3.4.4 Diferansiyel gelişim algoritması	38

İÇİNDEKİLER (devam)

	<u>Sayfa</u>
3.4.5 SAGA.....	41
3.4.6 GASA.....	41
3.4.7 DESA.....	41
3.5 Statik Çizelgeleme Adımları.....	42
3.6 Dinamik Çizelgeleme Adımları.....	46
4. DEĞERLENDİRME ÖLÇÜTLERİ VE KULLANILAN VERİ SETLERİ.....	48
4.1 Değerlendirme Ölçütleri	48
4.1.1 Tamamlanma zamanı.....	48
4.1.2 Minimum tamamlanma zamanı	49
4.1.3 Sanal makine yüklerinin standart sapması.....	49
4.1.4 Sanal makine yüklerinin varyansı.....	49
4.2 Kullanılan Verisetleri.....	49
5 DENEYSEL SONUÇLAR	51
5.1 Parametreler.....	51
5.1.1 Sistem parametreleri	51
5.1.2 Algoritma parametreleri.....	51

İÇİNDEKİLER (devam)

Sayfa

5.2 Algoritmalarla Göre Ölçütler Üzerinden Statik Çizelgeleme Sonuçları.....	53
5.2.1 Tamamlanma zamanı	53
5.2.2 Minimum tamamlanma zamanı.....	57
5.2.3 Sanal makine yüklerinin standart sapması.....	60
5.2.4 Sanal makine yüklerinin varyansı	63
5.3 Algoritmalarla Göre Ölçütler Üzerinden Dinamik Çizelgeleme Sonuçları	65
5.3.1 Tamamlanma zamanı	65
5.3.2 Minimum tamamlanma zamanı	68
5.3.3 Sanal makine yüklerinin standart sapması	71
5.3.4 Sanal makine yüklerinin varyansı	74
6. SONUÇLAR	78
KAYNAKLAR DİZİNİ	80
TEŞEKKÜR.....	85
ÖZGEÇMİŞ	86
EKLER.....	

ŞEKİLLER DİZİNİ

<u>Şekil</u>	<u>Sayfa</u>
2.1	Bulut bilişimde servis modelleri..... 10
2.2	Sanallaştırmadan önce ve sonraki bilgisayar sistem mimarisi 14
2.3	Beş soyutlama düzeyinde donanımdan uygulamalara değişen sanallaştırma 14
2.4	Yüksek seviye bulut mimarisi 17
2.5	Bulut bilişim ortamındaki görev çizelgeleme mimarisi 18
2.6	Cloudsim çalışma şekli..... 21
2.7	Cloudsim veri akışı..... 22
3.1	Geliştirilen kromozom yapısı 28
3.2	Kaynaklara atanacak görevler ve yükleri 30
3.3	Formül 3.2' ye göre hesaplanan anlık sistem yükleri ve maxspan..... 30
3.4	Uniform order based crossover 35
3.5	Order based crossover 35
3.6	Tezdeki çaprazlama 35
3.7	SA sözdekod..... 37
3.8	DE' nin adımları..... 39
3.9	DE'de dönüşüm işlemleri..... 40

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
3.10 SAGA işleyişi	42
3.11 GASA işleyişi	42
3.12 DESA işleyişi.....	42
3.13 Statik görev çizelgeleme akış şeması.....	45
3.14 Dinamik görev çizelgeleme akış şeması.....	47
5.1 50 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	54
5.2 100 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	54
5.3 200 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	55
5.4 300 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	55
5.5 400 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	56
5.6 500 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	56
5.7 50 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	57
5.8 100 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	58

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
5.9 200 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği	58
5.10 300 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği	59
5.11 400 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği	59
5.12 500 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği	60
5.13 Statik çizelgelemede tüm görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin standart sapma değerleri grafiği.....	62
5.14 Statik çizelgelemede tüm görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği.....	64
5.15 50 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	66
5.16 100 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	66
5.17 200 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	67
5.18 300 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	67
5.19 400 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	67

ŞEKİLLER DİZİNİ (devam)

<u>Şekil</u>	<u>Sayfa</u>
5.20 500 adet görev üzerinden algoritmaların tamamlanma zamanı değerleri grafiği	68
5.21 50 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği	69
5.22 100 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	69
5.23 200 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	70
5.24 300 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	70
5.25 400 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	71
5.26 500 adet görev üzerinden algoritmaların minimum tamamlanma zamanı değerleri grafiği.....	71
5.27 Dinamik çizelgelemede tüm görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin standart sapma değerleri grafiği.....	73
5.28 50 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	74
5.29 100 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	75
5.30 200 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	75

ŞEKİLLER DİZİNİ (devam)ŞekilSayfa

5.31	300 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	76
5.32	400 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	76
5.33	500 adet görev üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri grafiği	77

ÇİZELGELER DİZİNİ

<u>Çizelge</u>	<u>Sayfa</u>
5.1 Simülasyon parametreleri	52
5.2 Algoritma parametreleri.....	52
5.3 Değişen görev sayıları üzerinden algoritmalara göre tamamlanma zamanı değerleri tablosu	53
5.4 Değişen görev sayıları üzerinden algoritmalara göre minimum tamamlanma zamanı değerleri tablosu	57
5.5 Değişen görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin standart sapma değerleri tablosu.....	61
5.6 Değişen görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri tablosu	63
5.7 Değişen görev sayıları üzerinden algoritmalara göre tamamlanma zamanı değerleri tablosu	65
5.8 Değişen görev sayıları üzerinden algoritmalara göre minimum tamamlanma zamanı değerleri tablosu.....	68
5.9 Değişen görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin standart sapma değerleri tablosu.....	72
5.10 Değişen görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin varyans değerleri tablosu.....	74

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

API	(ing.) Application Program Interface (tr.) Uygulama Program Arayüzü
AVG	(ing.) Average (tr.) Ortalama
DE	(ing.) Differential Evolution (tr.)Diferansiyel Gelişim Algoritması
DESA	(ing.) Differential Evolution- Simulated Annealing (tr.)Diferansiyel Gelişim- Tavlama Benzetimi Yöntemi
GA	(ing.) Genetic Algorithm (tr.)Genetik Algoritma
GASA	(ing.) Genetic Algorithm- Simulated Annealing (tr.)Genetik Algoritma- Tavlama Benzetimi Yöntemi
MI	(ing.)Million Instruction (tr.) Milyon Komut
MIPS	(ing.) Million Instruction per Second (tr.) Saniyedeki Milyon Komut Sayısı
Min	(ing.) minimum (tr.) en az
QoS	(ing.) Quality of Service (tr.)Hizmet Kalitesi
SA	(ing.) Simulated Annealing (tr.)Tavlama Benzetimi

SAGA	(ing.) Simulated Annealing- Genetic Algorithm (tr.)Tavlama Benzetimi- Genetik Algoritma Yöntemi
SD	(ing.) Standard Deviation (tr.) Standart Sapma
SLA	(ing.) Service Level Agreement (tr.)Hizmet Düzeyi Anlaşması
VM	(ing.) Virtual Machine (tr.)Sanal Makine



1. GİRİŞ

Son yıllarda yüksek ölçeklenebilirliği, güvenilirliği, bilgi paylaşımı ve düşük maliyetinden dolayı dağıtık bilişim paradigması, tek işlemcili makinalara göre daha fazla dikkat toplamaktadır. Şimdiki duruma bakıldığında dağıtık bilişim paradigması içinde “bulut bilişim” öne çıkmaktadır (Kalra and Singh, 2015). Bulut bilişim, son kullanıcılar arasında dinamik olarak bilişim altyapısı ve diğer servisleri sağlayan bir teknolojidir. Geleneksel yüksek performanslı hesaplama sistemleri yerine bulutu kullanmanın bazı avantajları vardır, bunlar: (1) talep üzerine kaynak sağlanması, (2) kaynaklar, “kullandığın kadar öde” modeline göre paylaştırıldığından donanım için yapılan harcamanın azalması, (3) kullanıcı ve servis sağlayıcı arasında bir anlaşma olan Servis Seviyesi Anlaşması (SLA) nı takip ettiğinden artan kullanıcı memnuniyeti, (4) sanal makinaların kullanımıyla artan kaynak istifadesi. Tüm bu avantajlardan dolayı araştırmacılar bu konuya yönelmektedir (Singh et al., 2017).

Bu teknoloji son kullanıcılara dağıtılmış, sanallaştırılmış ve elastik kaynakları yardımcı servisler olarak sunmayı amaçlamaktadır. Bulut, yakın gelecekte “faydalı hizmet olarak bilişim” kavramını tam farkındalıkla destekleyecek potansiyele sahiptir. Sanallaştırma teknolojisinin desteği ile bulut platformları, işletmelere sanal makineler şeklindeki bilgisayar gücünü kullanıcılara kiralama imkânı veriyor. Bu kullanıcılar yüzlerce sanal makine kullanabildiklerinden bulutta görevleri kaynaklara el ile atamak zordur. Bu yüzden bulut sisteminde etkili bir görev çizelgeleme algoritmasına ihtiyaç vardır (Li et al., 2011).

Kalite, bir dizi doğal özelliğin gereksinimi karşılama derecesi olarak tanımlanır. Bir sistemin ya da uygulamanın kalitesini etkileyen birçok faktör bulunmaktadır bunlar: esneklik, sürdürülebilirlik ve okunabilirlik, performans ve verimlilik, ölçeklenebilirlik, uygunluk ve dayanıklılık, kullanılabilirlik ve kolayca ulaşılabilirlik, platform uyumluluğu, güvenlidir. “Esneklik”, yazılımın sistemi tahrip etmeden fonksiyonelliği yönetme yeteneğidir. “Sürdürülebilirlik” esneklikle benzerdir ancak hata düzeltimi ile ilgili değişikliklere odaklanır. “Performans”, yazılımın tepki süresiyle ilgilidir. “Ölçeklenebilir sistem” kullanıcının eylemlerine

makul zamanda tepki verir. “Dayanıklı yazılım” hata durumunda bile ulaşılabilir olmalıdır (Ramadan and Kashyap, 2017).

Bulut uygulamalarında ortaya çıkan zorluklardan biri performans ve uygunlukla ilgili olarak yüksek düzeyde hizmet sağlamak için kaynakları uygulamaya tahsis etme problemi olan servis kalitesi (QoS) yönetimidir. İnternet ağı kullanıcıları günden güne artmakta ve daha iyi performansla ulaşmak için ağ gereksinimi de artmaktadır. Bunun sonucunda birçok çevrimiçi servis, büyük bant genişliği ve ağ performansına ihtiyaç duymaktadır. Ağ performansındaki olumsuzluklar kullanıcıları ve servis sağlayıcılarını rahatsız eden bir unsurdur. İnternet servis sağlayıcıları, rakiplerinden önce en iyi hizmeti sunmak için yeni teknolojiler getirmelidir. Servis kalitesi, ağların maksimum ağ genişliği elde etme ve gecikme, hata oranı, çalışma zamanı gibi ağ unsurlarıyla başa çıkma yeteneği anlamına gelir. Servis kalitesi, belirli veri türlerine öncelikler atayarak diğer ağ kaynaklarının yönetimini içerir (Ramadan and Kashyap, 2017).

Bulut bilişim, esnek, yüksek performanslı, istediğiniz gibi ödemeli, isteğe bağlı bir hizmet sunma avantajına sahiptir. Operatörler abonelere garanti vermeli ve Servis Seviyesi Anlaşmasına (SLA) sadık kalmalıdır. İş yayılımı çok uzun olursa kesinlikle müşteri memnun edilemez. Ayrıca, bulut bilişim platformunun kaynak kullanımını artırmak için sunucular arasında yükü dinamik olarak dengelemesi gerekiyor. Bu nedenle, görevleri dinamik ve etkin bir şekilde planlamak ve abone sayısını karşılamak, çözülmesi gereken kritik bir sorundur (Wang et al., 2014).

Görev çizelgelemesi problemi, görevler için belirlenen hedeflerin başarılabilmesi için, kullanılabilir kaynak kümesi (işlemciler / bilgisayar makineleri) üzerine farklı görevleri en uygun şekilde haritalama / atama olarak görülebilir (Singh et al., 2014). Görev çizelgelemesi, bulut platformunda en kritik konulardan biridir. Kullanıcı sayısı çok fazla ve veri hacmi oldukça büyüktür. Kaynak paylaşımı ve yeniden kullanım talepleri gittikçe zorunlu hale gelmektedir. Etkin bir görev planlama mekanizması, kullanıcıların gereksinimlerini karşılamalı ve kaynak kullanımını geliştirmeli ve böylece bulut bilişim ortamının genel performansını yükseltmelidir (Wang et al., 2014; Wadhonkar and Theng, 2016).

Geliştirilecek tezde bulut bilişim ortamının genel performansının etkin bir görev çizelgeleme mekanizması geliştirilerek yükseltilmesi amaçlanmıştır. Bu konu hakkında yapılan çalışmalar incelendiğinde, çalışmalarda simülasyon ortamlarının kullanıldığı görülmüştür. Bu simülasyon ortamlarından en yaygın olarak kullanılanı CloudSim olarak ön plana çıkmıştır.

Bulut bilişim sistemi ağ akışı, yük dengeleme, ölçeklenebilirlik gibi sorunlarla karşılaştığından, deneysel amaçlar için bulut bilişim sistemini doğrudan kullanmak yerine iyi bir simülatör kullanılmalıdır. CloudSim, bulut bilişim altyapısı ve uygulama servislerine modelleme, simülasyon ve deney yapma olanağı veren genelleştirilmiş bir simülasyon ortamıdır (Babu and Krishna, 2013).

Genel olarak bulut platformda görevlerin sınırsız sayıda kaynaklara eşlenmesi problemi NP hard olarak bilinen problem kategorisine aittir. Böyle problemlerin çözümünde polinom zamanda optimum çözüm üreten algoritma yoktur. Tam kapsamlı arama (exhaustive search) ya dayalı çözümler, çizelgeleme oluşturmanın maliyeti fazla olduğu için çok uygulanabilir değildir. Meta-sezgisel tabanlı teknikler uygun zaman içinde optimale yakın sonuçlar sağlayarak bu problemlerle başa çıkabilmektedir. Meta-sezgisel teknikler büyük ve karmaşık problemleri çözme konusundaki etkinliği nedeniyle son yıllarda büyük popülerlik kazanmıştır (Kalra and Singh, 2015).

Günümüzde, görev çizelgeleme problemlerini çözmek için birçok sezgisel evrimsel optimizasyon algoritması önerilmiştir (Nzanywayingoma and Yang, 2017). Sezgisel yöntemlerle üretilen çözümler çoğunlukla yerel minima problemine takılmakta ve meta-sezgisel yöntemler bu durumdan kaçınmada en etkili yaklaşım olarak kanıtlandı. “Meta-sezgisel” terimi Glover ve Laguna tarafından sezgisel algoritmalar üzerinde bazı değişiklikler yapılarak yerel arama optimizasyonu bağlamında daha iyi çözümler sunmak için sunulmuştur. Bu yaklaşımlar, performans ve çalışma zamanı açısından gerçek hayattaki herhangi bir optimizasyon problemine uyarlanabilir. Çeşitli meta-sezgisel yaklaşımlar iki kategori altında incelenir. Canlılıktan ilham alan meta-sezgisel yaklaşımlar ve sürü zekâsı tabanlı meta-sezgisel yaklaşımlar (Singh et al., 2017).

Bir diğ er sınıflandırmayı da (Saleh et al., 2019) řu řekilde yapmıřtır; tek   z m tabanlı meta-sezgisel algoritmalar ve pop lasyon tabanlı meta-sezgisel algoritmalar. Tek   z ml ler Tepe Tırmanıřı, Tavlama Benzetimi, Tabu Araması gibi tek bir aday   z m  deđiřtirip geliřtirmeye odaklanırlar. Diğ er yandan pop lasyon tabanlılar ise  ok sayıda aday   z m bulundururlar. Bu algoritmalar aramada yol g stermesi i in pop lasyon  zelliklerini kullanırlar. Meta-sezgisel algoritmalar genellikle deterministik (rastgelelik i ermeyen) algoritmalara g re daha yavařtır. Bu y zden bazı arařtırma alanlarında meta-sezgisel algoritmanın  alıřma s resini hızlandırmak i in girdi verisini  n iřlem den ge irmek pop ler hale gelmiřtir. Meta-sezgisel algoritmalaradaki son  alıřmalar genellikle 3 temel amaca odaklanmıřtır. (1) uygunluk fonksiyonunu deđiřtirmek, (2) operat rleri deđiřtirmek, (3) melez meta-sezgisel kullanmak.

Bu tezde bulut ortamında  alıřan g revler i in meta-sezgisel algoritmalara dayalı g rev  izelgeleme algoritmaları geliřtirilmiřtir. Geliřtirilen yaklařımlar ile bulut ortamındaki g revler i in dinamik g rev  izelgeleme tekniđi kullanılarak, makinalar  zerindeki bađımsız g revlerin y k dengelemesinin ve iřlemin tamamlanma zamanının iyileřtirilmesi ama lanmıřtır. Ayrıca geliřtirilen yaklařımlar statik g rev  izelgeleme tekniđinde de kullanılarak  eřitli tespitlerde bulunulmuřtur. Her iki teknikte de a g z ly yaklařım, meta-sezgisel algoritmalar ve bunlardan se ilen bazı algoritmaların melez kullanılması sonucu elde edilen  ıktılar se ilen metrikler  zerinden deđerlendirilmiřtir.

Tez kapsamında kullanılan veri seti tek bi imli dađılım (uniform distribution) la oluřturulmuřtur. 50, 100, 200, 300, 400, 500 adet g rev den oluřan g rev gruplarının her birinden 20 řer adet oluřturulmuřtur.(50 g rev den oluřan 20 metin dosyası, 100 g rev den oluřan 20 metin dosyası...) Her bir algoritma, oluřturulan her g rev grubunun metin dosyasında 5 kez  alıřtırılmıřtır. Elde edilen sonu ların aritmetik ortalaması alınarak ilgili g rev grubunun sonucuna ulařılmıřtır. Meta sezgisel algoritmalaradaki rastgelelikten dolayı her  alıřmada farklı sonu  elde edildiđi i in b yle bir y ntem izlenmiřtir.

Bu tez  alıřmasının katkıları  zet ile: Literat rde mevcut statik  izelgeleme y ntemi  zerine GA ve SA adapte edilmiřtir ve Greedy tekniđi uygulanmıřtır. Yeni

geliştirilen DE, SAGA, GASA ve DESA uygulanmıştır. Yeni bir dinamik çizelgeleme yöntemi önerilmiştir. Bu dinamik çizelgeleme yöntemi üzerine, statik çizelgelemede olduğu gibi GA ve SA adapte edilmiştir ve Greedy tekniği uygulanmıştır. Yeni geliştirilen DE, SAGA, GASA ve DESA uygulanmıştır.

Bu tez çalışmasının ikinci bölümünde bulut bilişim, sanallaştırma, görev çizelgeleme kavramları detaylıca ele alınmıştır, çalışmada kullanılan simülasyon ortamı anlatılmıştır, literatürde bu yönde yapılan çalışmalar incelenerek değerlendirilmiştir.

Üçüncü bölümde tez kapsamında uygulanan genetik algoritma, tavlama benzetimi, açgözlü yaklaşım ve diferansiyel gelişim algoritması açıklanmış, eklenen değişiklikler belirtilmiştir, geliştirilen melez algoritmalar açıklanmıştır.

Dördüncü bölümde bulut bilişimde görev çizelgelemesi konusunda literatürde kullanılan değerlendirme ölçütleri açıklanmıştır ve bu tezde kullanılan değerlendirme ölçütleri belirtilmiştir. Buna ek olarak, çalışmada kullanılan veri setinin oluşturulması ve özellikleri anlatılmıştır.

Beşinci bölümde statik ve dinamik çizelgeleme türlerinde uygulanan deneylerin sonuçları, belirlenen değerlendirme ölçütleri üzerinden gösterildi. Uygulanan algoritmalar arasında sonuçları dikkate alarak karşılaştırmalar yapıldı.

Altıncı bölümünde çalışmada ortaya konan yaklaşımlar değerlendirilmiştir, katkıları belirtilmiştir.

2. TEMEL KAVRAMLAR VE İLGİLİ ÇALIŞMALAR

Bu bölümde, bulut bilişimde meta sezgisel görev çizelgelemesiyle ilgili temel kavramlar anlatılmıştır ve literatürde bu konuyla ilgili yapılan çalışmalar incelenmiştir.

2.1. Bulut Bilişim

Bulut bilişim, gelişen bir paradigmadır. Amerikan Ulusal Standart Ve Teknoloji Enstitüsü (NIST)' in bulut bilişim hakkındaki tanımına göre bulut modeli beş temel özellik, üç servis modeli ve dört dağıtım modelinden oluşur.

Bulut altyapısı ise bulut bilişimin beş temel özelliğini mümkün kılan yazılım ve donanım bütünüdür. Bulut altyapısı fiziksel katman ve soyutlama katmanından oluşur. Fiziksel katman: Sunucu, depolama ve ağ bileşenlerinden oluşan, sağlanan bulut hizmetlerini desteklemek için gerekli donanım kaynaklarından oluşur. Soyutlama katmanı: Temel bulut özelliklerini gösteren fiziksel katman boyunca dağıtılan yazılımdan oluşur. Kavramsal olarak soyutlama katmanı, fiziksel katmanın üzerine oturur.

Bulut bilişim, asgari çaba veya hizmet sağlayıcı etkileşimi ile hızlıca tedarik edilip geri bırakılabilen, yapılandırılabilir ve paylaşılabilir kaynaklar havuzuna (ağlar, sunucular, depolama, uygulamalar ve servisler) yaygın, elverişli, istenildiği zaman yapılan ağ erişimine olanak sağlayan bir modeldir. Beş temel özellikten oluşur, bunlar:

Talep üzerine self servis: Müşteri ihtiyacı oldukça otomatik olarak servis sağlayıcısıyla insan etkileşimine geçmesine gerek kalmadan sunucu zamanı ve ağ depolaması gibi bilişim özelliklerine tek yönlü olarak ulaşabilir.

Geniş ağ erişimi: Ağ üzerinde uygun olan özellikler ve bunlara heterojen istemci platformları (mobil telefon, tablet, laptop, iş istasyonu) tarafından kullanımı teşvik eden standart mekanizmalar yoluyla erişim.

Kaynak havuzu oluřturma: Saęlayıcının kaynakları, birçok müşteriye hizmet vermek için çoklu kiracı modelini kullanarak bir kaynak havuzunda toplanır, farklı fiziksel ve sanal kaynaklar dinamik olarak atanır ve müşteri talep ettikçe yeniden atanabilir. Bu durumda lokasyon baęımsızlığı hissi vardır. Müşterinin genellikle kendisine saęlanan kaynakların lokasyonu ile ilgili tam bir kontrolü veya bilgisi yoktur ancak, lokasyon belki soyutlamanın üst seviyelerinde (ülkü, şehir, veri merkezi) belirlenebilir. Bahsedilen kaynaklara örnek olarak: depolama, işleme, bellek ve aę bant genişlięi verilebilir.

Hızlı elastikiyet: Elastiki bir şekilde saęlanıp geri bırakılabilen özellikler bazen otomatik olarak ölçeklendirilebilir. Müşteriye, kaynak tedarikinde uygun özellikler sınırsız görünür ve herhangi bir zamanda herhangi bir miktarda tahsis edilebilir.

Ölçülen hizmet: Bulut sistemler, hizmet tipine göre (depolama, işleme, bant genişlięi, aktif kullanıcı hesapları) uygun soyutlama seviyesinde ölçme özelliğini yükselterek otomatik bir şekilde kaynakları kontrol ve optimize edebilir.

Bulut bilişimin üç servis modeli şöyledir:

Hizmet olarak yazılım (SaaS): Tüketicie saęlanan, saęlayıcının bulut altyapısı üzerinde çalışan uygulamalarını kullanma özelliğidir. Bu uygulamalara çeşitli müşteri cihazlarından, web tarayıcısı (ör: web tabanlı e posta) gibi bir istemci arayüzü veya bir program arayüzü aracılığıyla erişilebilir. Müşteri; aę, sunucular, işletim sistemi, depolama dahil temel bulut altyapısını yönetmez veya kontrol etmez.

Yaygın SaaS Senaryoları: Outlook, Hotmail veya Yahoo! Mail gibi web tabanlı e-posta hizmetleri SaaS türüne örnektir. Bu gibi hizmetlere, genellikle bir web tarayıcısı kullanarak, İnternet üzerinden kişisel hesaba giriş yapılarak erişilir. E-posta yazılımı, hizmet saęlayıcısının ağında bulunur ve iletiler de burada depolanır. E-postalara ve depolanan iletilere bir bilgisayardaki veya İnternet'e baęlı cihazdaki web tarayıcısından erişilir. Bu örnekler kişisel kullanıma yönelik ücretsiz hizmetlerdir. Kurumsal kullanım

söz konusu olduğunda, e-posta, işbirliği ve takvim gibi üretkenlik uygulamalarını ve müşteri ilişkileri yönetimi (CRM), kurumsal kaynak planlama (ERP) ve belge yönetimi gibi gelişmiş kurumsal uygulamalar kiralanabilir. Bu uygulamalar için abone olarak veya kullanım düzeyinize bağlı olarak ödeme yapılır.

Hizmet olarak platform (PaaS): Müşteriye sağlanan; programlama dilleri, kütüphaneler, servisler, kullanılarak oluşturulmuş, müşteri tarafından oluşturulan veya edinilen uygulamaları bulut altyapısına yerleştirmektir. Müşteri; ağ, sunucu, işletim sistemi veya depolama gibi temel bulut altyapılarını yönetmez veya kontrol etmez. Ama yerleştirilen uygulamalar üzerinde ve bu uygulamaların barınma ortamıyla ilgili yapılandırma ayarları üzerinde kontrolleri vardır.

Yaygın PaaS senaryoları: Kuruluşlar PaaS'yi genellikle şu senaryolar için kullanır:

Geliştirme çerçevesi: PaaS, geliştiricilerin bulut tabanlı uygulamaları geliştirebileceği veya özelleştirebileceği bir çerçeve sunar. PaaS, geliştiricilerin tıpkı Excel makrosu oluştururken olduğu gibi, yerleşik yazılım bileşenlerini kullanarak uygulama oluşturmalarına olanak tanır. Ölçeklenebilirlik, yüksek düzeyde kullanılabilirlik ve birden çok kiracı desteği gibi bulut özellikleri, geliştiricilerin yazması gereken kod miktarını azaltır.

Analiz veya iş zekası: PaaS ile birlikte hizmet olarak sunulan araçlar sayesinde kuruluşlar, verilerini analiz edip araştırarak öngörüler ve modeller elde edebilir, bu sayede sonuçlarını tahmin ederek talep tahminini, ürün tasarım kararlarını, yatırım getirilerini ve diğer iş kararlarını iyileştirebilir.

Ek hizmetler: PaaS sağlayıcıları iş akışı, izin, güvenlik ve zamanlama gibi uygulamaları geliştirmeye yönelik başka hizmetler de sunabilir.

Hizmet olarak altyapı (IaaS): Müşteriye sağlanan ve müşterinin, işletim sistemi ve uygulamaları da içeren isteğe bağlı bir yazılımı dağıtabilip çalıştırabildiği temel bilişim kaynakları ve işleme, depolama, ağ tedarikidir. Müşteri temel bulut altyapısını yönetmez ve kontrol etmez ancak işletim

sistemleri, depolama ve dağıtılmış uygulamalar üzerinde kontrolü vardır. Ağ bileşenleri (ör: güvenlik duvarı) gibi, seçme konusunda kısıtlı kontrolü vardır.

Yaygın IaaS senaryoları: Kuruluşların IaaS ile yapabileceklerinden bazıları şunlardır:

Test ve geliştirme: Ekipler test ve geliştirme ortamlarını hızlıca oluşturup kaldırabilir. Bu sayede yeni uygulamalar pazara çok daha hızlı şekilde sürülebilir. IaaS, geliştirme ve test ortamlarının ölçeğini artırma ve azaltma işlemini hem hızlandırır hem de bu işlemin maliyetini düşürür.

Web sitesi barındırma: IaaS kullanılarak çalıştırılan web siteleri, geleneksel web barındırmaya göre daha hesaplıdır.

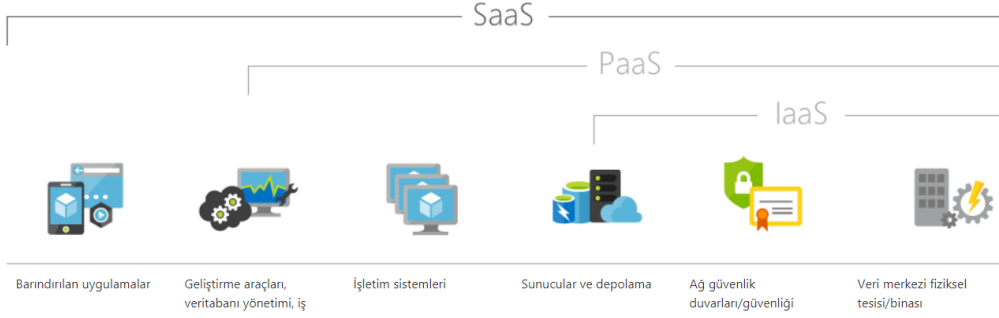
Depolama, yedekleme ve kurtarma: Kuruluşlar, veri yönetimi ve yasal gereksinimler ile uyumluluk gereksinimlerinin karşılanması için genellikle nitelikli personel gerektiren depolama yönetiminin karmaşıklığından ve depolama için yatırım masrafı yapmaktan kurtulur. IaaS, tahmin edilemeyen taleplerle ve giderek artan depolama ihtiyaçlarıyla başa çıkmaya yardımcı olur. Ayrıca, yedekleme ve kurtarma sistemlerinin planlama ve yönetim sürecini de basitleştirir.

Web uygulamaları: IaaS, web uygulamalarını ve ağ kaynaklarını desteklemek için gereken altyapıyı (depolama, web ve uygulama sunucuları gibi) sağlar. Kuruluşlar, web uygulamalarını IaaS’de hızlıca dağıtabilir ve uygulama talebinin tahmin edilemediği durumlarda altyapı ölçeğini kolayca artırıp azaltabilir.

Yüksek performanslı bilgi işlem: Süper bilgisayarlarda, bilgisayar şebekelerinde ve bilgisayar kümelerinde kullanılabilen yüksek performanslı bilgi işlem (HPC), milyonlarca değişken veya hesaplama içeren karmaşık problemlerin çözülmesine yardımcı olur. Deprem ve protein katlanması simülasyonu, iklim ve hava durumu tahminleri, mali modelleme ve ürün tasarımlarının değerlendirilmesi bu problemlere örnek olarak gösterilebilir.

Büyük veri analizi: Büyük veri; değerli olabilecek desenler, eğilimler ve ilişkiler içeren büyük ölçekli veri kümeleri için yaygın olarak kullanılan bir terimdir. Bu saklı desenleri bulabilmek veya ortaya çıkarabilmek için uygulanması gereken veri kümesi madenciliği, yüksek miktarda işlemci gücü gerektirir ve IaaS bu gücü uygun maliyetle sağlayabilir.

Şekil 2.1’ de bulut bilişimdeki servis modelleri gösterilmiştir. IaaS, PaaS ve SaaS modelleri katman şeklinde gösterilmiştir (Microsoft, 2019).



Şekil 2.1: Bulut bilişimde servis modelleri (Microsoft, 2019).

Bulut bilişimin dört dağıtım modeli şöyledir:

Özel Bulut: Özel bulut, özel olarak tek bir işletme veya kuruluş tarafından kullanılan bulut bilgi işlem kaynaklarından oluşur. Özel bulut, kuruluşun tesis içi veri merkezinde fiziksel olarak bulunabileceği gibi üçüncü taraf bir hizmet sağlayıcı tarafından da barındırılabilir. Ancak özel bulutta, hizmetler ve altyapı her zaman özel bir ağda bulunur; donanımlar ve yazılımlar yalnızca kuruluşa ayrılmıştır. Böylece özel bulut, kuruluşların belirli Bilgi Teknolojileri gereksinimlerini karşılamak için kaynaklarını özelleştirmesini kolaylaştırır. Özel bulutlar çoğunlukla devlet kurumları, finans kuruluşları ve ortamlarında yüksek denetim olanağı bulunmasını gerektiren, iş açısından kritik operasyonlara sahip diğer orta-büyük ölçekli kuruluşlar tarafından kullanılır.

Özel bulutun avantajları:

Daha fazla esneklik: Kuruluşun belirli iş ihtiyaçlarını karşılamak için bulut ortamını özelleştirebilir.

Yüksek güvenlik: Kaynaklar başkalarıyla paylaşılmadığından daha yüksek denetim ve güvenlik düzeyleri sağlanabilir.

Yüksek ölçeklenebilirlik: Özel bulutlar, genel bulutların ölçeklenebilirliğine ve verimliliğine sahiptir.

Topluluk Bulutu: Bulut altyapısı, ortak amaçları olan birkaç oluşum tarafından paylaşılır. Bir ya da daha fazla organizasyona, üçüncü partiye veya bunların kombinasyonuna ait olabilir, yönetilebilir, işletilebilir, tesis içinde veya dışında bulunabilir.

Genel Bulut: Genel bulut, bulut bilgi işleminden yararlanmaya başlamanın en yaygın yoludur. Sunucular ve depolama gibi bulut kaynakları, üçüncü taraf bulut hizmeti sağlayıcılarına aittir, bunlar tarafından işletilir ve internet üzerinden sunulur. Microsoft Azure, genel buluta bir örnektir. Genel bulutta tüm donanım, yazılım ve diğer destekleyici altyapı bulut sağlayıcısına aittir ve bu sağlayıcı tarafından yönetilir. Genel bulutta, bulut “kiracıları” olarak bilinen diğer kuruluşlarla aynı donanım, depolama ve ağ cihazları paylaşılır. Hizmetlere erişmek ve hesabı yönetmek için bir web tarayıcısı kullanılması gerekir. Web tabanlı e-posta, çevrimiçi ofis uygulamaları, depolama, test ve geliştirme ortamları sağlamak için sıklıkla genel bulut dağıtımları kullanılır.

Genel bulutun avantajları:

Düşük maliyet: Yazılım veya donanım satın almaya gerek yoktur ve yalnızca kullanılan hizmet için ödeme yapılır.

Bakım gerekmez: Bakım işlemlerini hizmet sağlayıcısı yapar.

Sınırsız yakın ölçeklenebilirlik: İşletme ihtiyaçlarını karşılamak için isteğe bağlı olarak kaynak tahsis edilir.

Yüksek güvenilirlik: Büyük ölçekli sunucu ağı sayesinde arızalardan korunmayı sağlar.

Karma Bulut: Şirket içi altyapıyı, diğer bir deyişle özel bulutları genel bulutlarla bir araya getiren karma bulutlar, kuruluşların bu iki bulut türünün de sağladığı avantajları bir arada elde etmesini sağlar. Karma bulutta, verilerle uygulamalar daha yüksek esneklik ve daha fazla dağıtım seçeneği sağlayacak şekilde özel ve genel bulutlar arasında taşınabilir. Örneğin web tabanlı e-posta gibi düşük güvenlik gereksinimine sahip ve yüksek hacimli ihtiyaçlar için genel bulut, finansal raporlama gibi hassas, iş açısından kritik ihtiyaçlar için de özel bulut (veya şirket içi altyapı) kullanılabilir.

Karma bulutun avantajları:

Denetim: Kuruluşun hassas varlıkları için özel bir altyapı kullanabilir.

Esneklik: İhtiyaç olduğunda genel buluttaki ek kaynaklardan yararlanabilir.

Maliyet etkinliği: Genel buluta ölçeklendirme özelliği sayesinde yalnızca ihtiyaç olduğunda ek bilgi işlem gücü için ödeme yapılır.

Kolaylık: İş yüklerini zaman içinde ve aşamalar halinde geçirebileceğinden buluta geçmek zor değildir (Mell and Grance, 2011; Microsoft,2019).

2.2. Sanallaştırma

Sanallaştırma kırk yıllık bir tekniktir ancak her zaman teknolojilerle sınırlıdır. Günümüzde sanallaştırma bulut bilişimin temel unsuru haline gelmiştir. Sanallaştırma donanım, çalışma zamanı (yürütme) ortamlarına, depolama ve ağlara uygulanabilir. Sanallaştırma talep üzerine sunulan bilgi işlem hizmetlerinin temel ihtiyaçları olan uygun derecede kişiselleştirme, güvenlik, izolasyon ve yönetilebilirliğe izin verir. Bulut sistemleri donanım ve programlama dili sanallaştırması, sunucu konsolidasyonu ve sanal makine yer değişimi tekniklerini kullanır. Donanım sanallaştırması IaaS teki çözümlere uygulanır. Programlama dili sanallaştırması PaaS te kullanılan bir teknolojidir. Sunucu konsolidasyonu ve sanal makine yer değişimi, bilişim hizmeti verirken izolasyon ve yönetilebilirliği sağlamak için uygulanır (Amalarethinam and Beena, 2014).

Sanallaştırma, aynı donanım makinesi içerisinde birden fazla sanal makinenin (VM) çoğaltıldığı bir bilgisayar mimarisi teknolojisidir. Sanal makine fikri, 1960'lı yıllara dayanır. Bir VM' nin amacı, birçok kullanıcı tarafından kaynak paylaşımını artırmak, kaynak kullanımı ve uygulama esnekliği açısından bilgisayar performansını iyileştirmektir. Donanım kaynakları (işlemci, bellek, Girdi / Çıktı cihazları vb.) veya yazılım kaynakları (işletim sistemi ve yazılım kütüphaneleri) çeşitli fonksiyonel katmanlarda sanallaştırılabilir. Bu sanallaştırma teknolojisi, dağıtık ve bulut bilişime olan talebin son yıllarda keskin bir şekilde artmasıyla canlandı.

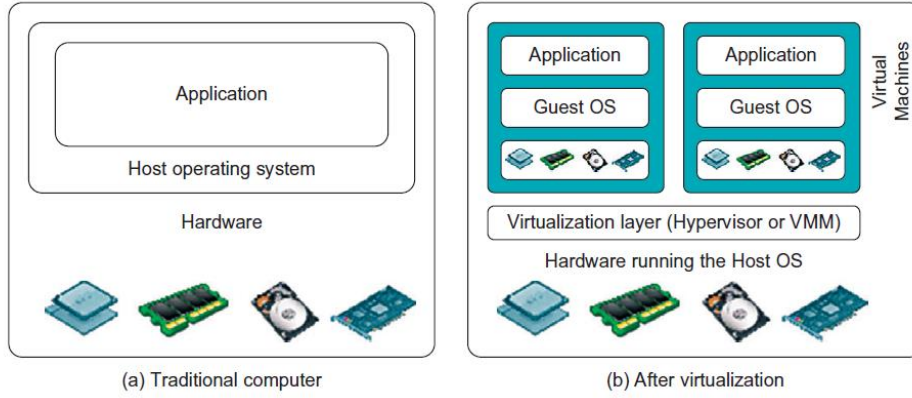
Buradaki fikir, daha iyi sistem verimliliği sağlamak için donanımı yazılımdan ayırmaktır. Örneğin, bilgisayar kullanıcıları sanal bellek kavramı ortaya çıktığında

genişletilmiş bellek alanlarına erişim kazandılar. Benzer şekilde, hesaplama motorlarının, ağların ve depolamanın kullanımını geliştirmek için sanallaştırma teknikleri uygulanabilir. Bu başlık altında, VM' ler ve bunların dağıtık sistemler kurmak için uygulamaları ele alınacaktır. 2009 Gartner Raporuna göre sanallaştırma, bilgisayar endüstrisini değiştirmeye hazır olan en büyük stratejik teknolojiydi. Yeterli depolama alanıyla, farklı komut setleri olan işlemciler kullanıyor olsalar ve aynı donanım üzerinde farklı işletim sistemleriyle çalışıyor olsalar bile, herhangi bir bilgisayar platformu başka bir ana bilgisayara kurulabilir. (Hwang et al., 2013).

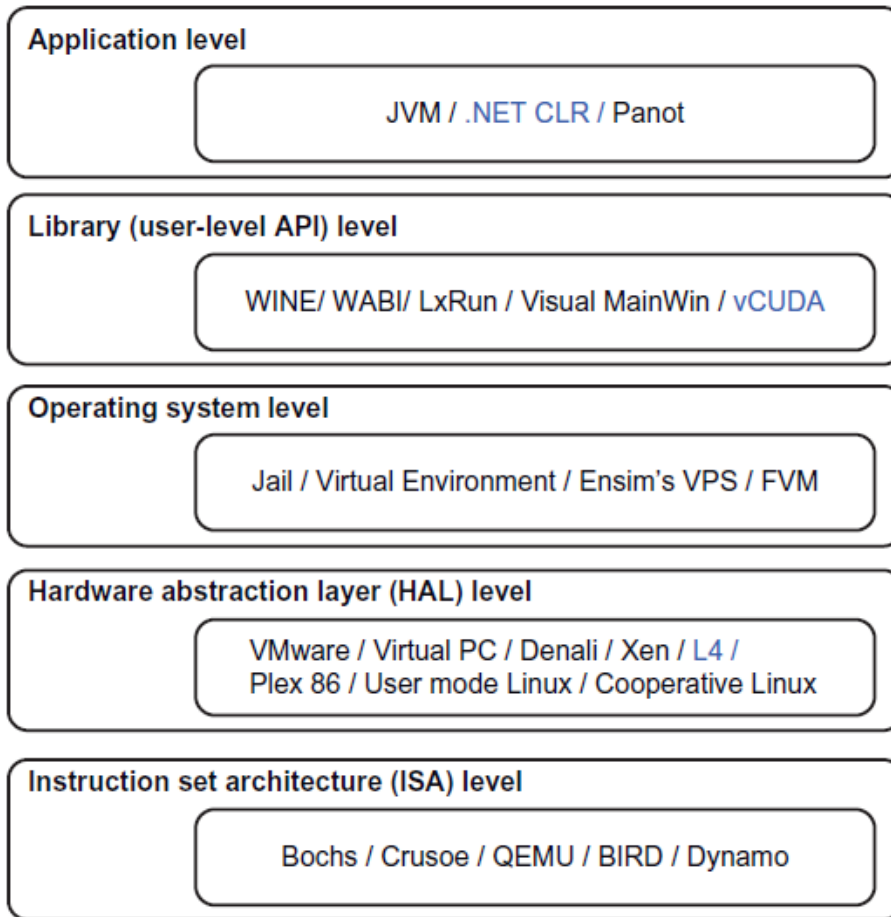
2.2.1. Sanallaştırma uygulama seviyeleri

Geleneksel bir bilgisayar, Şekil 2.2 (a) 'da gösterildiği gibi, donanım mimarisi için özel olarak tasarlanmış bir ana bilgisayar işletim sistemiyle çalışır. Sanallaştırmadan sonra, kendi işletim sistemleri (misafir işletim sistemi) tarafından yönetilen farklı kullanıcı uygulamaları, ana işletim sisteminden bağımsız olarak aynı donanımda çalışabilir. Bu genellikle Şekil 2.2 (b) 'de gösterildiği gibi sanallaştırma katmanı olarak adlandırılan ek bir yazılım ekleyerek yapılır. Bu sanallaştırma katmanı hipervizör veya sanal makine monitörü (VMM) olarak bilinir. Sanal Makineler, sanallaştırılmış CPU, bellek ve Girdi / Çıktı kaynakları üzerinden uygulamaların kendi konuk işletim sistemiyle çalıştığı üst kutularda gösterilir.

Sanallaştırma için yazılım katmanının ana işlevi, bir ana makinenin fiziksel donanımını, yalnızca VM' ler tarafından kullanılacak sanal kaynaklara sanallaştırmaktır. Bu, kısaca tartışacağımız gibi, çeşitli operasyonel seviyelerde uygulanabilir. Sanallaştırma yazılımı, sanallaştırma katmanını bir bilgisayar sisteminin çeşitli düzeylerinde araya sokarak VM' lerin soyutlamasını oluşturur. Yaygın sanallaştırma katmanları, şekil 2.3 teki gibi, komut seti mimarisi (ISA) seviyesi, donanım seviyesi, işletim sistemi seviyesi, kütüphane destek seviyesi ve uygulama seviyesini içerir.



Şekil 2.2: Sanallaştırmadan önce ve sonraki bilgisayar sistem mimarisi (Hwang et al., 2013).



Şekil 2.3: Beş soyutlama düzeyinde donanımdan uygulamalara değişen sanallaştırma (Hwang et al., 2013).

2.2.1.1. Komut seti mimari (ISA) seviyesi

ISA seviyesinde sanallaştırma, belirli bir ISA'yı ana makinenin ISA'sı tarafından taklit ederek gerçekleştirilir. Örneğin, MIPS ikili kodu, ISA emülasyonu yardımıyla x86 tabanlı bir ana bilgisayarda çalışabilir. Bu yaklaşımla, herhangi bir yeni donanım ana bilgisayarında çeşitli işlemciler için yazılmış büyük miktarda eski ikili kod çalıştırmak mümkündür. Komut seti emülasyonu, herhangi bir donanım makinesinde oluşturulan sanal ISA'lara imkan sağlar.

2.2.1.2 Donanım soyutlama seviyesi

Donanım düzeyinde sanallaştırma, doğrudan donanımın üstünde gerçekleştirilir. Bir yandan, bu yaklaşım bir VM için sanal bir donanım ortamı oluşturur. Öte yandan süreç, sanallaştırma yoluyla alttaki donanımı yönetir. Buradaki fikir, bilgisayarın işlemci, bellek ve Girdi / Çıktı cihazları gibi kaynaklarını sanallaştırmaktır. Amaç, donanım kullanım oranını aynı anda birden fazla kullanıcı tarafından yükseltmektir. Bu fikir, 1960'larda IBM VM / 370'da uygulandı. Daha yakın zamanlarda Xen hipervizörü, Linux veya diğer konuk işletim sistemi uygulamalarını çalıştırmak için x86 tabanlı makinelerin sanallaştırılmasında kullanıldı.

2.2.1.3 İşletim sistemi seviyesi

Bu, geleneksel işletim sistemi ve kullanıcı uygulamaları arasındaki bir soyutlama katmanını ifade eder. İşletim sistemi düzeyinde sanallaştırma, veri merkezlerindeki yazılım ve donanımı kullanmak için fiziksel sunucuda izole edilmiş bir konteynır veya işletim sistemi nesnesi oluşturur. Bu konteynırlar gerçek sunucular gibi davranır. İşletim sistemi düzeyinde sanallaştırma, donanım kaynaklarını çok sayıda kullanıcı arasında paylaşmak için sanal barındırma ortamları oluşturmada yaygın olarak kullanılır. Ayrıca, daha az bir ölçüde, ayrı ana bilgisayarlardaki hizmetleri bir sunucudaki konteynırlara veya VM'lere taşıyarak sunucu donanımını pekiştirmede (sağlamlaştırma) kullanılır.

2.2.1.4 Kütüphane destek seviyesi

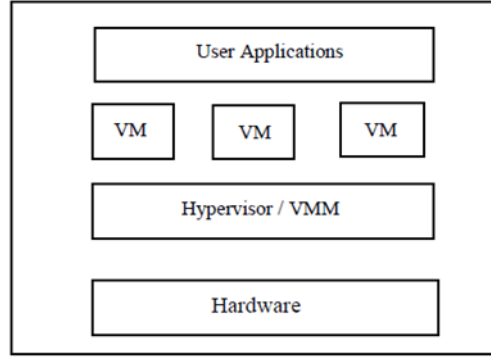
Çoğu uygulama, OS tarafından yapılan uzun sistem çağrılarını kullanmak yerine, kullanıcı düzeyinde kütüphaneler tarafından verilen API'leri kullanır. Çoğu sistem iyi dökümanlı edilmiş API'ler sağladığından böyle bir arayüz, sanallaştırma için bir aday haline gelir. Uygulamalar ve sistemin geri kalanı arasındaki iletişim bağlantısını API aracılığıyla kontrol ederek kütüphane arayüzleriyle sanallaştırma mümkündür.

2.2.1.5 Kullanıcı-uygulama seviyesi

Uygulama düzeyinde sanallaştırma, bir uygulamayı VM olarak sanallaştırır. Geleneksel bir işletim sisteminde, bir uygulama genellikle bir işlem olarak çalışır. Bu nedenle, uygulama düzeyinde sanallaştırma, işlem düzeyinde sanallaştırma olarak da bilinir. En popüler yaklaşım, yüksek seviyeli dil (HLL) sanal makinelerini dağıtmaktır. Bu senaryoda sanallaştırma katmanı, işletim sisteminin üstüne bir uygulama programı olarak oturur ve katman, belirli bir soyut makine tanımına göre yazılmış ve derlenmiş programları çalıştırabilen bir VM'nin soyutlanmasını dışarı aktarır. HLL'de yazılmış ve bu VM için derlenmiş herhangi bir program bu makine üzerinde çalışabilecektir. Microsoft .NET CLR ve Java Sanal Makinesi (JVM) bu VM sınıfına iki iyi örnektir (Hwang et al., 2013).

2.3. Bulut Mimarisi

Şekil 2.4, yüksek seviye mimarisinin bileşenlerinin genel bir görünümünü sunar. Üst katmanda kullanıcı arayüzü, hipervizör veya sanal makine yöneticisi tarafından sunulan hizmetlere erişim sağlar. Hipervizör genellikle temel donanımın soyutlanmasını (ayrılmasını) sağlayan bir program veya yazılım ve donanım birleşimidir. Hipervizörde görev dağıtıcı, çizelgeleyici ve yorumlayıcı olmak üzere üç temel modül bulunmaktadır.



Şekil 2.4: Yüksek seviye bulut mimarisi (Amalarethinam and Beena, 2014).

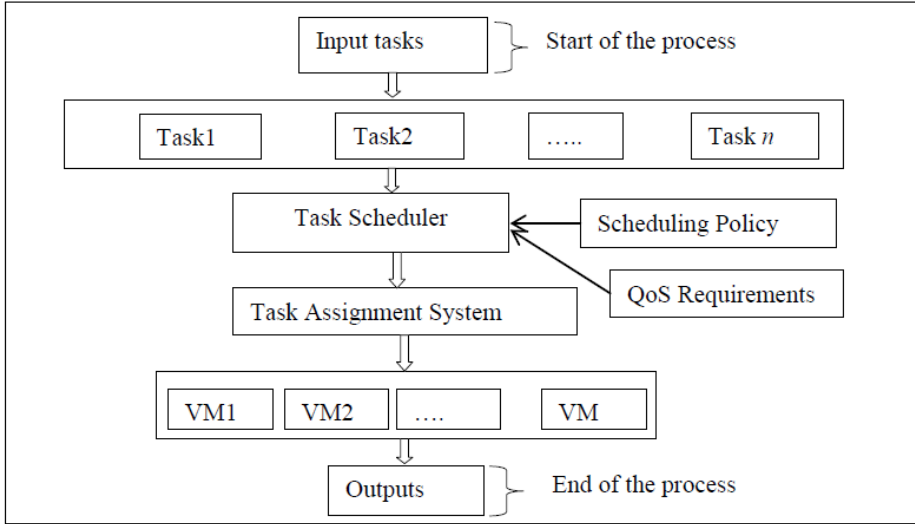
Görev dağıtıcı, denetimin giriş noktasını oluşturur ve sanal makine örnekleri tarafından verilen talimatları diğer iki modülden birine yeniden yönlendirir. Çizelgeleyici, sanal makineye sağlanacak sistem kaynaklarının kararlaştırılmasından sorumludur. Sanal bir makine, söz konusu VM ile ilişkili makine kaynaklarını değiştirmeye neden olan bir komutu uygulamaya çalıştığında, çizelgeleyici, görev dağıtıcı tarafından çağrılır. Yorumlayıcı modülü, yorumlayıcı rutinlerinden (program parçası) oluşur. Bunlar, bir sanal makine ayrıcalıklı bir komutu yürüttüğünde, karşılık gelen rutin çalıştırıldığında yürütülür. Ana rol, çizelgeleyici tarafından oynanır. Çizelgeleyici, çeşitli görevleri gerçekleştirmek için fiyatlandırma ve faturalandırma, izleme, rezervasyon, QoS / SLA yönetimi, VM deposu ve VM havuzu yöneticisi gibi diğer bileşenlerle etkileşime girer. Alt katman ise, üstünde sanal makine yöneticisinin çalıştığı fiziksel altyapıdan (işlemci, bellek) oluşur (Amalarethinam and Beena, 2014).

2.4. Görev Çizelgeleme

Görev çizelgeleme, sanallaştırılmış kaynakların belirli bir işe belirli bir süre için atanması olarak tanımlanabilir. Çizelgelemenin amacı, bulut kaynaklarını etkin bir şekilde kullanmaktır. Geleneksel çizelgeleme algoritmaları, bulut bilişimde çizelgeleme için uygun değildir, çünkü bulut için sağlanan ve kaldırılan kaynaklar dinamiklidir, bu yüzden daha uygun bir çizelgeleme algoritması gereklidir (Sudheer et al., 2017).

2.4.1. Problem tanımı

Çizelgeleme stratejilerine göre, zayıf görev planlama algoritmaları, tüm bulut bilişim sisteminin verimliliğini önemli ölçüde etkileyebilir. Bir bulut servis sağlayıcısının m adet makineye n adet görev atadığını varsayalım ($m < n$). $(t_1, t_2, t_3, \dots, t_n)$ çizelgelenmeyi bekleyen görevleri temsil etsin. Ardından, çizelgeleme politikasına ve QoS gerekliliklerine dayanan görev çizelgeleyici, Şekil 2.5' te gösterildiği gibi, hangi görevin sanal makineler koleksiyonundaki hangi makineye atanacağı konusunda karar verir (Nzanywayingoma and Yang, 2017). Bu tez çalışmasındaki problem tanımı, statik ve dinamik çizelgeleme ile görevlerin tamamlanma zamanı ve yük dengelemesini iyileştirecek şekilde sanal makinelere atanmasıdır. Sanal makineler *vmid*, *mips*, *ram*, *bw*, *cpu* özellikleri ile temsil edilir. *vmid*, sanal makine numarasını belirtir. *mips*, sanal makinenin saniyedeki milyon komut sayısı üzerinden hızını belirtir. *ram*, sanal makinenin hafızasını belirtir. *bw*, bant genişliğini belirtir. *cpu* ise sanal makinenin işlemcisini belirtir.



Şekil 2.5: Bulut bilişim ortamındaki görev çizelgeleme mimarisi (Nzanywayingoma and Yang, 2017).

Yapılan çalışmaların çoğunda statik çizelgeleme üzerinde çalışılmıştır ancak görevlerin tamamlanma zamanını iyileştirmeyi düşündüğümüzde, farklı zaman aralıklarında sisteme gönderilen görevlere dinamik olarak yani tüm görevlerin sisteme gönderilmesini beklemeden ve görevler hakkında bilgi sahibi olmadan çizelgeleme işlemi uygulanması bize gerçek zamanlı bir sonuç verir. Bu sebeple bu

çalışmada hem statik hem dinamik çizelgeleme yapılarak sonuçlar belirlenen ölçütler üzerinden incelenmiştir. Çizelgeleme, kesintisizdir. Yani çizelgeleme işleminde bir görev bölünemez, yarıda bırakılamaz. Görev tamamen işlenir. Görevler bağımsızdır. Bir görev sadece bir makineye atanabilir.

Görevlerin mevcut kaynaklara atanması, görevlerin maksimum tamamlanma zamanı, minimum tamamlanma zamanı ve kaynakların (makinaların) yük dengelenmesinin hesaplanmasında kullanılır. Uygunluk fonksiyonları ve bu değerlerin hesaplanmasına sonraki bölümlerde değinilecektir.

2.4.2. Çizelgeleme algoritmaları

Görev çizelgelemesi, gelen görevleri mevcut kaynaklara tahsis etmek için kullanılan bir prosedürdür. Görev çizelgeleme algoritmalarının temel amacı, bulutun servis parametrelerini etkilemeden kaynak kullanımını en üst düzeye çıkarmaktır. Beklendiği gibi, çok sayıda görevin çizelgelenmesi bilgisayar verimliliğinin düşmesine neden olur. Bu, uzun tamamlanma süresi, uzun bekleme süreleri ve artan maliyetlere yol açar. Bu nedenle, mevcut sanal makinelere çok sayıda görevin optimum tahsis edilmesi sorunu büyük bir zorluktur.

Bulut bilişimde çizelgeleme, büyük bir çözüm alanına sahip bir problem olarak kabul edilir. Bu nedenle, en uygun çözümü bulmak uzun zaman alır. Deterministik çizelgeleme algoritmalarının uygulanması çok daha kolay ve daha hızlıdır çünkü hepsi görevleri yönetmek ve düzenlemek için bir veya birkaç kurala dayanarak tasarlanır. Ancak, bu algoritmalar, özellikle sorun karmaşıklaştığında veya görev sayısı çok büyük olduğunda, makul bir sürede en uygun çözümü bulamamaktadır. Buna karşılık, meta sezgisel temelli tekniklerin, bu tür problemler için makul bir süre içinde en uyguna yakın çözümlere ulaştığı kanıtlanmıştır.

Meta-sezgisel tabanlı algoritmalar genellikle kalite açısından deterministik algoritmalarından daha iyi sonuçlar verir. Bu nedenle, meta-sezgisel algoritmalar, verimliliği ve etkinliği nedeniyle büyük bir popüleriteye sahiptir. Meta-sezgisel çizelgeleme algoritmalarının iki türü vardır; tek çözüm tabanlı meta sezgiseller ve popülasyon tabanlı meta sezgiseller. Tek çözüm tabanlı algoritmalar; HC (tepe

tırmanışı), SA (tavlama benzetimi) ve TS (tabu araması) gibi tek bir aday çözümün değiştirilmesine ve geliştirilmesine odaklanır. Öte yandan, popülasyon tabanlı algoritmalar; GA (genetik algoritma), DE (diferansiyel gelişim), birden fazla aday çözümle ilgilenmektedir. Bu nedenle, bu algoritmalar aramayı yönlendirmek için popülasyon özelliklerini kullanır. Meta-sezgisel algoritmalar, genellikle deterministik algoritmalarından daha yavaştır. Bu nedenle, bazı araştırma alanlarındaki son eğilim meta sezgisel algoritmaların yürütme süresini hızlandırmak için girdi verilerini önceden işlemeye çalışmaktadır. Meta sezgisel algoritmalar üzerinde çalışan son araştırmalar genellikle üç ana hedefe odaklanır: uygunluk fonksiyonunu değiştirmek, operatörleri değiştirmek ve melez meta sezgisel algoritmalar kullanmak (Saleh et al., 2019). Bu çalışmada statik veya dinamik çizelgelemeye göre uygunluk fonksiyonu değiştirilmiştir. Farklı operatörler kullanılmıştır. SAGA, GASA ve DESA gibi melez algoritmalar uygulanmıştır.

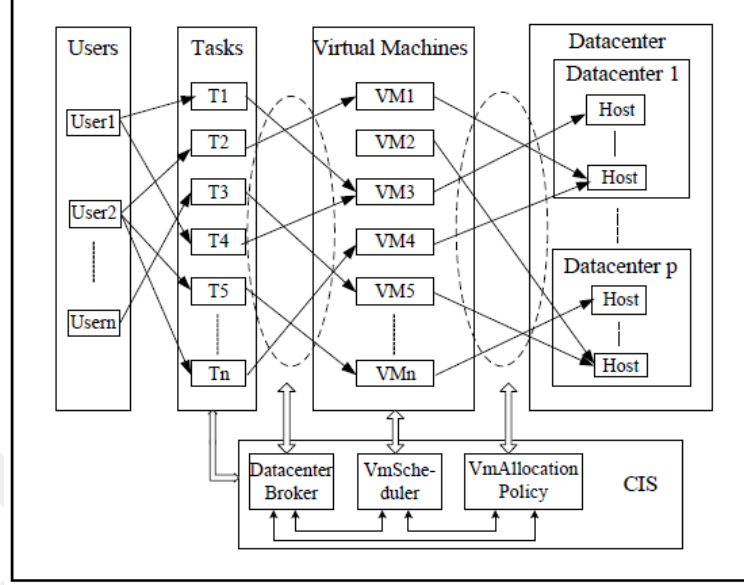
2.5. Simülasyon Ortamı: Cloudsim

CloudSim, IaaS (Hizmet Olarak Altyapı), PaaS (Hizmet Olarak Platform) ve SaaS (Hizmet Olarak Yazılım) modelleme senaryolarının simülasyonunu sağlar. Host, sanal makineler ve yukarıdaki üç tür hizmeti modelleyen uygulamalar gibi temel bileşenler sunar. Cloudsim aşağıdaki yeni özellikleri sunar: (i) Tek fiziksel hesaplama düğümü üzerindeki veri merkezi ve java sanal makinesi içeren büyük ölçekli bulut bilişim altyapısının destek modeli ve örneklenmesi; (ii) veri merkezlerini, servis brokerlerini, çizelgeleme ve tahsis politikalarını modellemek için müstakil bir platform; (iii) bir veri merkezi düğümünde birden fazla, bağımsız ve birlikte barındırılan sanallaştırılmış hizmetlerin oluşturulmasına ve yönetimine yardımcı olan sanallaştırma motorunun kullanılabilirliği; (iv) işlem çekirdeklerinin sanallaştırılmış hizmetlere alan paylaşımlı ve zaman paylaşımlı tahsisi arasında geçiş yapma esnekliği (Li et al., 2011; Calheiros et al., 2009).

2.5.1. Cloudsim çalışma mimarisi

Cloudsim çalışma biçimi Şekil 2.6'da gösterilmiştir. Genel olarak, farklı kullanıcılardan gelen görevler nispeten bağımsızdır; $User_1$, $User_2$, ..., $User_m$ olarak

m kullanıcı vardır. $T_1, T_2, \dots, T_n$ olarak n bağımsız görev vardır. $VM_1, VM_2, \dots, VM_n$ olarak n sanal makine vardır. $Datacenter_1, Datacenter_2, \dots, Datacenter_p$ olarak p veri merkezi vardır.



Şekil 2.6: Cloudsim çalışma şekli (Li et al., 2011).

CIS: CIS (Bulut Bilgi Servisi) veri tabanı düzeyinde eşleştirme hizmetleri sağlar; kullanıcı isteklerini uygun bulut sağlayıcılarına eşler. CloudSim'in CIS ve Veri merkezi Broker'ı, kaynak keşfi ve bilgi etkileşimini gerçekleştirir; bu, benzetimi yapılan çizelgelemenin temelini oluşturur.

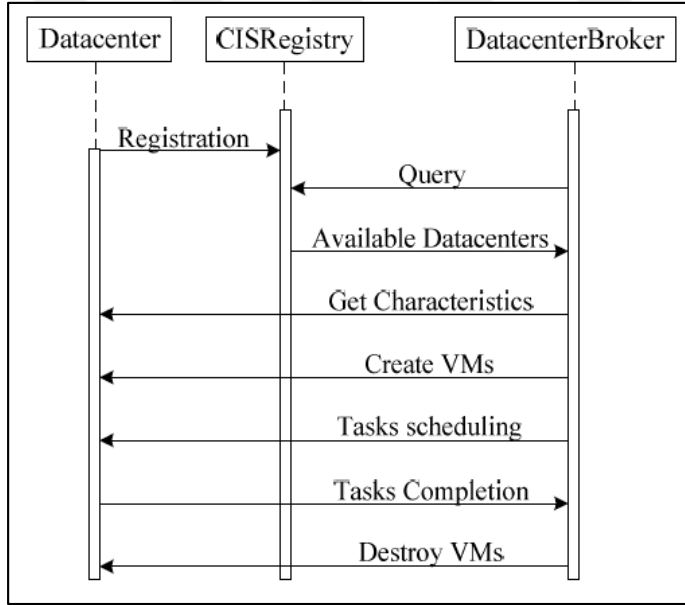
DatacenterBroker: Bu sınıf, kullanıcıların QoS gereksinimlerine bağlı olarak kullanıcılar ve servis sağlayıcılar arasındaki bağlantıyı sağlamaktan sorumlu olan broker'ı modellemektedir. Broker, görevleri bulut üzerinde dağıtır. Kullanıcı tarafından geliştirilen çizelgeleme algoritmaları bu sınıfta kodlanır. Bu sebeple, araştırmacılar ve sistem geliştiriciler bu sınıfı genişletmelidirler.

VmScheduler: Soyut bir sınıftır. Sanal makinelere işlem gücü tahsis etmek için gereken politikaları (alan paylaşımlı veya zaman paylaşımlı) temsil eder. Bu sınıfın fonksiyonelliği, belirli işlemci paylaşım politikalarına uyum sağlaması için kolaylıkla değiştirilebilir.

VmAllocationPolicy: Bu soyut sınıf, bir sanal makine gözlemcisinin sanal makineleri ana bilgisayarlara tahsis etmek için kullandığı tedarik politikasını temsil eder. Bu sınıfın temel işlevi, bir sanal makine yayılımı için bellek, depolama ve kullanılabilirlik gereksinimlerini karşılayan bir veri merkezindeki mevcut ve kullanılabilir ana bilgisayarları seçmektir (Li et al., 2011; Calheiros et al., 2009).

2.5.2. Birimler arası iletişim

Şekil 2.7, temel CloudSim (CloudSim, 2019) birimleri arasındaki iletişim akışını göstermektedir. Simülasyonun başlangıcında, her bir veri merkezi birimi, kendisini CIS (Bulut Bilgi Servisi) Kayıt Defterine kaydeder, ve DatacenterBroker birimler arasındaki bilgi etkileşimini yönetir (Li et al., 2011; Calheiros et al., 2009).



Şekil 2.7 Cloudsim veri akışı (Li et al., 2011).

2.6. İlgili Çalışmalar

Görev çizelgeleme, dağıtık sistemlerde oldukça popüler ve dikkat çeken bir çalışma alanıdır. Bittencourt et al.'a (2018) göre dağıtık sistemlerin ortaya çıkışı, kümeler, gridler ve daha yakın zamanda bulut olmak üzere bilgisayar sistemlerinde çizelgeleme konusunda yeni sorunlar ortaya çıkardı. Dağıtık sistemde, görev çizelgeleyici, her bir görevi (veya işi) hangi kaynağının işleyeceğini seçme, böylece

eşzamanlı olarak yürütülecek işleri dağıtma rolüne sahiptir. Her işin yürütme süresi ve sonuç olarak tüm işlerin bitirme süresi, oluşturulan çizelgelemeye göre belirlenir. Görev çizelgeleyici, kaynak sıralarındaki görevleri amaç fonksiyonuna bağlı olarak organize eder. Dağıtık sistemler literatüründe bulunan en yaygın amaç, mevcut kaynaklara görevlerin uygun şekilde tahsis edilmesiyle elde edilen tamamlanma zamanını en aza indirmektir.

Zomaya and Teh (2001), dağıtık sistemler üzerinde statik çizelgeleme kullanarak genetik algoritma ile dinamik yük dengeleme problemini çözmeye çalışmışlardır. Statik çizelgeleme olmasının sebebi, çalışmalarında tüm görevlerin önceden oluşturulup saklandığı “görev havuzu” nu kullanmalarındır. Görev havuzundaki görevleri, belli sayıda görev taşıma kapasitesi olan “kayan pencere” yöntemiyle seçerek çizelgeleme işlemine göndermişlerdir. Merkezi çizelgeleyici ile çalışılmıştır. Merkezi çizelgeleyici tarafından tespit edilen, sistemdeki işlemcilerden biri “boş” konumuna geldiği yani kendi kuyruğundaki tüm görevleri bitirdiği anda yük dengeleme mekanizması başlatılır böylece işlemcilerin “boş” konumda kalması önlenir ve yük dengelemesi sağlanır. Bu çalışmada önerilen genetik algoritma tabanlı dinamik yük dengeleme mekanizması, özellikle çok sayıda görevler üzerinde etkili sonuçlar vermiştir. Merkezi çizelgeleyici kullanılması, işlemciler arası iletişimi en aza indirdiği için verimliliği artırmıştır. “First fit” algoritmasıyla “tamamlanma zamanı” ve “işlemci kullanımı” ölçütleri üzerinden karşılaştırıldığında daha iyi sonuçlar verdiği görülmektedir. Dezavantajı ise dinamik çizelgeleme yöntemi kullanmamaları ve sabit taşıma kapasiteli “kayan pencere” kullanmalarındır. Dinamik çizelgeleme, statik çizelgelemeye göre gerçek zamana daha yakın sonuçlar verir.

Tsai et al. (2014), bulut istemleri için hiper sezgisel çizelgeleme algoritması geliştirmişlerdir. Önerilen algoritmanın temel fikri; tavlama benzetimi, genetik algoritma, parçacık sürü optimizasyonu ve karınca kolonisi optimizasyonu gibi yaklaşımları tek bir algoritmada entegre ederek bu yaklaşımların etkinliğini artırmaktır. Önerilen yöntem, günümüzde en gelişmiş birkaç çizelgeleme yöntemiyle kıyaslanmıştır. Sonuçlar, önerilen yöntemin fark edilir derecede “tamamlanma zamanını” azalttığını göstermektedir. Geliştirilen yaklaşımın avantajları: görevlerin tamamlanma zamanının azaltılması, hem sıralama bağımlı

hem sıralama bağımsız çizelgeleme problemlerine uygulanabilmesi, bu çalışmanın hem simülatör (CloudSim,2019) hem de gerçek sistem (Hadoop) üzerinde test edilmiş olmasıdır.

Alla et al. (2018), dinamik dağıtım kuyruklarına (TSDQ) ve melez meta-sezgisel algoritmalara dayalı yeni bir yaklaşım kullanarak görev çizelgelemesi optimizasyonuna odaklanmışlardır. Birincisi “parçacık sürü optimizasyonu ile bulanık mantık” (TSDQ-FLPSO), ikincisi “parçacık sürü optimizasyonu ile tavlama benzetimi” (TSDQ-SAPSO) olmak üzere iki melez algoritma önermişlerdir. Sentetik ve gerçek veri setleri kullanılarak açık kaynak simülatörü olan Cloudsim ortamında deneyler yapılmıştır. Deneysel sonuçlara göre, özellikle yüksek boyutlu problemlerde TSDQ-FLPSO’ nun etkinliği ve optimal sonuçlara ulaştığı TSDQ-SAPSO ve diğer çizelgeleme algoritmalarıyla kıyaslandığında ortaya çıkmıştır. TSDQ-FLPSO; bekleme zamanı, kuyruk uzunluğu, tamamlanma zamanı, maliyet, kaynak kullanımı, dengesizlik seviyesi ve yük dengeleme ölçütleri üzerinden değerlendirildiğinde büyük avantaj sağlamıştır.

İbrahim et al. (2017), bulut sistemlerde kaynakların uygunluk seviyesine göre yapılmış bir dinamik görev çizelgeleme yöntemi önermişlerdir. Bu yöntemde, görevlerin çalışma zamanı sırasında kaynakların o anki uygunluk seviyelerini ölçerek gelen görevleri en uygun kaynağa atamaktır. Kaynakların uygunluk seviyesi; işlem gücü, maliyet ve üzerinde çalışan görev sayısına bakarak belirlenir. Önerilen yöntemin performansı, Round Robin ve Minimum Tamamlanma Zamanı algoritmalarıyla tamamlanma zamanı ve çalışan görevlerin maliyeti ölçütleri üzerinden karşılaştırılmıştır. Önerilen yöntemin, diğer iki algoritmadan daha iyi sonuçlar verdiği ve tamamlanma zamanı ile görevlerin çalışma maliyetini azalttığı görülmüştür.

Ebadifard and Babamir (2018), yük dengeleme tekniği kullanarak parçacık sürü optimizasyonu ile çizelgeleme tekniğini geliştiren bir yaklaşım ortaya koymuşlardır. Görevlerin yarıda kesilemeyen ve bağımsız olduğu varsayılan statik bir çizelgeleme yöntemidir. Önerilen yöntem; round robin, gelişmiş PSO ve yük dengeleme tekniğiyle karşılaştırılmıştır. Deneyler için Cloudsim simülatörü kullanılmıştır. Simülasyon sonuçları, geliştirilen yöntemin temel PSO ya kıyasla

%22 lik kaynak kullanımı artışı ve %33 tamamlanma zamanının azalmasıyla daha iyi performans gösterdiğini ortaya çıkarmıştır. Ayrıca geliştirilen yöntem, yakın optimal sonuca temel PSO ya göre daha hızlı yakınsamıştır ve görev sayısı arttıkça yöntemin etkinliği de artmıştır.

Gabi et al. (2018), bulut ölçeklendirilebilir çok amaçlı bir kedi sürüsü optimizasyonu tabanlı tavlama benzetimi algoritması (CSM-CSOSA) önermişlerdir. Önerilen yöntemde, ortogonal Taguchi yaklaşımı, ölçeklenebilirlik performansı için önerilen CSM-CSOSA algoritmasının yerel aramasına dahil edilen SA'yı geliştirmek için uygulanır. Önerilen algoritmanın CloudSim aracı üzerindeki etkinliğini iki farklı veri setiyle değerlendirmek için yürütme süresi ve yürütme maliyeti ölçütlerine dayanan çok amaçlı bir QoS modeli sunulmuştur. Deney sonuçlarına göre önerilen CSM-CSOSA; çok amaçlı genetik algoritma, çok amaçlı karınca kolonisi ve çok amaçlı parçacık sürü optimizasyonundan daha iyi performans göstermektedir.

Gan et al. (2010), bulut bilişimde genetik algoritma ve tavlama benzetimini birlikte kullanarak melez bir algoritma önermişlerdir. Mevcut popülasyon üzerinde sırasıyla seçme, çaprazlama, mutasyon, tavlama benzetimi işlemleri yapılarak yeni popülasyon elde edilir. Deneysel sonuçlar, algoritmanın bulut bilişim ortamında kaynak arama ve tahsisi verimli bir şekilde tamamladığını göstermektedir.

Luo and Fu (2018), genetik tavlama benzetimi algoritmasına dayalı ve kaotik parçacık sürü optimizasyonu algoritmasına dayalı 2 farklı Hadoop sistem performansının yapılandırma optimizasyon yöntemi üzerinde çalışmışlardır. Hadoop' un açık kaynaklı bulut bilişim platformundaki performansı çerçevesinde, bir optimizasyon yöntemi önerilmiştir. Genetik tavlama benzetimi algoritmasına dayanarak her yapılandırma şeması; çoklu seçim, çaprazlama ve mutasyon için bir kromozom olarak kullanılmıştır. Tavlama benzetimi prensibiyle birlikte yeni kromozomun hayatta kalması (yeni nesle aktarılması) ve tüm algoritmanın toplam tekrar sayısı kontrol edilir ve sistemin optimal yapılandırma şeması bulunur. Deneysel sonuçlara göre bu yöntem, işlem verimliliğini büyük ölçüde artırmıştır.

Literatürdeki diğer hibrit yaklaşımlar da, Ge et al. (2018) ve Kamalinia and Ghaffari (2017) 'nin çalışmalarıdır. Ge et al. (2018), çalışmasında diferansiyel gelişim algoritması ve karınca kolonisi optimizasyonunun kombinasyonu ile elde edilen DEACO yaklaşımını ortaya koymuşlardır, değerlendirme ölçütlerinde en kısa görev tamamlanma zamanı, maliyet ve yük dengelemesini kullanmıştır. Çalışma Cloudsim ortamında yapılmıştır, sonuçları min-min ve ACO algoritmalarıyla karşılaştırılmıştır. DEACO' nun bu algoritmalarından daha üstün olduğu görülmüştür. Kamalinia and Ghaffari (2017) çalışmalarında HEFT, genetik algoritma ve diferansiyel gelişim algoritmasının birlikte kullanıldığı hibrit bir yöntem önermişlerdir. Bu yöntem, rastgele oluşturulmuş doğrudan döngüsüz graflarda tamamlanma zamanı ölçütüne göre bazı sezgiseller ve genetik algortimadan daha iyi sonuçlar vermiştir.

Ge et al. (2017), geliştirilmiş diferansiyel gelişim algoritması tabanlı bulut bilişimde görev çizelgeleme yöntemi önermişlerdir. Bu yöntemin, görev yürütme zamanını ve maliyeti azalttığını belirtmişlerdir.

Gawali and Shinde (2018) ve Fan et al. (2013), bulut bilişimin kaynak tahsisi problemine yönelmişlerdir. Bu problemde sanal makinelerin daha az sayıda ana bilgisayara atanması maliyetin azalması, veri merkezinin hızlı ve verimli çalışması açısından gereklidir. Fan et al. (2013), kaynak tahsisi ve çizelgelemesi problemini çözmek için tavlama benzetimi yük dengeleme algoritmasını önermiştir. Önerilen yöntem yük dengelemesinde başarılı olmuştur. Round robin ve temel tavlama benzetimi algoritmalarından daha iyi performans göstermiştir. Gawali and Shinde (2018), hem görev çizelgeleme hem de kaynak tahsisi problemleri için sezgisel bir yaklaşım önermiştir. Önerilen sezgisel yaklaşım geri dönüş süresi ve yanıt süresi açısından mevcut BATS ve IDEA çerçeveleriyle karşılaştırıldığında, daha gelişmiş sonuçlar verdiği görülmüştür.

3. GELİŞTİRİLEN YAKLAŞIMLAR

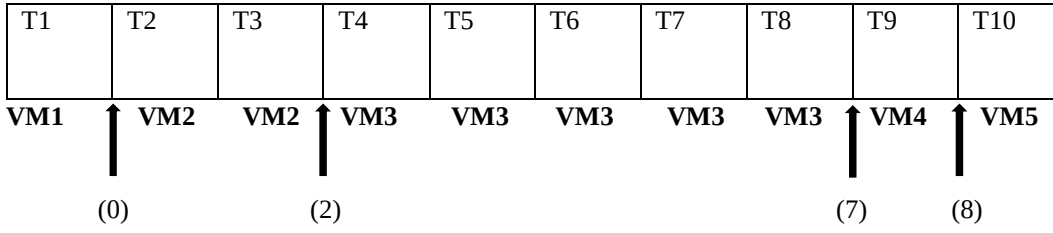
Bu bölümde, tezin geliştirilmesinde kullanılan temel teknikler, formüller, fonksiyonlar ve uygulanan algoritmalar anlatılmıştır. Ardından statik çizelgeleme ve dinamik çizelgeleme adımları anlatılmıştır.

Bu çalışmada; genetik algoritma (GA), tavlama benzetimi (SA), açgözlü yaklaşım (Greedy), diferansiyel gelişim algoritması (DE) ile üç adet melez algoritma; SAGA, GASA, DESA olmak üzere toplam 7 algoritma çeşitli sayıda görev içeren veri setleri üzerinde uygulanmıştır. Aynı veri setleri üzerinde hem statik hem dinamik çizelgeleme uygulanmış ve sonuçları belirtilmiştir. Her iki çizelgeleme türünde de “kayan pencere” tekniği uygulanmıştır. Statik çizelgelemede her kayan pencere için yukarıda belirtilen algoritmalar tek başına uygulanmıştır. Dinamik çizelgelemede ise “kayan pencere” deki görevlerin değişen sayısına göre ya yukarıdaki algoritmalarından biri ya da açgözlü yaklaşım uygulanmıştır. Bu tekniklerin detayları ileri bölümlerde anlatılacaktır.

3.1. Kromozom Yapısı

Meta sezgisel GA, SA, DE, SAGA, GASA VE DESA algoritmalarında kromozom yapısı gereklidir. Kromozom; *görev (cloudlet) listesi* ve bu görevlerin hangi makinaya atanacağını gösteren *kromozom için makine id listesi* ve kromozomun uygunluk değerini gösteren *fitness* değerinden oluşmaktadır. Kromozomun her bir geni bir görevi tutar. Popülasyon ilk kez oluşturulurken belirtilen sayı kadar boş kromozom oluşturulur ve bir sonraki bölümde anlatılacak olan “kayan pencere” içindeki görevler kromozomun genlerine aktarılır. Bu işlemden sonra kromozomdaki her bir görevin hangi makinaya atanacağı belirlenir. Bunun için 0’dan başlayarak kromozomdaki görev sayısının 2 eksiğine kadar olan tam sayılar arasından, makine sayısının 1 eksiği adedince sayı rastgele seçilir. Seçilen sayılar küçükten büyüğe sıralanır. Bu sayılar, kromozomdaki kesim noktalarını oluşturur. Kromozomun başlangıç, bitiş ve kesim noktaları dikkate alınarak, her makineye en az 1 görev atanmak üzere, görevlerin atanacağı makineler belirlenir. Örneğin 5 adet sanal makine (VM1, ..., VM5) ve 10 genden (T1, ..., T10)

oluşan bir kromozom olsun. Kesim noktaları 0, 2, 7, 8 olsun, buna göre oluşan yapı Şekil 3.1' deki gibidir.



Şekil 3.1: Geliştirilen kromozom yapısı.

3.2. Kayan Pencere Tekniği

Amaç, görevler geldikçe görev çizelgelerini "anında" belirlemek olduğundan, o sırada sistem tarafından bilinen görevlerin listesini göstermek için popülasyondaki kromozomlar kullanılır. Bir seferde atanmayı bekleyen çok fazla görev olabileceğinden, kayan pencere tekniği kullanılır, böylece yalnızca pencerede bulunan görevler her seferinde yürütme için kabul edilir (Zomaya and Teh, 2001; Pico and Wainwright, 1994). Kayan pencere, görevleri barındıran bir listedir.

Bu çalışmadaki statik çizelgelemede, kayan pencere sabit boyuttadır. Sistemde herhangi bir makine boş duruma düştüğünde, kayan pencere önceden oluşturulmuş görev havuzundan kendi boyutu kadar görevi alır çizelgelenmesi için gönderir. Makinelerin durumuna göre yapılan değerlendirmeler sonucunda kayan penceredeki belli görevler çizelgelemeye kabul edilir, kabul edilmeyenler kayan pencerede kalır. Tekrar herhangi bir makine boş duruma düştüğünde kayan pencere güncellenir. Bu işlem sırasında kayan pencerede kalmış görevler sabit kalır, boş yerlere görev havuzundan yeni görevler eklenerek tekrar sisteme gönderilir. Bu işlem görev havuzunda ve kayan penceredeki tüm görevler bitene kadar devam eder.

Bu çalışmadaki dinamik çizelgelemede, kayan pencere değişken boyuttadır. Bu çizelgelemede önceden oluşmuş görev havuzu yoktur. Görevler farklı zamanlarda oluşur. Görevler geldikçe kayan pencereye eklenir. Sistemde herhangi bir makine boş duruma düştüğünde, o anda kayan pencerede olan görevler sisteme

gönderilir. Penceredeki tüm görevler sisteme kabul edilir ve çizelgeleme işlemi yapılır. Tekrar herhangi bir makine boş duruma düştüğünde, o anda kayan pencerede olan görevler sisteme gönderilir. Bu işlem, tüm görevlerin çizelgelenmesi tamamlandığında sona erer.

Geliştirilen yöntemlerde kullanılan temel kavramlar sonraki bölümde açıklanacaktır.

3.3. Maxspan, Ortalama Kaynak Kullanımı, Kabul Edilebilir Kuyruk Sayısı ve Uygunluk Fonksiyonu

Zomaya and Teh (2001), *maxspan* tanımını şu şekilde yapmışlardır: Maxspan, temelde sistemdeki tüm işlemciler arasındaki en büyük görev tamamlama zamanıdır. Maxspan, tamamlanma zamanından (*makespan*) farklıdır. Bu tez çalışmasındaki tamamlanma zamanı, sistemde işlenen son görevin işlenmesinin bittiği andaki sistem zamanıdır. Maxspan ise belli bir zamanda sistemdeki kaynaklara bakıldığında kaynaklar üzerinde devam eden görevlerin, bekleme listesinde ataması yapılmış olan görevlerin ve elimizdeki çözüme (*kromozoma*) göre ataması yapılacak olan görevlerin her bir kaynak üzerinde ne kadar zamanda biteceğinin hesaplanarak en uzun zamanda bu görevleri tamamlayacak olan kaynağın bu görevleri kaç birim zamanda bitireceğidir. Bir örnekle açıklanırsa: elimizde beş sanal makine olsun ve bunların şu andaki yükü (şu andaki görevin yükü ve işlemcinin bekleme listesindeki görevlerin yükü) zaman cinsinden 3, 4, 17, 13 ve 8 olsun. Elimizdeki çözüme (kayan penceredeki görevlere meta- sezgisel algoritmaların uygulanması sonucu elde edilen çözüm *kromozomu*) göre sanal makinelerle atanacak görevlerin yükleri Şekil 3.2' de belirtilmiştir. Şekilde gösterilen görevlerde ilk sayı görevin numarasını belirtir, parantez içindeki değer ise görevin zaman cinsinden boyutunu belirtir. Bu değere, görevin boyutunu (*MI* cinsinden) atandığı makinenin saniyede işleyebileceği milyon komut sayısına (*MIPS*) bölerek ulaşılır. Bu işlem, Formül 3.1' de belirtilmiştir. Her bir makinenin yükünü bulduran formül, Formül 3.2'de belirtilmiştir. Buradaki en büyük sonuç *maxspan* değerini verir. Tüm bunlar dikkate alındığında makinelerin yükleri Şekil 3.3' te gösterilmiştir.

VM1	T1 (7)					
VM2	T2 (10)	T3 (10)				
VM3	T4 (2)	T5 (8)	T6 (3)	T7 (3)	T8 (2)	
VM4	T9 (14)					
VM5	T10 (28)					

Şekil 3.2: Kaynaklara atanacak görevler ve yükleri.

VM_j ye atanan t_i nin yükü :

$$load_{ij} = \frac{MI(t_i)}{MIPS(VM_j)} \quad (3.1)$$

VM_j (yük)= VM_j (anlık yük) + VM_j (bekleyen görevler listesindeki toplam yük) + VM_j (çözümde atanan görevlerin toplam yükü)

(3.2)

	Toplam Yük						
VM1	3	7					10
VM2	4	10	10				24
VM3	17	2	8	3	3	2	35
VM4	13	14					27
VM5	8	28					36

Şekil 3.3: Formül 3.2' ye göre hesaplanan anlık sistem yükleri ve maxspan.

Şekil 3.3' e göre maxspan 36 zaman birimidir. Maxspan in daha kısa olması, çizelgelemenin daha iyi yapıldığı ve görevlerin daha kısa sürede tamamlandığı anlamına gelir. Maxspan' i azaltmak amaç fonksiyonunu iyileştirir.

Zomaya and Teh (2001)' e göre *ortalama kaynak kullanımı* çözüm kromozomunun uygunluk değeri için önemli faktörlerden biridir. Kaynak kullanımının yüksek olması yükün tüm kaynaklar arasında dengeli dağıldığını

gösterir. Kaynak kullanımını artırarak, tamamlanma zamanı düşürülebilir. Her bir kaynağın, verilen görev atamasına göre beklenen kullanımı hesaplanmalıdır. Bu, her kaynağın yükünü maxspan değerine bölerek elde edilir Formül 3.3' te belirtilmiştir.

$$VM_j(utilization) = VM_j(yük) / maxspan \quad (3.3)$$

Şekil 3.3' te belirtilen makinelerin kaynak kullanımı ve bu makinelerin ortalama kaynak kullanımı, tüm makinelerin kaynak kullanımının toplanıp makine sayısına bölünmesiyle bulunur:

$$VM1 = 10/36 = 0.27$$

$$VM2 = 24/36 = 0.66$$

$$VM3 = 35/36 = 0.97$$

$$VM4 = 27/36 = 0.75$$

$$VM5 = 36/36 = 1$$

$$\begin{aligned} \text{Average Resource Utilization (ARU)} &= (0.27 + 0.66 + 0.97 + 0.75 + 1) / 5 \\ (\text{Ortalama Kaynak Kullanımı}) &= 0.73 \end{aligned}$$

Ortalama kaynak kullanımı optimize edilerek, kaynakların uzun süre boşa kalma olasılığı azalacaktır.

Zomaya and Teh (2001)' e göre, uygunluk fonksiyonu için optimize edilmesi gereken üç amaç; *maxspan*'in azalması, *kaynak kullanımının* artması ve *kabul edilebilir makine kuyruklarını* iyileştirmektir. Makine kuyrukları, yani çözüm kromozomunda oluşan hangi görevin hangi makineye atanacağı durumunun makinelerde fazla yüke veya az yüke sebep olup olmayacağı kontrol edilmelidir. Bu amaç sadece statik çizelgelemede kullanılır, dinamik çizelgelemede; kayan pencerede bulunan tüm görevlerin atanmasını yapmak için bu amaç kullanılmamıştır. Bu atanan görevlerin (Şekil 3.1) kabul edilip edilmeyeceği, makine için fazla veya az yüke sebep olup olmayacağı her makine için belirlenen *alt eşik değeri* ve *üst eşik değerine* göre belirlenir. Makinelerin toplam yükü bu eşik değerleri arasındaysa, bu makinelerin görev kuyruğu kabul edilir. Bu amacın optimize edilmesi için kabul edilebilen kuyruk sayısının tüm kaynak sayısına bölünmesinden elde edilen oran kullanılmıştır. Bu oranın yükselmesi yük dengelemesi bakımından daha iyi bir çizelgeleme sağlar.

Uygunluk fonksiyonunda yer alan üç hedefe değindikten sonra, görev çizelgeleme kalitesini artırmak için kullanılan uygunluk fonksiyonu statik ve dinamik çizelgelemede farklılık göstermektedir. Statik çizelgelemede kullanılan uygunluk fonksiyonu Formül 3.4' te belirtilmiştir.

$$fitness = \frac{1}{maxspan} \times average\ resource\ utilization \times \frac{\# acceptable\ queues}{\# resources} \quad (3.4)$$

Dinamik çizelgelemede kullanılan uygunluk fonksiyonu Formül 3.5' te belirtilmiştir.

$$fitness = \frac{1}{maxspan} \times average\ resource\ utilization \quad (3.5)$$

Kullanılan parametreler değerlendirildiğinde bu uygunluk fonksiyonları maksimize edilmelidir.

3.4. Geliştirilen Algoritmalar

Bu çalışma kapsamında GA, SA, Greedy, DE, SAGA, GASA, DESA algoritmaları kullanılmıştır. Çalışmada kullanılan bu algoritmalar sırasıyla anlatılacaktır.

3.4.1. Genetik algoritma

Genetik algoritmalar, doğal seçim ve doğal genetik tekniğine dayanan arama algoritmalarıdır. İnsan arayışının yenilikçi özelliklerinden bazılarını kullanarak bir arama algoritması oluşturmak için, kromozomlar arasında en uygun olanın hayatta kalması, yapılandırılmış ancak içinde rastgelelik barındıran bir bilgi alışverişiyle birleştirilmektedir. Her nesilde, genlerin ve eski neslin en iyi kromozomlarının parçaları kullanılarak yeni yapay yaratıklar (kromozom) seti yaratılır; zaman zaman yeni bir parça iyi ölçüm almak için denenir. Genetik algoritmalar, geçmişteki bilgileri etkili bir şekilde kullanarak, beklenen iyileştirilmiş performansla yeni arama noktaları üzerinde tahminde bulunurlar.

Basit genetik algoritmanın mekaniği şaşırtıcı bir şekilde basittir; dizileri kopyalamaktan ve dizilerin bazı kısımlarını değiştirmekten daha karmaşık bir şey içermez. Bu basit sürecin neden işe yaradığının açıklaması çok daha ince ve güçlüdür. İşlemin basitliği ve etki gücü, genetik algoritma yaklaşımının ana çekim merkezlerinden ikisidir (Goldberg, 1989).

Birçok pratik problemde iyi sonuçlar veren basit bir genetik algoritma, üç operatörden oluşur. Bunlar; *seçilim*, *çaprazlama* ve *mutasyon*dur. GA 'nın etkinliği *keşfetme* ve *faydalanma* nın uygun şekilde birleşimine bağlıdır. Uygunluk değerine göre seçim yapma *keşfetme* nin kaynağıdır. *Faydalanma* nın kaynağı ise çaprazlama ve mutasyon dur.

Amaç fonksiyonu, çözümün kalitesini değerlendirmek için kullanıldığından optimizasyon metotlarının en önemli bileşenlerindendir. Minimize veya maksimize edilmesi gereken numerik değerdir. Buradaki asıl hedef, tüm işlemciler arasında minimum çalışma süresi, maksimum işlemci kullanımı ve dengeli bir yük dağılımı elde edecek görev çizelgelemesine ulaşmaktır. Ardından, amaç fonksiyonu GA' nın uygunluk fonksiyonuna dahil edilir. Bu uygunluk fonksiyonu kromozomların, algoritmanın hedefleriyle ilgili performansını ölçmek için kullanılır (Zomaya and Teh, 2001).

3.4.1.1. Seçilim, çaprazlama ve mutasyon

Geliştirilen yöntemlerde kullanılan temel kavramlar ve uygunluk fonksiyonu için Zomaya and Teh (2001) 'in çalışmasından yararlanılmıştır. Onlar çalışmalarında *roulette wheel selection*, *cycle crossover* ve *swap mutation* teknikleriyle GA'yı kullanmışlardır. Ancak bu tez çalışmasında *elitism* yöntemiyle, *tournament selection*, *order based crossover* ve *swap mutation* kullanıldığında diğer çalışmaya göre daha iyi uygunluk değerleri elde edilmiştir. Bu sebeple tez çalışmasında elitism yöntemi, turnuva seçilimi, order based çaprazlama ve swap mutasyon kullanılmıştır. Bunlar sırasıyla açıklanacaktır.

Bazen, çaprazlama veya mutasyon ebeveynlerden daha zayıf olan yavrular ile sonuçlandığında iyi bireyler kaybedilebilir. Bununla mücadele etmek için elitizm

olarak bilinen bir özellik kullanabiliriz (Cheng et al., 2011) . Elitizm yönteminde bir önceki neslin en iyi kromozomu, üzerinde hiçbir değişiklik yapılmadan olduğu gibi yeni nesile aktarılır. Böylece en iyi kromozom korunmuş olur. GA'nın operasyonel bir özelliği olan elitizm, en iyi kromozomun özelliklerinin gelecek nesillere geçmesine / kopyalanmasına izin vererek genetik sapmanın azalmasını sağlar (Ahn and Ramakrishna, 2003).

GA' lar, bir eşleşme havuzuna eklenecek bireyleri popülasyon içinden seçmek için bir seçim mekanizması kullanır. Eşleşme havuzundan bireyler, yeni yavrular üretmek için kullanılır; sonuçta ortaya çıkan yavrular, gelecek nesillerin temelini oluşturur. Eşleşme havuzundaki bireyler, genleri gelecek nesillere miras kalanlar olduğundan, eşleşme havuzunun "iyi" bireylerden oluşması arzu edilir. Seçim baskısı, daha iyi bireylerin tercih edilme derecesidir: seçim baskısı ne kadar yüksek olursa, bireyler o kadar çok tercih edilir. Bu seçim baskısı, GA' yı sonraki nesillere göre popülasyon uygunluğunu iyileştirmeye itmektedir. Turnuva seçimi, s turnuva boyutu olmak üzere, s rakipleri arasında bir turnuva tutarak seçim baskısı sağlar. s kromozom içinden en fazla uygunluk değerine sahip olan kromozom turnuvanın galibidir. Turnuvanın galibi eşleşme havuzuna eklenir. Eşleşme havuzunun ortalama uygunluk değeri popülasyondan daha yüksektir. Bu uygunluk değeri farkı, GA' ya sonraki her neslin uygunluğunu geliştirmeye iten seçim baskısını sağlar. Daha büyük bir turnuvanın kazananı, ortalama olarak, daha küçük bir turnuva kazananından daha yüksek bir uygunluk değerine sahip olacağından, turnuva boyutunu arttırmak suretiyle seçim baskısı artırılabilir (Miller and Goldberg, 1995). Tez çalışmasında, yeni bir birey oluşturmak için, popülasyondaki birey sayısının yarısı boyutunda iki adet turnuva popülasyonu oluşturulmuştur. Rastgele seçilen bireyler bu popülasyona eklenmiş ve her iki turnuva popülasyonunun en iyi bireyleri seçilmiştir. Bu iki en iyi birey, yeni oluşacak bireyin ebeveynlerini oluşturur.

Tez çalışmasında uygulanan çaprazlama yönteminde, Sastry et al. (2005)' in belirttiği *uniform order based crossover* ve *order based crossover* tekniklerinin birleşiminden oluşan bir teknik kullanılmıştır. Bu çaprazlama sonucunda iki ebeveynden tek çocuk oluşmaktadır. Ebeveynler P1 ve P2, çocuk C1 olsun. P1 için iki tane random kesim noktası belirlenir, bu iki kesim noktasın arasında kalan genler

konumları korunarak C1' e kopyalanır. C1' de boş kalan genler içinse sırayla P2' nin ilk geninden başlanarak C1 de bulunmayan görev numarasına sahip görevlerin bulunduğu genler C1'e aktarılır. Bu teknik, kesim noktası oluşturulması özelliğini *order based crossover*' dan, P2'den genlerin C1'e sırayla aktarılması özelliğiyle de *uniform order based crossover*' dan almıştır. Şekil 3.4 *uniform order based crossover*' ı, Şekil 3.5 *order based crossover*' ı, Şekil 3.6 ise tez çalışmasında kullanılan crossover' ı göstermektedir.

Parent P ₁	A	B	C	D	E	F	G
Parent P ₂	E	B	D	C	F	G	A
Template	0	1	1	0	0	1	0
Child C ₁	E	B	C	D	G	F	A
Child C ₂	A	B	D	C	E	G	F

Şekil 3.4: Uniform order based crossover (Sastry et al., 2005).

Parent P ₁	A	B	C	D	E	F	G
Parent P ₂	C	B	G	E	F	D	A
Child C ₁	?	?	C	D	E	?	?
Child C ₂	?	?	G	E	F	?	?
Child C ₁	F	A	C	D	E	B	G
Child C ₂	C	D	G	E	F	A	B

Şekil 3.5: Order based crossover (Sastry et al., 2005).

Parent P1	A	B	C	D	E	F	G
Parent P2	C	B	G	E	F	D	A
Child C1	B	G	C	D	E	F	A

Şekil 3.6: Tezdeki çaprazlama.

Mutasyonda Zomaya and Teh (2001) 'in kullandığı *swap mutation* kullanılmıştır. Buna göre çaprazlama sonucu oluşan kromozom üzerinde farklı sanal makine Id lerine sahip iki gen (görev) seçilir. 0 ile 1 aralığında random sayı seçilir. Bu sayı mutasyon oranından küçükse genler arasında değiş tokuş işlemi yapılır.

Tez çalışmasındaki GA' nın genel işleyişi, kayan pencerede bulunan görevler kromozoma aktarılır. Statik çizelgelemede kayan pencerede 10 görev bulunur, dinamik çizelgelemede o ana kadar biriken görevler bulunur, sayısı değişkendir. Populasyon 10 kromozomdan oluşur. Sistemde 5 sanal makine bulunur. İlk popülasyon $Pop_{current}$ için kayan penceredeki görevler random karıştırılarak 10 adet kromozom elde edilir. Bu kromozomlara Bölüm 3.1' de anlatıldığı gibi *vmId* ataması yapılır. $Pop_{current}$ daki tüm kromozomların fitness değeri Formül 3.4 veya 3.5 teki gibi hesaplanır. En iyi kromozom doğrudan Pop_{next} 'e aktarılır. Pop_{next} 'teki diğer 9 kromozomun her biri $Pop_{current}$ 'tan turnuva seçilimi yöntemiyle iki kromozomun seçilerek Bölüm 3.4.1.1' de anlatıldığı gibi çaprazlama işlemi yapılır. Oluşan çocuk Pop_{next} ' e eklenir. Pop_{next} ' teki kromozoma Bölüm 3.1' de anlatıldığı gibi *vmId* ataması yapılır, mutasyon işlemi uygulanır. Tüm bu işlemler Pop_{next} ' teki tüm kromozomlar oluşana kadar devam eder. Toplamda 20 nesil oluşturulur. Son neslin en iyi uygunluk değerine sahip kromozomu sonuç kromozomudur ve burada belirtilen görev-sanal makine ataması gerçekleştirilir.

3.4.2. Tavlama benzetimi

Russel and Norvig (2010)'a göre metalürjide tavlama, metalleri ve camı yüksek bir sıcaklığa ısıtıp ardından yavaş yavaş soğutarak temperlemek veya sertleştirmek için kullanılan ve böylece malzemenin düşük enerjili kristalli bir duruma ulaşmasını sağlayan bir işlemdir. Tavlama benzetimini açıklamak için, engebeli bir yüzeyde bir pipon topunu en derin çukura yerleştirme görevini düşünülebilir. Top yuvarlanması için bırakılırsa, lokal minimumda kalacaktır. Yüzey sallanırsa top lokal minimumdan çıkarılabilir. Önemli olan topu lokal minimumdan çıkaracak kadar yüzeyi sallamaktır, global minimumdan çıkaracak kadar kuvvetli sallamamaktır. Bu örneklemeden yola çıkarak tavlama benzetimi

başlangıçta kuvvetli sallamayla (yüksek sıcaklık) başlar, sonrasında aşamalı bir şekilde sallamanın yoğunluğu (düşük sıcaklık) azaltılır.

```

function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”

  current  $\leftarrow$  MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to  $\infty$  do
    T  $\leftarrow$  schedule(t)
    if T = 0 then return current
    next  $\leftarrow$  a randomly selected successor of current
     $\Delta E \leftarrow next.VALUE - current.VALUE$ 
    if  $\Delta E > 0$  then current  $\leftarrow$  next
    else current  $\leftarrow$  next only with probability  $e^{\Delta E/T}$ 

```

Şekil 3.7: SA sözdekod (Russel and Norwig,2010).

Şekil 3.7’deki döngü kromozom üzerindeki en iyi hareketi seçmek yerine random hareketi seçer. Tez çalışmasında bu hareket swap mutasyon hareketidir. Eğer bu hareket uygunluk değerini yükseltiyorsa her zaman kabul edilir. Aksi durumda $e^{-\Delta E/T}$ ifadesinin 0-1 aralığında rastgele seçilen bir sayıdan büyük olması durumunda hareket kabul edilir. *schedule* fonksiyonu sıcaklığı azaltır. Eğer fonksiyon sıcaklığı yeterince yavaş bir şekilde düşürürse, olasılık 1’e yaklaşırken algoritma global optimumu bulur.

3.4.3. Ağözlü yaklaşım

Tez çalışmasında uygulanan bu yaklaşımda, çizelgeme aşamasında kayan pencerede bulunan görevler boyutuna (*length*) göre büyükten küçüğe sıralanır. O anda en az yüke sahip olan makine bulunup görev çizelgelenmesi için o makineye gönderilir. Makinenin yükü güncellenir. Bu işlem kayan penceredeki tüm görevler makinelere gönderilene kadar devam eder. Bu işlemde bir rastgelelik yoktur, aynı veri seti için her çalışmada aynı sonucu (aynı görevlerin aynı makinelere atanması) verir.

3.4.4. Diferansiyel gelişim algoritması

Diferansiyel gelişim (DE) algoritması temelde üç avantaja sahip yeni bir sezgisel yaklaşımdır: başlangıçtaki parametre değerlerine bakılmaksızın gerçek minimum değeri bulmak, hızlı yakınsama ve az sayıda kontrol parametresi kullanmak. DE algoritması, benzer operatörleri kullanan (çaprazlama, mutasyon ve seçim) genetik algoritmalar gibi bir popülasyon bazlı bir algoritmadır. Daha iyi çözümler inşa etmedeki ana fark, DE mutasyon işlemine dayanırken GA'nın çaprazlamaya dayanmasıdır. Bu ana işlem, popülasyondaki rastgele örneklenmiş çözüm çiftlerinin farklılıklarına dayanmaktadır (Karaboğa and Ökdem, 2004). Tez çalışması için kullanılan DE'nin temel adımları şöyledir:

```

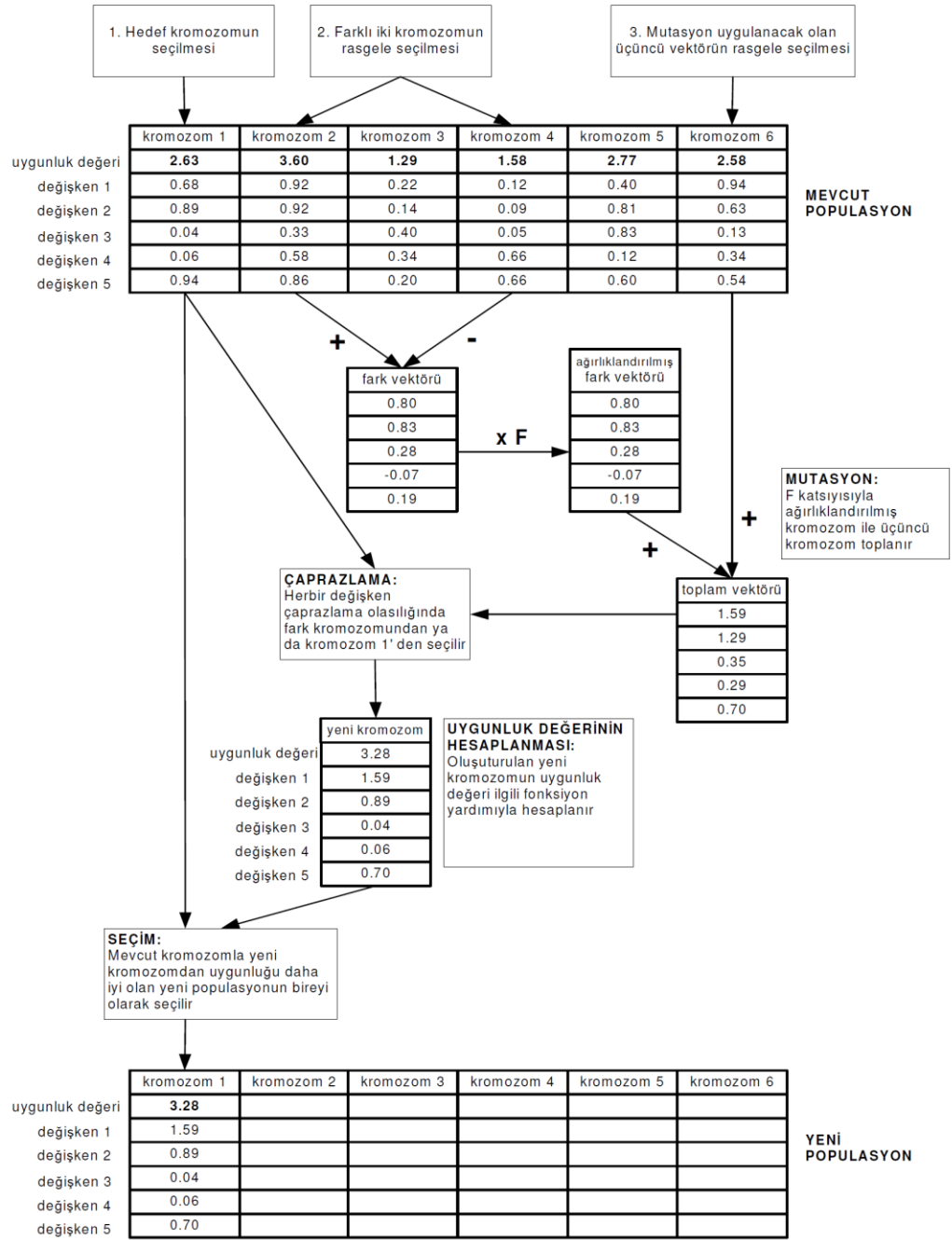
Begin
  Initialization
  Evaluation
  Forward Transformation
  Repeat
    Mutation
    CrossOver
    Backward Transformation
    Evaluation
    Selection
  Until (termination criteria are met)
End

```

Şekil 3.8' de temel adımlar anlaşılır bir biçimde gösterilmiştir. Tez çalışmasında ilk popülasyon, kayan penceredeki görevlerin karılmasıyla oluşur. Bu yaklaşımda her görevin *normalizedID* özelliği vardır ve *cloudletId* gibi benzersizdir. Burada amaç görevlerin id lerini [0-1] aralığında tutmaktır. Kayan penceredeki her görevin *normalizedID* lerinin hesaplanmasına *forward transformation* adı verilir. Burada Engelbrecht and Pampara (2007) 'nın çalışmasından yararlanılmıştır. Bu işlem için Formül 3.6 kullanılmıştır. Görevler t_i şeklinde gösterilmiştir.

$$normalizedID(t_i) = \frac{cloudletID(t_i) - cloudletID(t_{min})}{cloudletID(t_{max}) - cloudletID(t_{min})} \quad (3.6)$$

Sonrasında mutasyon işlemi için Formül 3.7 kullanılmıştır. Genlerle yapılan tüm işlemler *normalizedID*' ler üzerinden yapılır. Formüllerde kullanılan parametreler aşağıda belirtilmiştir.



Şekil 3.8: DE' nin adımları (Kesintürk, 2006).

Parametreler

NP : popülasyon büyüklüğü (kromozom sayısı) $NP \geq 4$

D : değişken sayısı (gen sayısı) (1, 2, 3, ...j)

CR : çaprazlama oranı $[0,1-1,0] = 0,8$

G : nesil (1, 2, 3,... $G_{\max}$)

F : ölçekleme faktörü $[0-2] = 0,8$

$x_{j,i,G}$: G neslinde, i kromozomunun j parametresi (gen)

$n_{j,i,G+1}$: mutasyon ve çaprazlamaya tabi tutulmuş ara kromozom

$u_{j,i,G+1}$: $x_{j,i,G}$ den bir sonraki nesil için üretilen kromozom $r_{1,2,3}$:

yeni kromozomun üretilmesinde kullanılacak rasgele seçilmiş kromozomlar

$r_{1,2,3} \in \{1,2,3,...NP\}$ $r_1 \neq r_2 \neq r_3 \neq i$

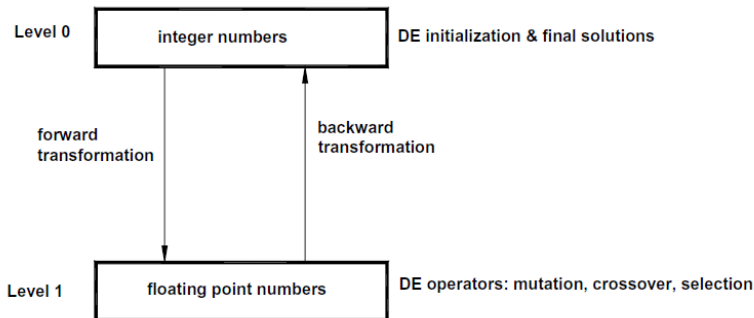
Mutasyon Formülü:

$$\forall j \leq D: n_{j,i,G+1} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \quad (3.7)$$

Çaprazlama Formülü:

$$\forall j \leq D: x_{j,u,G+1} \text{ (trial kromozom)} = \begin{cases} x_{j,n,G+1} \text{ (mutant kromozom)}, & \text{eğer } rand[0,1] \leq CR \vee j = j_{rand} \\ x_{j,i,G} \text{ (mevcut kromozom)} & , \text{ aksi durumda} \end{cases} \quad (3.8)$$

Formül 3.8' deki formüle göre çaprazlama işlemi yapılır. Bunun sonucunda elde edilen trial kromzom, ondalıklı sayılardan oluşmaktadır. Bu kromozomun görevlerden oluşan kromozoma dönüştürülmesi için *backward transformation* yapılmalıdır . Şekil 3.9' da dönüşümler gösterilmiştir.



Şekil 3.9: DE’de dönüşüm işlemleri (Onwubolu and Davendra, 2006).

Geriye dönüşüm’de izlenen yöntem şöyledir. Kayan penceredeki görevler *cloudletID* ye göre küçükten büyüğe sıralanır. Trial vektördeki ondalıklı sayılar küçükten büyüğe sıralanır ve bu şekilde bire bir eşleştirilir. Böylece trial vektördeki her sayıya karşılık gelen *cloudletID* bulunur ve buna göre trial vektördeki sıralama korunarak görevlerden oluşan kromozom oluşturulur.

Değerlendirme aşamasında, oluşan kromozoma, *vmID* ler atanır ve önceki bölümlerde anlatıldığı gibi uygunluk fonksiyonu hesaplanır. Sonraki nesle aktarılacak kromozomun seçimi Formül 3.9’ da belirtilmiştir.

$$\forall i \leq NP: x_{i,G+1} = \begin{cases} x_{u,G+1} & , \text{ eğer } f(x_{u,G+1}) \geq f(x_{i,G}) \\ x_{i,G} & , \text{ diğer durumlarda} \end{cases} \quad (3.9)$$

3.4.5. SAGA

SA çalışırken en iyi uygunluk değerine sahip 10 kromozomu tutmak için boyutu 10 olan bir öncelik kuyruğu kullanılır. SA da yeni kromozom, kabul edilebilir ise bu kromozom kuyruğa eklenir. Kuyruğa ekleme işlemi uygunluk değerine göre yapılır. SA tamamlandıktan sonra kuyruktaki kromozomlardan bir popülasyon oluşturulur. Bu popülasyon GA için ilk popülasyon olarak kullanılır. GA bu popülasyon üzerinde 20 nesil üretecek kadar çalıştırılır, son popülasyondaki en iyi kromozoma göre görev-makine ataması yapılır. SAGA’ nın işleyişi Şekil 3.10’da gösterilmiştir.

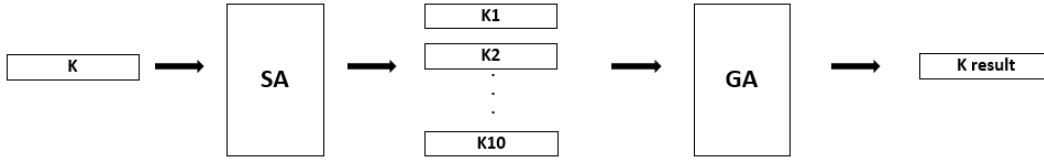
3.4.6. GASA

GA 20 nesil çalıştırılır ve sonuç popülasyonundaki her bir kromozoma SA uygulanır. SA’nın ilk sıcaklığı 100.000, soğuma oranı 0,003’ tür. Bunun sonucunda oluşan en iyi kromozoma göre görev-makine ataması yapılır. GASA’ nın işleyişi Şekil 3.11’ de gösterilmiştir.

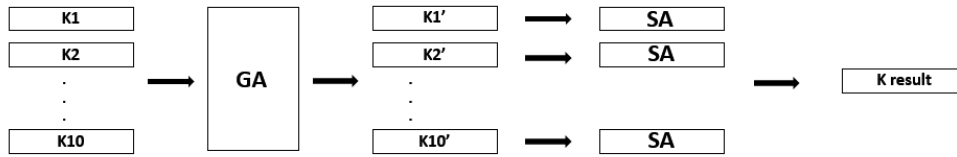
3.4.7. DESA

DE çalıştırılır ve sonuç popülasyonundaki her bir kromozoma SA uygulanır. SA’nın ilk sıcaklığı 100.000, soğuma oranı 0,003’ tür. Bunun sonucunda oluşan en

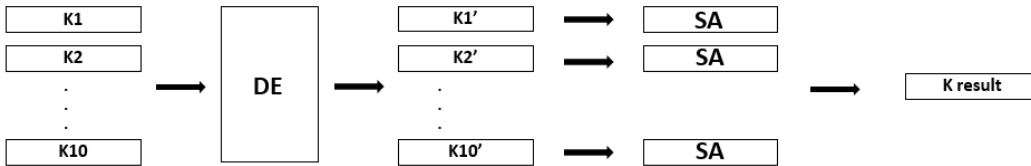
iyi kromozoma göre görev-makine ataması yapılır. DESA'nın işleyişi Şekil 3.12' de gösterilmiştir.



Şekil 3.10: SAGA işleyişi.



Şekil 3.11: GASA işleyişi.



Şekil 3.12: DESA işleyişi.

3.5. Statik Çizelgeleme Adımları

Bölüm 3.4' te anlatılan tüm yaklaşımlar statik ve dinamik çizelgelemenin ilgili kısımlarında kullanılmıştır.

Statik çizelgelemede, 20.000-60.000 aralığında rastgele belirlenen büyüklükteki görevler görev havuzunda bulunmaktadır. Tüm görevler mevcuttur. Çizelgeleme işlemi sırasında yeni gelen görev olmaz. 5 sanal makine bulunmaktadır. Simülasyon başlatıldığında ilk olarak 5 makineye aynı anda 5 görev (her makineye 1 görev düşecek şekilde) gönderilir.

Adım 1: Sistemin anlık durumu. Sistemdeki makinelerden biri boş konuma düştüğünde yani makinenin *executionList* ve *waitingList* inde hiç görev kalmadığında çizelgeme başlar. Bu anda sistemin zamanı durur (yeni event' e geçtiği için) ve bu adımlar sonundaki sonuca göre görevler *submit* edilene kadar sistem zamanı ilerlemez. Amaç boşta makine kalmasını mümkün mertebede engellemek, makinanın boşta kaldığı süreyi azaltmaktır. Sisteme gönderilecek görevler belirlenmelidir.

Adım 2: Kayan pencerenin doldurulması. Sisteme gönderilecek görevler, görev havuzundan alınarak kayan pencereye eklenir. Kayan pencerenin boyutu sabittir ve kromozomdaki gen sayısına eşittir. Kayan pencere tamamen boş olabilir veya bazı hücreler dolu olabilir. Tüm hücreleri dolu olacak şekilde pencere doldurulur.

Adım 3: Metasezgisel algoritmanın uygulanması. Kayan penceredeki görevlerin kromozoma aktarılmasıyla bölüm 3.4'teki ilgili meta sezgisel algoritma çalıştırılır ve görev-makine atama eşleşmesi (çözüm önerisi) oluşur.

Adım 4: Alt ve üst eşik değerlerinin bulunması. Bu değerleri bulmak için *ortalamaYük* bulunmalıdır. *ortalamaYük* ise zaman cinsinden; kayan penceredeki görevlerin yükleri, makinelerin işlemekte oldukları görevlerden kalan kısmının yükleri ve makinelerin *waitingList* 'lerindeki görevlerin yüklerinin toplanarak makine sayısına bölünmesiyle elde edilir. L_{ave} şeklinde gösterilir. Alt eşik değeri T_L , üst eşik değeri T_H olarak gösterilmektedir. Zomaya and Teh (2001)' in uyguladığı Formül 3.10'a göre hesaplanır.

$$\begin{aligned} T_H &= 1,2 \times L_{ave} \\ T_L &= 0,8 \times L_{ave} \end{aligned} \quad (3.10)$$

Adım 5: Her bir makinenin kabul edilebilir yükünün hesaplanması. Formül 3.11'e göre her makinenin kabul edilebilir yükü hesaplanır. Her makinenin toplam kuyruk yükü, Adım 4' teki sonuç kromozomuna göre hesaplanır.

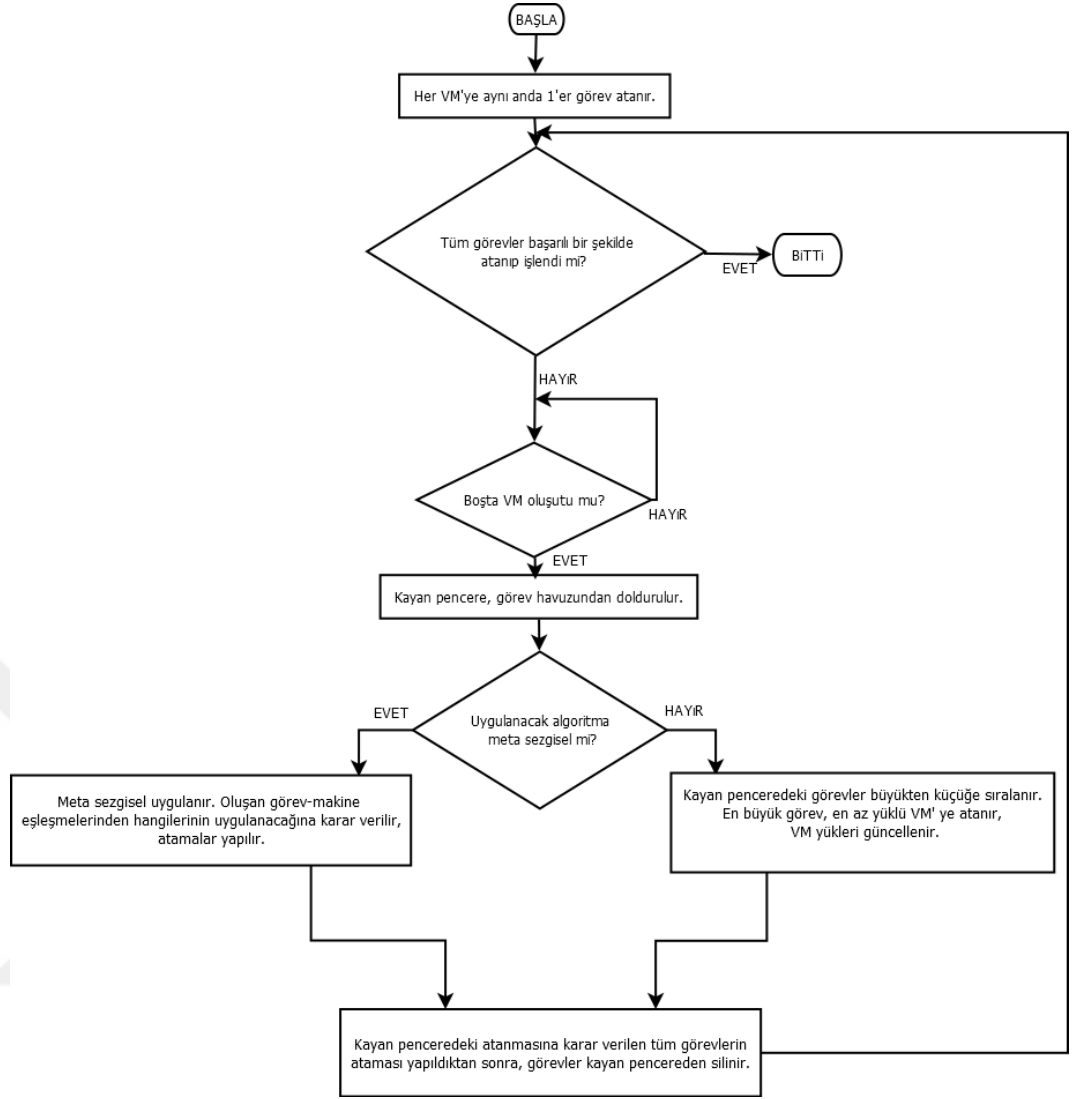
$$Kabul\ Edilebilir\ Y\ddot{u}k_j \leq T_H - makine_j' \text{ nin toplam kuyruk y\ddot{u}k\ddot{u}} \quad (3.11)$$

Adım 6: Atama kararının verilmesi. Her bir makinenin kabul edilebilir yükü bulunur, adım 4 teki sonuç kromozomunda her makineye atanan görevlerin yükleri toplanır, durum Formül 3.11' e uyuyorsa atama yapılır, atanan görevler kayan pencereden silinir, uymuyorsa o makineye atanacağı belirtilen görevler atanamaz, kayan pencerede kalır.

* Bu adımlar görev havuzunda hiç görev kalmayıncaya kadar devam eder. En son görev havuzunda kayan pencereyi dolduracak kadar görev kalmazsa, penceredeki ilk görev yükü en az olan makinaya atanır ve makinaların yükleri güncellenir. Bu şekilde tüm görevler bitene kadar devam eder.

* Bu sistem, makinelerden birinin boş konuma geçmesiyle faaliyete geçmektedir. Amaç da boş konuma geçen makineye bir an önce atama yapmak olduğu için, eğer meta sezgisel algoritma sonucu oluşan atama eşleşmesinde boş konumdaki makine bulunmuyorsa, boş konumdaki makine; atama eşleşmesinde yer alana kadar meta sezgisel tekrar çalıştırılır.

Statik çizelgeleme akış şeması Şekil 3.13' de gösterilmiştir.



Şekil 3.13: Statik görev çizelgeleme akış şeması

3.6. Dinamik Çizelgeleme Adımları

Dinamik çizelgelemede, 20.000-60.000 aralığında rastgele belirlenen büyüklükteki ve 5-10 birim aralığında rastgele değişen aralıkların ard arda eklenmesiyle oluşan zaman birimlerinde sisteme gelen görevler bulunmaktadır. Çizelgeleme işlemi başladıktan sonra da görevler sisteme gelmeye devam eder. 5 sanal makine bulunmaktadır. Simülasyon başlatıldığında ilk olarak 5 makineye aynı anda 5 görev (her makineye 1 görev düşecek şekilde) gönderilir.

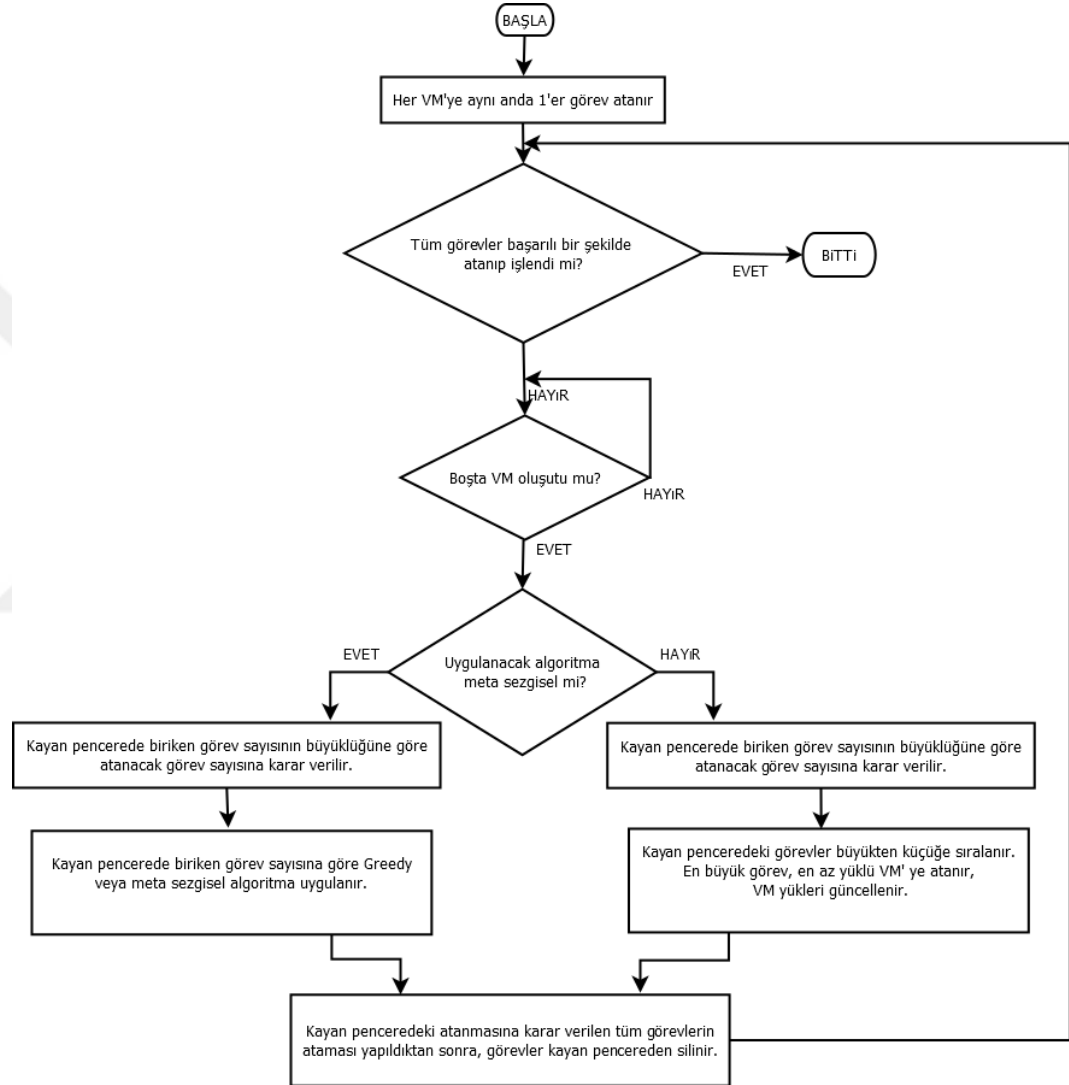
Adım 1: Sistemin anlık durumu. Sistemdeki makinelerden biri boş konuma düştüğünde yani makinenin *executionList* ve *waitingList* inde hiç görev kalmadığında çizelgeme başlar. Bu anda sistemin zamanı durur (yeni event' e geçtiği için) ve bu adımlar sonundaki sonuca göre görevler *submit* edilene kadar sistem zamanı ilerlemez. Amaç boşa makine kalmasını mümkün mertebede engellemek, makinenin boşa kaldığı süreyi azaltmaktır. Sisteme gönderilecek görevler belirlenmelidir.

Adım 2: Kayan penceredeki görev sayısına göre uygulanacak algoritmaya karar verilmesi. Sistem devam ettiği sürece görevler gelmeye devam eder, gelen görevler kayan pencereye eklenir. Adım 1'de belirtildiği gibi herhangi bir makine boş konuma düştüğünde kayan penceredeki görev sayısına bakılır. Kayan pencerenin boyutu değişkendir, çünkü sistem ne kadar görev geleceği bilgisine sahip değildir. Kayan penceredeki görev sayısı; sanal makine sayısının 1 katı veya daha azı ise Greedy uygulanır, sanal makine sayısının 1 katından fazla ve 4 katından az ise ilgili meta-sezgisel uygulanır, sanal makine sayısının 4 katından fazla ise 4 katı kadarına ilgili meta-sezgisel uygulanır artan görevler kayan pencerede kalır, diğer seferde değerlendirilir.

* Boşa makine bulunduğunda kayan pencerede hiç görev birikmemiş ise boştaki makineyi kaybetmemek için makineID'si listede tutulur. Greedy algoritmasında atama yapılırken öncelikle bu listede bulunan makineler göz önünde bulundurulur. Meta sezgisel algoritmalarda tüm makinelere görev atama eşleşmesi yapıldığı için bir listeye gerek yoktur.

* Greedy algoritmasında, makineID listesindeki makinelere atama eşleşmesi yapıldığı için, çizelgelemeyi başlatan boştaki makineye atama yapılmamışsa, bu makine ID'si de listeye eklenir ve Greedy' de yapılan atama eşleşmelerinde öncelikle bu listedeki makineler göz önünde bulundurulur.

Dinamik çizelgeleme akış şeması Şekil 3.14' de gösterilmiştir.



Şekil 3.14: Dinamik görev çizelgeleme akış şeması

4. DEĞERLENDİRME ÖLÇÜTLERİ VE KULLANILAN VERİ SETLERİ

Bu bölümde tez çalışmasında kullanılan değerlendirme ölçütleri açıklanmıştır. Tez çalışması için oluşturulan veri setinden bahsedilmiştir.

4.1. Değerlendirme Ölçütleri

Singh et al. (2017) ve Kalra and Singh (2015) ' e göre bulut sistemlerde temel olarak iki aktör vardır. Bunlar; kullanıcı ve servis sağlayıcıdır. Bu durumda optimizasyon, her iki taraf için farklı ölçütler üzerinden değerlendirilmektedir. Kullanıcı tarafındaki ölçütler: tamamlanma zamanı, maliyet, bekleme zamanı, son tarih, güvenlik, güvenilirlik. Servis sağlayıcı tarafındaki ölçütler: yük dengeleme, kaynak kullanımı, erteleme, enerji verimliliğidir. Bulut bilişimde görev çizelgelemesi için kullanılan bazı meta-sezgisel teknikler incelenip yukarıda anlatılan değerlendirme ölçütleri dikkate alınarak analiz edildiğinde, bu çalışmalar içerisinde en çok kullanılan ölçütler; tamamlanma zamanı(%40), maliyet(%14), son tarih (%14), yük dengeleme(%14) olarak belirtilmiştir.

Bu tez çalışması kapsamında her algoritma için tamamlanma zamanı, minimum tamamlanma zamanı, yük dengelemeyle ilgili olarak makinelerin tamamlanma zamanlarının standart sapma ve varyans değerleri performans ölçütü olarak kullanılmıştır.

4.1.1. Tamamlanma zamanı

Görevlerin, yürütülmesinin tamamladığı zamanı ve gönderildiği zamanı göz önüne alarak tüm iş akışını yürütmek için gereken toplam süre olarak tanımlanır. Kullanıcının iş akışını gönderdiği zaman ile iş akışının tamamlanıp sonuç verdiği zaman arasındaki süreç olarak tanımlanabilir. Sistemdeki son görevin bitme zamanıdır. Formül 4.1' de gösterilmiştir.

$$Makespan = \max_{i \in vm} \{CT_i\}, \text{ where } CT_i \text{ denotes the completion time of } vm_i \quad (4.1)$$

4.1.2. Minimum tamamlanma zamanı

Her bir sanal makinenin kendilerine atanan tüm görevleri tamamlama zamanları içindeki en küçük tamamlama zamanıdır. Formül 4.2' de gösterilmiştir. Yük ne kadar dengeli dağıldıysa minimum tamamlanma zamanı ile tamamlanma zamanı değerleri birbirine o kadar yakın olur.

$$\begin{aligned} \text{Minimum Completion Time} = \\ \min_{i \in \text{vm}} \{CT_i\}, \text{ where } CT_i \text{ denotes the completion time of } \text{vm}_i \end{aligned} \quad (4.2)$$

4.1.3. Sanal makine yüklerinin standart sapması

Formül 4.3' te gösterilmektedir. Formülde; N: sanal makine sayısını, x_i : i . sanal makinenin yükünü, $\bar{x}$: makinelerin yüklerinin ortalamasını belirtir.

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (4.3)$$

4.1.4. Sanal makine yüklerinin varyansı

Varyans, standart sapmanın karesidir. Standart sapma ve varyans, merkezi dağılım ölçüsüdür. Yük dengelenmesinin ölçütleri standart sapma ve varyanstır.

4.2. Kullanılan Verisetleri

Tez kapsamında kullanılan veri seti tek biçimli dağılım (uniform distribution) la oluşturulmuştur. 50, 100, 200, 300, 400, 500 adet görevden (cloudlet) oluşan görev gruplarının her birinden 20 şer adet oluşturulmuştur (50 görevden oluşan 20 metin dosyası, 100 görevden oluşan 20 metin dosyası...). Oluşturulan görevlerin büyüklükleri 20000 ile 60000 arasında, sisteme gelme zaman aralıkları 5 ile 10 arasında random değerlerden oluşturulmuştur, *pes* yani kullandıkları CPU sayısı 1 dir. Her bir algoritma, oluşturulan her görev grubunun metin dosyasında 5 kez çalıştırılmıştır. Elde edilen sonuçların (bölüm 4.1'deki ölçütler) aritmetik

ortalaması alınarak ilgili görev grubunun sonucuna ulaşılmıştır. Meta sezgisel algoritmalarındaki rastgelelikten dolayı her çalışmada farklı sonuç elde edildiği için böyle bir yöntem izlenmiştir.



5. DENEYSEL SONUÇLAR

Bu bölümde, değişen görev sayılarında algoritmaların önceki bölümde açıklanan değerlendirme ölçütlerinin sonuçları gösterilmiştir. Öncelikle sistem ve algoritma parametrelerinden bahsedilmiştir sonrasında sırayla statik ve dinamik çizelgeleme sonuçlarına yer verilmiştir.

5.1. Parametreler

Tezin çalışması için kullanılan sistem parametreleri ve tez çalışması kapsamında uygulanan algoritmaların parametreleri açıklanmıştır.

5.1.1. Sistem parametreleri

Bu tez çalışması; intel(R) Core(TM) i7-3517U CPU@ 1.90GHz 2.40 GHz işlemciye ve 6.00GB hafızaya sahip bilgisayar üzerinde Eclipse Oxygen.3 Release (4.7.3) geliştirme ortamında CloudSim 4.0 simülatörü üzerinde yapılmıştır.

Simülasyon ortamında 1 datacenter içerisinde 1 host ve host içerisinde 5 sanal makine bulunmaktadır. Datacenter, host, sanal makine ve cloudlet (görev)' lerin parametreleri ve değerleri Çizelge 5.1'de gösterilmiştir.

5.1.2. Algoritma parametreleri

Algoritma parametreleri Çizelge 5.2' de gösterilmiştir. Çizelgede belirtilmeyen melez algoritmalar (GASA, SAGA, DESA), çizelgedeki ilgili tekli algoritmaların art arda uygulanmasıyla oluşur, parametreleri çizelgedeki gibidir.

Çizelge 5.1: Simülasyon parametreleri.

KAYNAK	PARAMETRE	DEĞER
	Datacenter sayısı	1
	Datacenterdaki host sayısı	1
	Hosttaki sanal makine sayısı	5
	Cloudlet sayısı	50-500
DATACENTER	Sistem mimarisi	x86
	İşletim sistemi	Linux
	Sanal makine modeli	Xen
	Saat Dilimi	10
HOST	Ram	16 GB
	Depolama	1TB
	Bant genişliği	10 GB/s
	İşleme gücü	5000 mips
	İşlemci sayısı	1
	Sanal makine çizelgeleme türü	Zaman paylaşımli
SANAL MAKİNE	Depolama	10 GB
	Ram	0,5 GB
	Bant genişliği	1 GB/s
	İşleme gücü	1000 mips
	İşlemci sayısı	1
	Cloudlet çizelgeleme türü	Alan paylaşımli
CLOUDLET	Boyut	20000-60000 mi
	Gerekli işlemci sayısı	1

Çizelge 5.2: Algoritma parametreleri.

ALGORİTMA	PARAMETRE	DEĞER
GA	Populasyon boyutu	10
	İterasyon Sayısı	20
	Mutasyon Oranı	0,015
SA	İlk sıcaklık	100.000
	Son sıcaklık	1
	Soğutma oranı	0,003
DE	Populasyon boyutu	10
	İterasyon Sayısı	20
	Çaprazlama Oranı	0,8
	F sabiti	0,8

5.2. Algoritmalar Göre Ölçütler Üzerinden Statik Çizelgeleme Sonuçları

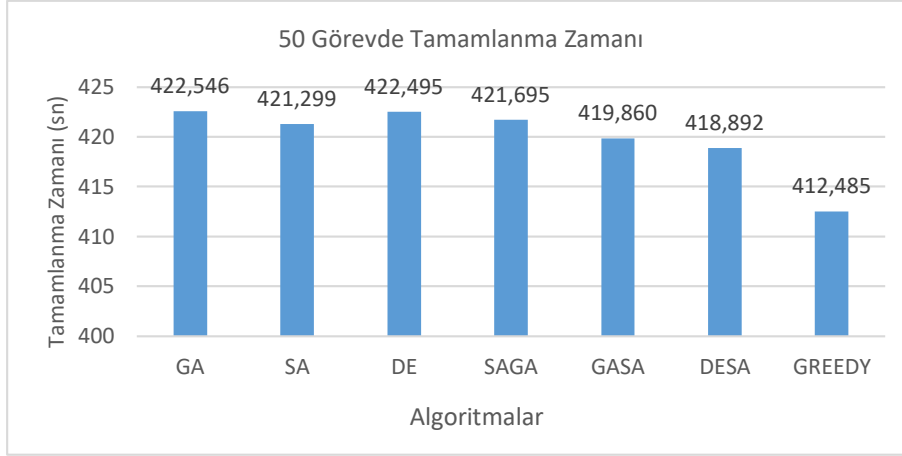
50-100-200-300-400-500 görev sayılarında değerlendirme ölçütlerinin değişimi gösterilmiştir.

5.2.1. Tamamlanma zamanı

Statik çizelgeleme için tamamlanma zamanı sonuçları, uygulanan 7 algoritma için değişen görev sayıları üzerinden Çizelge 5.3' te belirtilmiştir. Buna ek olarak her görev grubu için minimum teorik limit hesaplanmıştır. Bu değer, ilgili görev grubundaki her bir görevin yürütme sürelerinin toplanarak sanal makine sayısına bölünmesiyle elde edilir.

Çizelge 5.3: Değişen görev sayıları üzerinden algoritmalar göre *tamamlanma zamanı* değerleri tablosu.

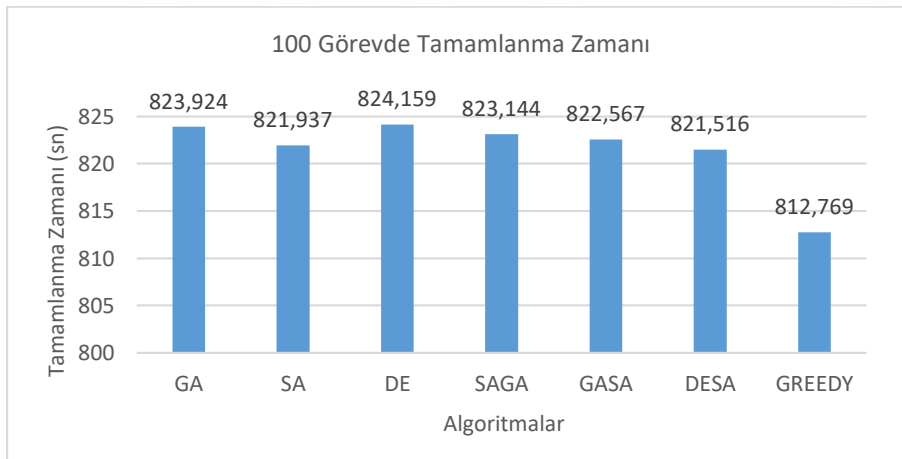
		GA	SA	DE	SAGA	GASA	DESA	GREEDY	Teorik Limit
50 GÖREV	AVG	422,546	421,299	422,495	421,695	419,860	418,892	412,485	403,416
	SD	5,164	6,472	6,154	6,240	4,899	4,643	0,000	
100 GÖREV	AVG	823,924	821,937	824,159	823,144	822,567	821,516	812,769	803,497
	SD	6,007	6,663	6,064	7,044	6,755	6,318	0,000	
200 GÖREV	AVG	1621,465	1619,814	1622,054	1620,549	1620,463	1619,521	1610,810	1601,180
	SD	5,608	7,278	5,471	7,102	7,073	9,471	0,000	
300 GÖREV	AVG	2420,631	2417,785	2419,460	2419,486	2419,848	2420,023	2407,535	2399,672
	SD	5,706	6,640	6,518	6,906	8,012	8,572	0,000	
400 GÖREV	AVG	3239,664	3237,336	3240,256	3238,262	3239,350	3240,572	3228,957	3218,351
	SD	8,519	6,460	6,107	6,522	7,992	7,952	0,000	
500 GÖREV	AVG	4028,279	4026,112	4028,075	4027,754	4029,680	4028,721	4016,056	4007,696
	SD	5,669	6,912	5,113	6,355	7,283	6,407	0,000	



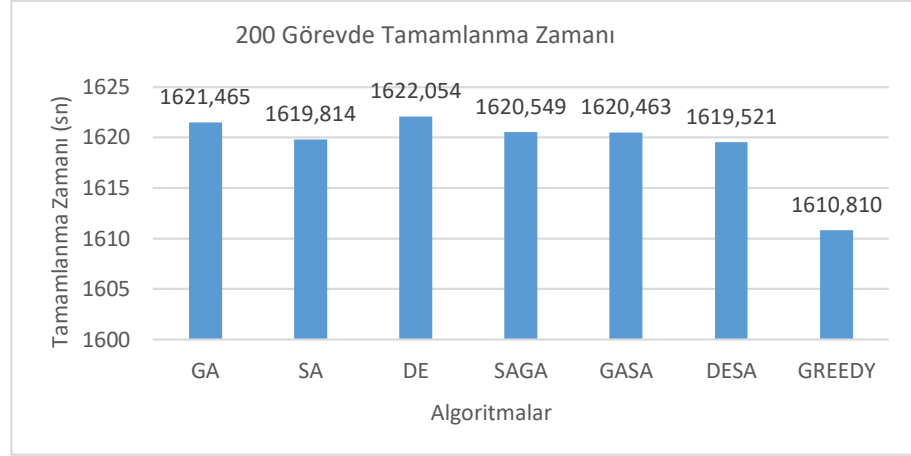
Şekil 5.1: 50 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

50 adet görev için sonuçlar Şekil 5.1’deki gibidir. En düşük tamamlanma zamanını veren algoritmanın Greedy, sonrasında DESA olduğu görülmüştür.

100 adet görev için sonuçlar Şekil 5.2’de gösterilmiştir. En düşük tamamlanma zamanını veren algoritmaların sırasıyla Greedy, DESA ve SA olduğu görülmektedir.



Şekil 5.2: 100 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

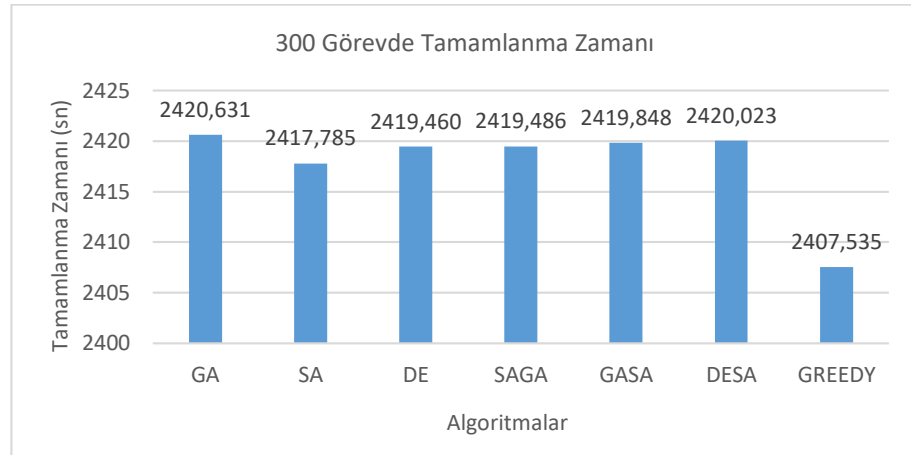


Şekil 5.3: 200 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

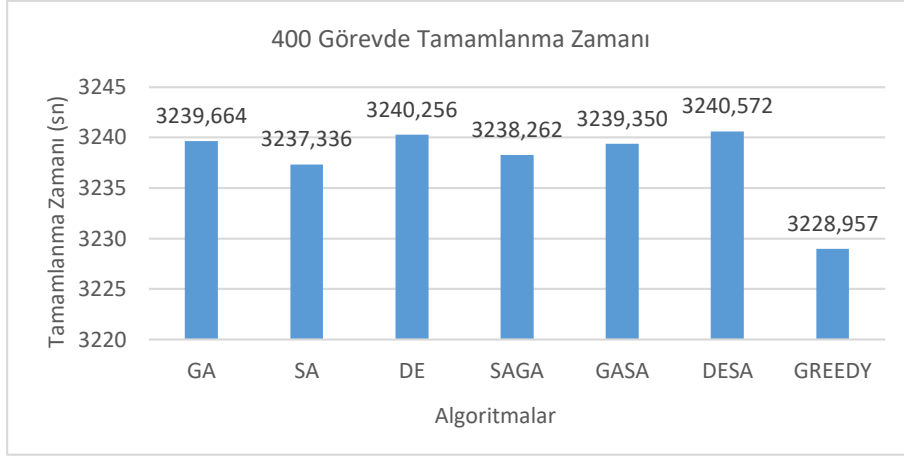
200 adet görev için sonuçlar Şekil 5.3'te gösterilmiştir. En düşük tamamlanma zamanını veren algoritmalar sırasıyla Greedy, DESA ve SA' dır.

300 adet görev için sonuçlar Şekil 5.4'te gösterilmiştir. En düşük tamamlanma zamanını veren algoritmalar sırasıyla Greedy ve SA' dır.

400 adet görev için sonuçlar Şekil 5.5'te gösterilmiştir. En düşük tamamlanma zamanını veren algoritmalar sırasıyla Greedy ve SA' dır.



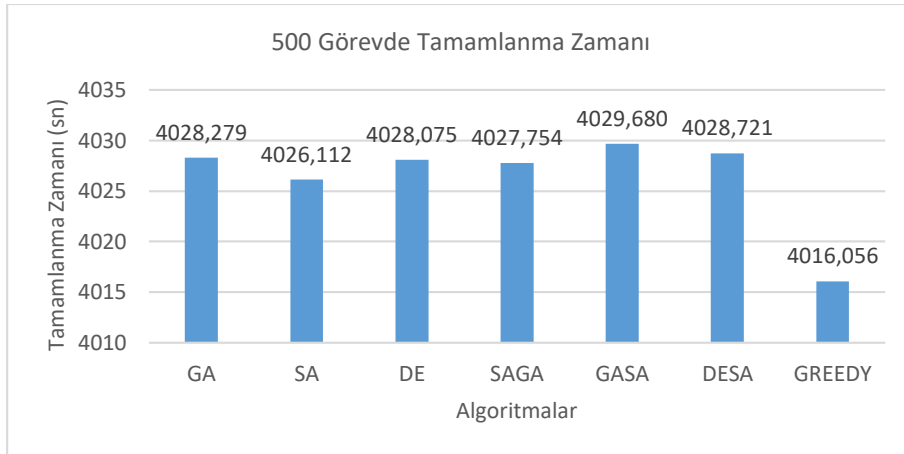
Şekil 5.4: 300 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.



Şekil 5.5: 400 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

500 adet görev için sonuçlar Şekil 5.6'da gösterilmiştir. En düşük tamamlanma zamanını veren algoritmalar sırasıyla Greedy ve SA' dır.

Statik çizelgelemeye tamamlanma zamanı ölçütü üzerinden genel olarak bakıldığında tüm görev sayılarında en düşük sonucu Greedy yaklaşımda vermiştir. 50, 100 ve 200 adet görevle yapılan deneylerde ise ikinci en düşük sonucu hibrit olarak bu tez çalışması kapsamında geliştirilen DESA algoritması vermektedir. 300, 400 ve 500 adet görevle yapılan deneylerde ise ikinci en düşük sonucu SA algoritması vermektedir.



Şekil 5.6: 500 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

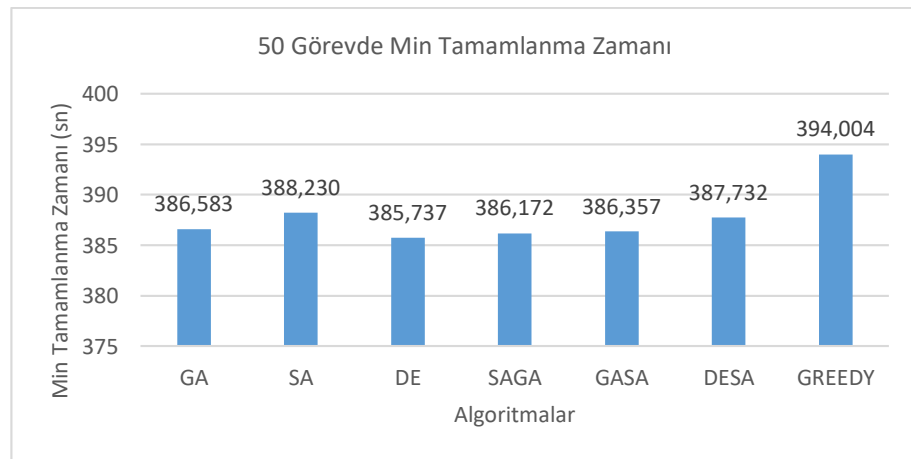
5.2.2. Minimum tamamlanma zamanı

Statik çizelgeleme için minimum tamamlanma zamanı sonuçları, uygulanan 7 algoritma için değişen görev sayıları üzerinden Çizelge 5.4’ te belirtilmiştir. Çizelgede minimum teorik limit belirtilmiştir.

Çizelge 5.4: Değişen görev sayıları üzerinden algoritmalarla göre *minimum tamamlanma zamanı* değerleri tablosu.

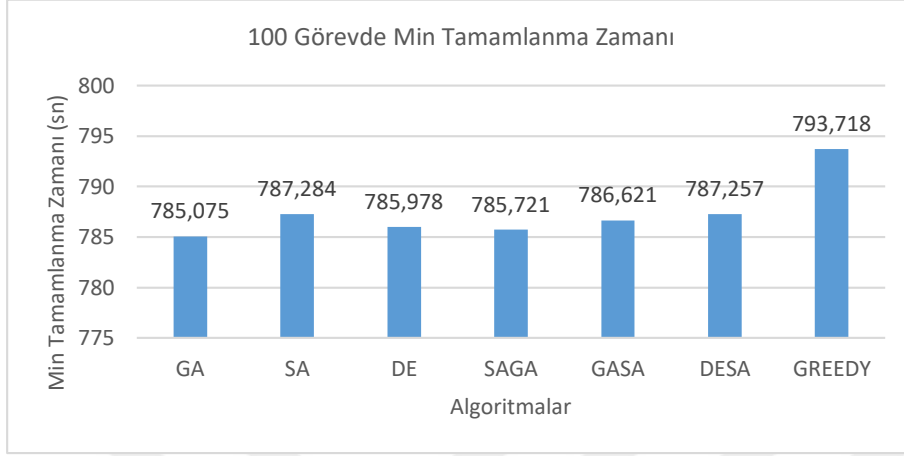
		GA	SA	DE	SAGA	GASA	DESA	GREEDY	Teorik Limit
50 GÖREV	AVG	386,583	388,230	385,737	386,172	386,357	387,732	394,004	403,416
	SD	5,259	4,801	5,018	6,137	6,169	4,593	0,000	
100 GÖREV	AVG	785,075	787,284	785,978	785,721	786,621	787,257	793,718	803,497
	SD	5,267	5,423	5,690	5,507	5,783	6,502	0,000	
200 GÖREV	AVG	1585,207	1584,185	1583,514	1584,209	1584,321	1586,733	1592,108	1601,180
	SD	4,996	6,687	4,978	5,675	6,430	7,327	0,000	
300 GÖREV	AVG	2381,156	2383,068	2381,983	2382,470	2381,850	2381,889	2390,989	2399,672
	SD	4,848	5,800	5,205	5,649	7,244	6,980	0,000	
400 GÖREV	AVG	3200,982	3202,012	3199,540	3199,930	3201,887	3201,491	3208,454	3218,351
	SD	4,806	5,778	5,618	5,051	7,266	5,954	0,000	
500 GÖREV	AVG	3989,469	3992,883	3989,914	3989,285	3989,198	3988,667	3997,926	4007,696
	SD	5,274	4,631	5,614	5,797	6,782	6,596	0,000	

50 adet görev için sonuçlar Şekil 5.7’ deki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla DE, SAGA ve GASA olduğu görülmüştür.

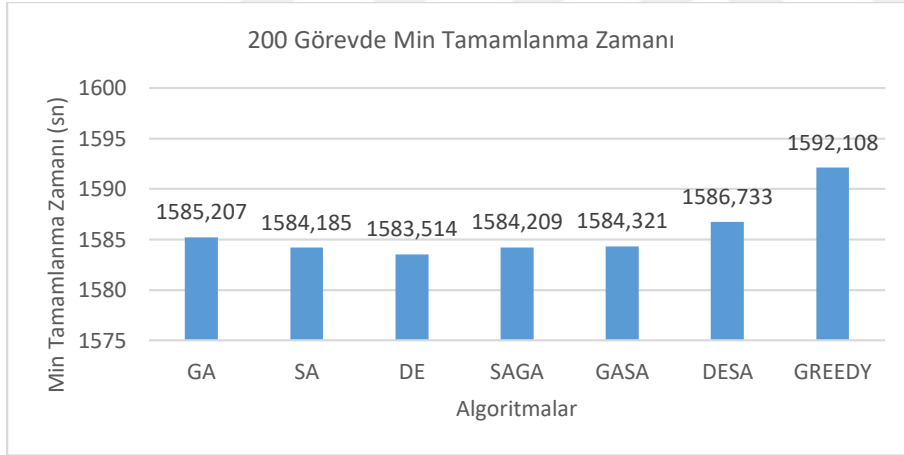


Şekil 5.7: 50 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

100 adet görev için sonuçlar Şekil 5.8’ deki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla GA, SAGA ve DE olduğu görülmüştür.



Şekil 5.8: 100 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

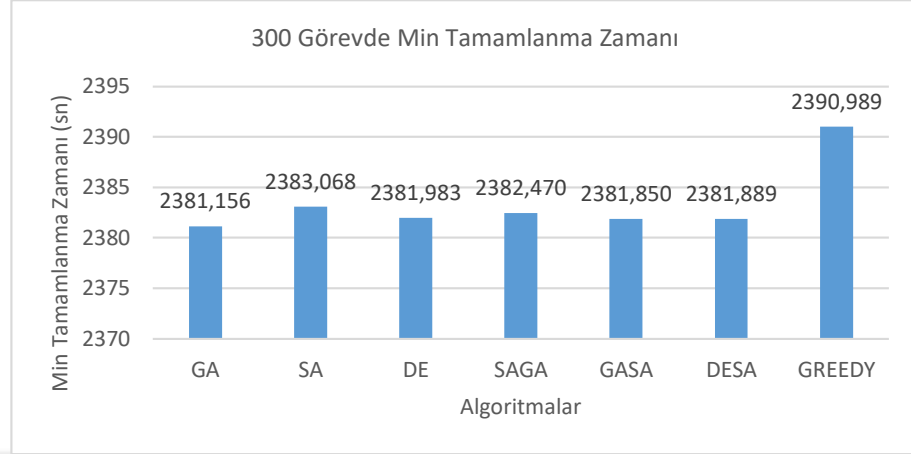


Şekil 5.9: 200 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

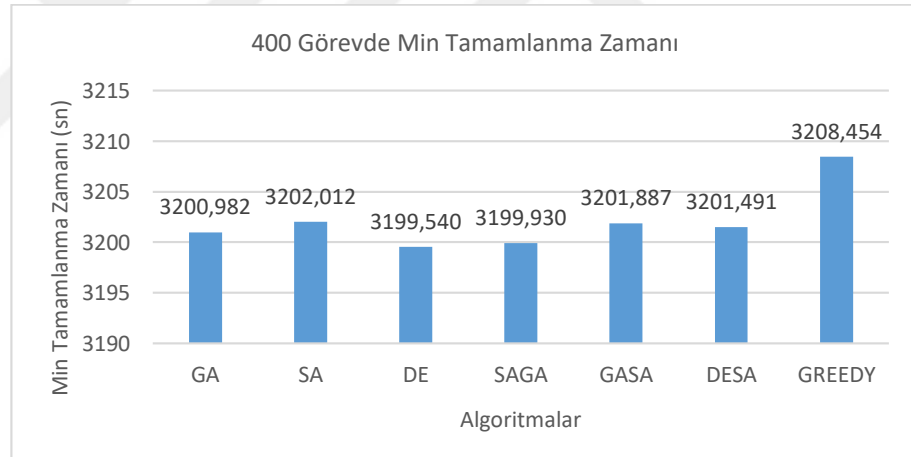
200 adet görev için sonuçlar Şekil 5.9’ deki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla DE, SA ve SAGA dır.

300 adet görev için sonuçlar Şekil 5.10’ daki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla GA, GASA ve DESA dır.

400 adet görev için sonuçlar Şekil 5.11’ deki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla DE, SAGA ve GA dır.

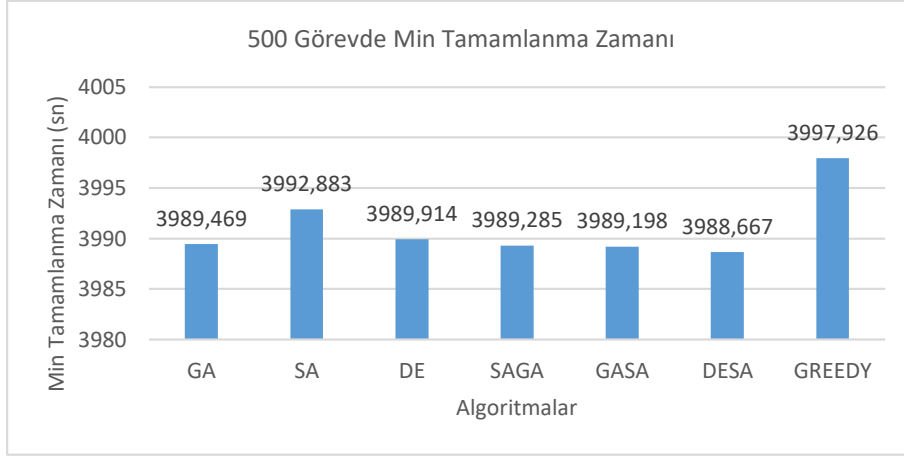


Şekil 5.10: 300 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.



Şekil 5.11: 400 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

500 adet görev için sonuçlar Şekil 5.12’ deki gibidir. En düşük minimum tamamlanma zamanını veren algoritmaların sırasıyla DESA, GASA ve SAGA dır.



Şekil 5.12: 500 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

Statik çizelgelemeye minimum tamamlanma zamanı ölçütü üzerinden genel olarak bakıldığında 6 farklı görev sayısı üzerinden yapılan deneylerin 5 inde ilk sıradaki algoritma GA veya DE olmuştur. Bu da GA' nın minimum tamamlanma zamanı ölçütünde, tamamlanma zamanı ölçütüne göre daha etkin sonuçlar ortaya çıkardığını göstermektedir. Ayrıca tez çalışmasında kullanılan DE algoritması literatürde bulunan DE algoritmalarının farklı kısımlarının (forward- backward transformation) kullanılmasıyla oluştuğundan özgün ve etkili olduğu görülmüştür. Tüm görev sayılarında en düşük değerli ilk üç algoritmaya bakıldığında, içinde mutlaka hibrit algoritmaların (SAGA-GASA-DESA) birinin bulunduğu görülmüştür.

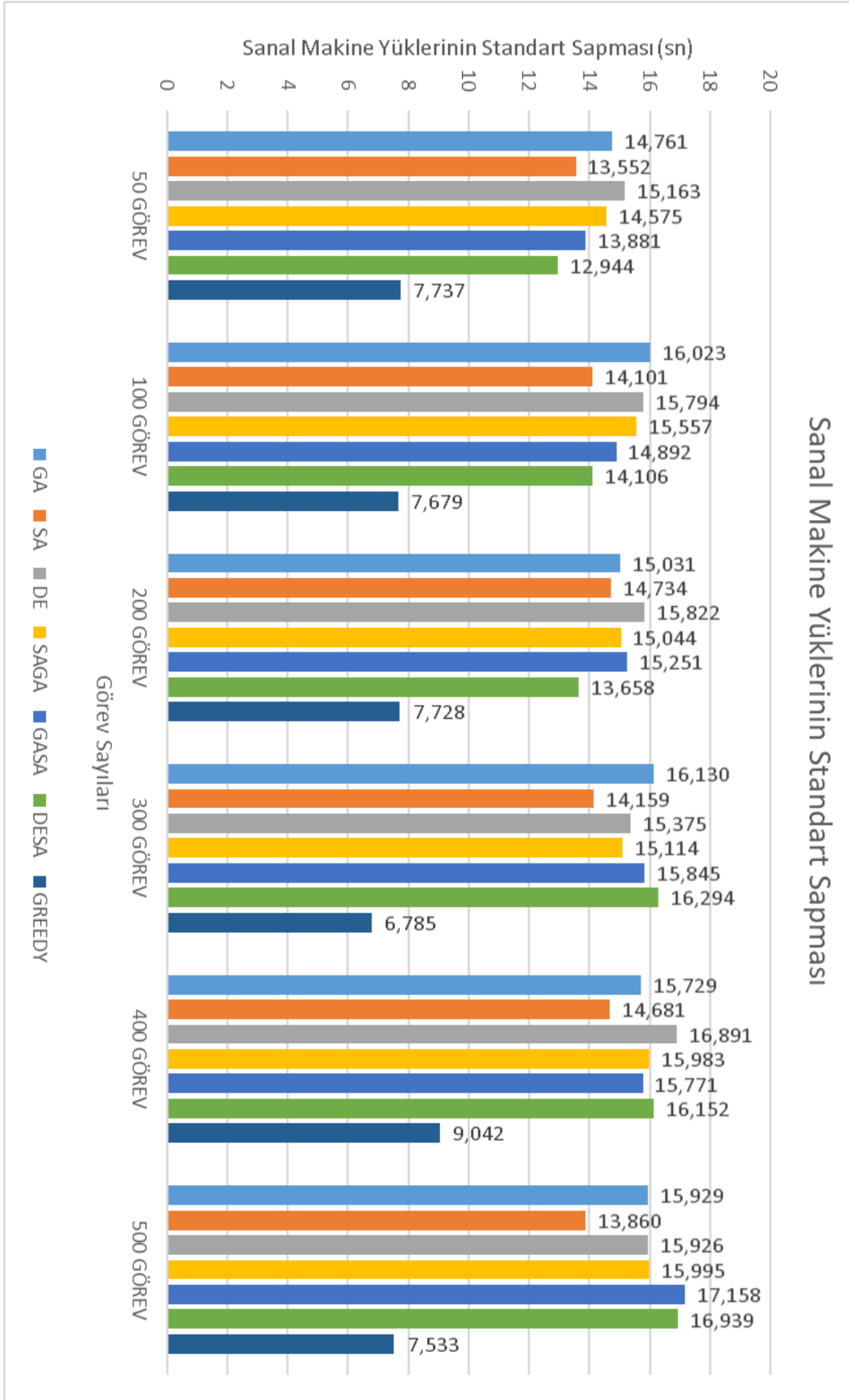
5.2.3. Sanal makine yüklerinin standart sapması

Sanal makine yüklerinin standart sapması ölçütü, sanal makinelerin yük dengelemesi hakkında bilgi verir. Çizelge 5.5' te tüm görev sayıları için sanal makine yüklerinin standart sapma değerleri verilmiştir.

Çizelge 5.5: Değişen görev sayıları üzerinden algoritmalara göre *sanal makine yüklerinin standart sapma* değerleri tablosu.

		GA	SA	DE	SAGA	GASA	DESA	GREEDY
50 GÖREV	AVG	14,761	13,552	15,163	14,575	13,881	12,944	7,737
	SD	3,617	3,764	3,541	3,967	3,646	3,301	0,000
100 GÖREV	AVG	16,023	14,101	15,794	15,557	14,892	14,106	7,679
	SD	3,622	3,938	3,686	3,935	4,045	4,384	0,000
200 GÖREV	AVG	15,031	14,734	15,822	15,044	15,251	13,658	7,728
	SD	3,477	4,588	3,514	4,433	4,858	6,514	0,000
300 GÖREV	AVG	16,130	14,159	15,375	15,114	15,845	16,294	6,785
	SD	3,664	4,157	3,907	4,316	4,994	5,550	0,000
400 GÖREV	AVG	15,729	14,681	16,891	15,983	15,771	16,152	9,042
	SD	4,578	4,538	3,532	3,916	5,418	4,743	0,000
500 GÖREV	AVG	15,929	13,860	15,926	15,995	17,158	16,939	7,533
	SD	3,629	4,075	3,364	4,285	5,018	4,007	0,000

Makine yüklerinin standart sapmasının düşük olması, yüklerin birbirine yakın olduğunun göstergesidir. Bu da dengeli bir dağılımı gösterir. 50 adet görev için en düşük sonucu sırasıyla Greedy, DESA ve SA vermektedir. 100 adet görev için en düşük sonucu sırasıyla Greedy, SA ve DESA vermektedir. 200 adet görev için en düşük sonucu sırasıyla Greedy, DESA ve SA vermektedir. 300 adet görev için en düşük sonucu sırasıyla Greedy, SA ve SAGA vermektedir. 400 adet görev için en düşük sonucu sırasıyla Greedy, SA ve GA vermektedir. 500 adet görev için en düşük sonucu sırasıyla Greedy, SA ve DE vermektedir. Şekil 5.13' te tüm görev sayıları için sanal makine yüklerinin standart sapması verilmiştir.



Şekil 5.13: Statik çizelgelemede tüm görev sayıları üzerinden algoritmalara göre sanal makine yüklerinin standart sapma değerleri grafiği.

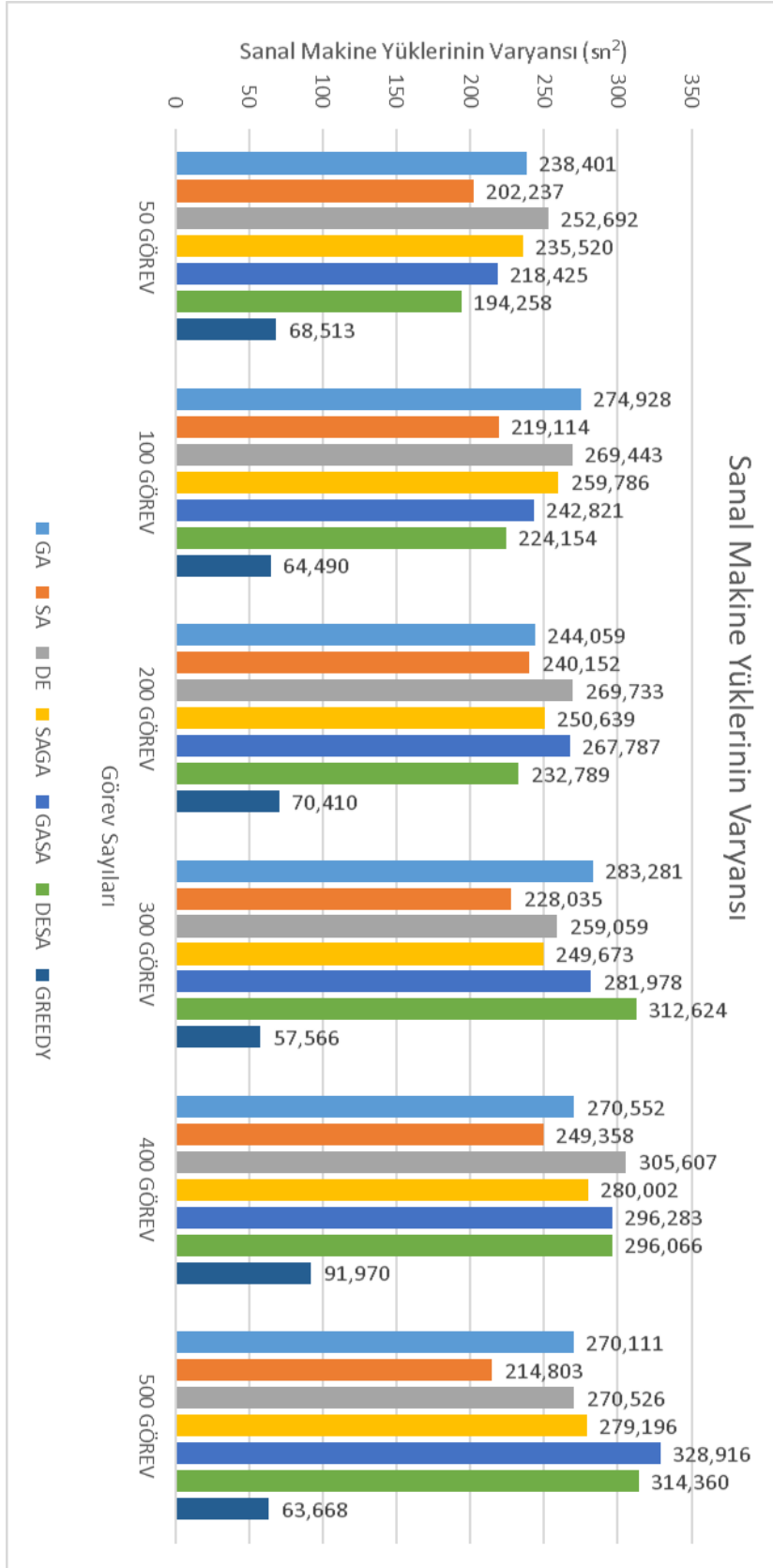
Genel olarak bakıldığında, tüm görev sayılarında Greedy algoritması yük dengelemesi için kullanılan sanal makine yüklerinin standart sapması ölçütünde açık ara en düşük sonucu vermektedir. SA ise tüm görev sayılarında Greedy den sonra ikinci veya üçüncü sırada yer almaktadır.

5.2.4. Sanal makine yüklerinin varyansı

Makine yüklerinin varyansı da bir başka yük dengeleme ölçütüdür. Makine yüklerinin varyansının düşük olması, yüklerin birbirine yakın olduğunun göstergesidir. Bu da dengeli bir dağılımı gösterir. Çizelge 5.6' daki verilerle Şekil 5.14 oluşturulmuştur.

Çizelge 5.6: Değişen görev sayıları üzerinden algoritmalara göre *sanal makine yüklerinin varyans* değerleri tablosu.

		GA	SA	DE	SAGA	GASA	DESA	GREEDY
50 GÖREV	AVG	238,401	202,237	252,692	235,520	218,425	194,258	68,513
	SD	108,488	104,067	106,175	112,882	106,073	88,407	0,000
100 GÖREV	AVG	274,928	219,114	269,443	259,786	242,821	224,154	64,490
	SD	116,924	109,220	109,706	120,174	115,606	120,252	0,000
200 GÖREV	AVG	244,059	240,152	269,733	250,639	267,787	232,789	70,410
	SD	102,188	137,956	107,743	133,271	146,865	182,181	0,000
300 GÖREV	AVG	283,281	228,035	259,059	249,673	281,978	312,624	57,566
	SD	125,816	123,554	122,189	129,933	156,632	163,771	0,000
400 GÖREV	AVG	270,552	249,358	305,607	280,002	296,283	296,066	91,970
	SD	136,020	141,789	115,602	121,804	162,727	135,978	0,000
500 GÖREV	AVG	270,111	214,803	270,526	279,196	328,916	314,360	63,668
	SD	111,797	115,466	104,191	133,484	155,323	125,851	0,000



Şekil 5.14: Statik çizelgelemede tüm görev sayıları üzerinden algoritmalarla göre sanal makine yüklerinin varyans değerleri grafiği.

Tüm görev gruplarında en düşük sonucu açıkça Greedy göstermektedir. 50, 100 ve 200 adet görevden oluşan deneylerde ikinci ve üçüncü en düşük sonucu veren algoritmalar DESA veya SA olmuştur. 300 adet görevde SA ve SAGA; 400 ve 500 adet görevde SA ve GA sırasıyla ikinci ve üçüncü en düşük sonucu veren algoritmalarlardır.

5.3. Algoritmalarla Göre Ölçütler Üzerinden Dinamik Çizelgeleme Sonuçları

50-100-200-300-400-500 görev sayılarında değerlendirme ölçütlerinin değişimi gösterilmiştir.

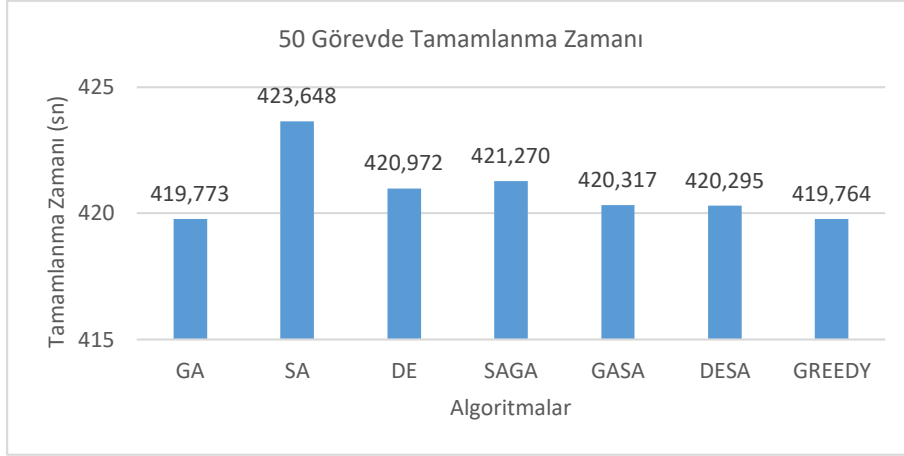
5.3.1. Tamamlanma zamanı

Çizelge 5.7’ de tüm görev sayılarına göre dinamik çizelgelemedeki tamamlanma süreleri gösterilmiştir. Buna ek olarak her görev grubu için minimum teorik limit hesaplanmıştır. Bu değer, ilgili görev grubundaki her bir görevin yürütme sürelerinin toplanarak sanal makine sayısına bölünmesiyle elde edilir.

Çizelge 5.7: Değişen görev sayıları üzerinden algoritmalarla göre *tamamlanma zamanı* değerleri tablosu.

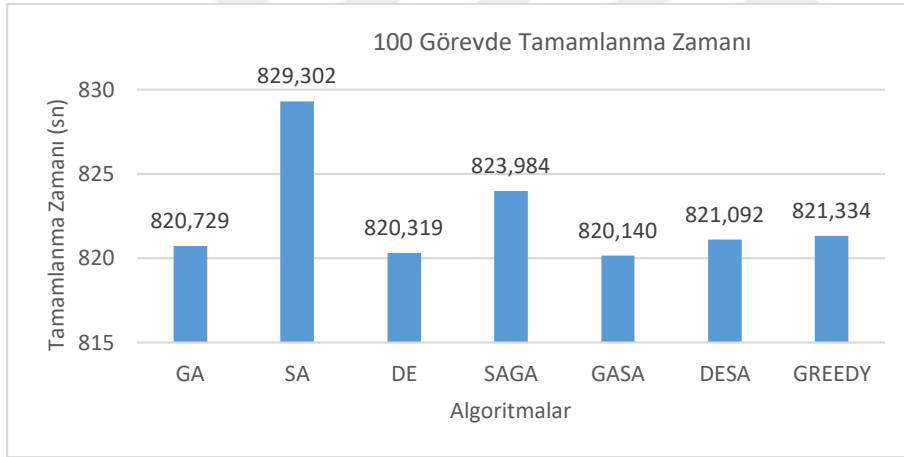
		GA	SA	DE	SAGA	GASA	DESA	GREEDY	Teorik Limit
50 GÖREV	AVG	419,773	423,648	420,972	421,270	420,317	420,295	419,764	403,416
	SD	3,344	7,476	3,828	3,137	2,516	2,555	0,000	
100 GÖREV	AVG	820,729	829,302	820,319	823,984	820,140	821,092	821,334	803,497
	SD	5,659	15,123	5,803	8,135	6,801	7,728	0,000	
200 GÖREV	AVG	1620,262	1648,717	1618,055	1626,735	1613,039	1611,980	1614,462	1601,180
	SD	7,288	33,324	7,439	11,466	6,565	6,189	0,000	
300 GÖREV	AVG	2424,375	2476,366	2421,222	2439,418	2412,604	2412,506	2417,453	2399,672
	SD	10,253	50,100	7,107	18,771	6,974	6,587	0,000	
400 GÖREV	AVG	3241,719	3341,002	3240,598	3265,887	3230,500	3228,291	3232,198	3218,351
	SD	9,135	64,053	7,844	23,411	6,321	5,121	0,000	
500 GÖREV	AVG	4032,069	4138,130	4032,022	4058,330	4020,540	4020,959	4026,587	4007,696
	SD	8,863	68,186	7,706	18,353	5,472	7,006	0,000	

Şekil 5.15’ e göre 50 adet görev sayısı ile yapılan deneyde en düşük sonuçları sırasıyla Greedy, GA ve DESA vermektedir.



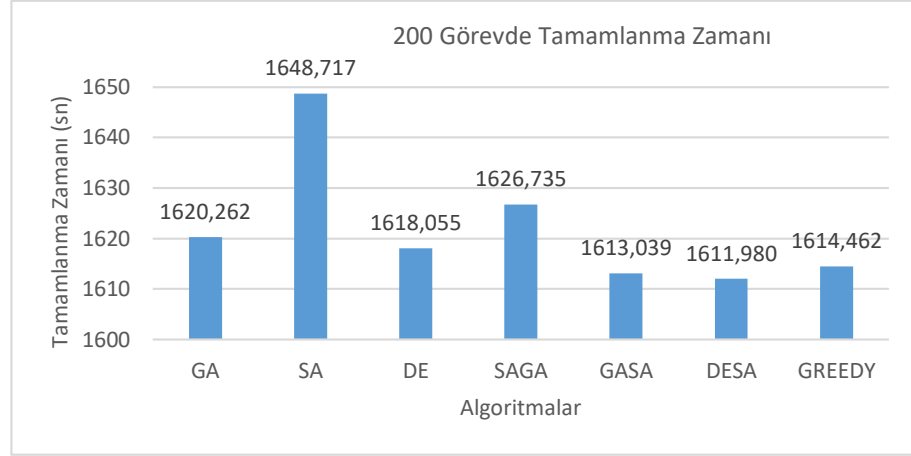
Şekil 5.15: 50 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

Şekil 5.16' ya göre 100 adet görev sayısı ile yapılan deneyde en düşük sonuçları sırasıyla GASA, DE ve GA vermektedir.

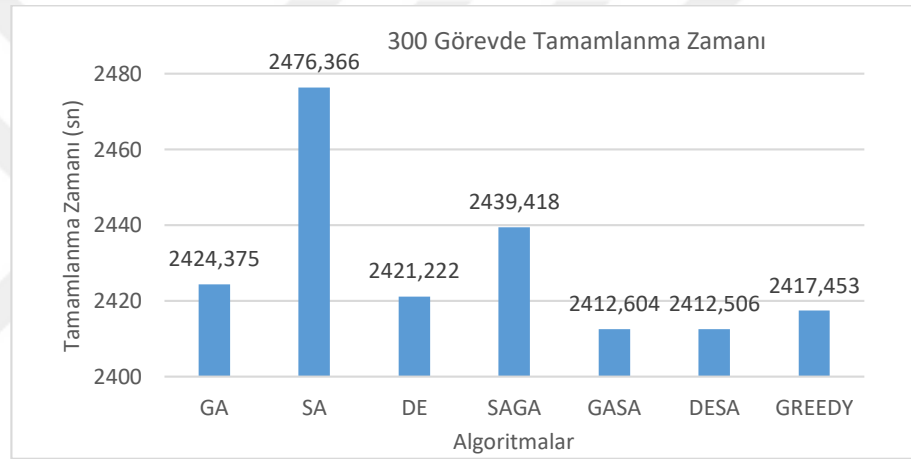


Şekil 5.16: 100 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği

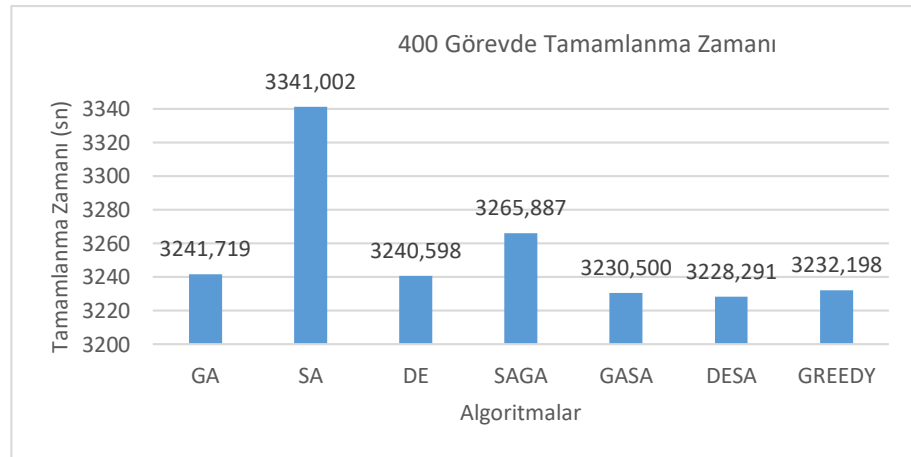
Şekil 5.17, 5.18 ve 5.19' a göre 200-300-400 adet görev sayısı ile yapılan deneyde en düşük sonuçları sırasıyla DESA, GASA ve Greedy vermektedir.



Şekil 5.17: 200 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği

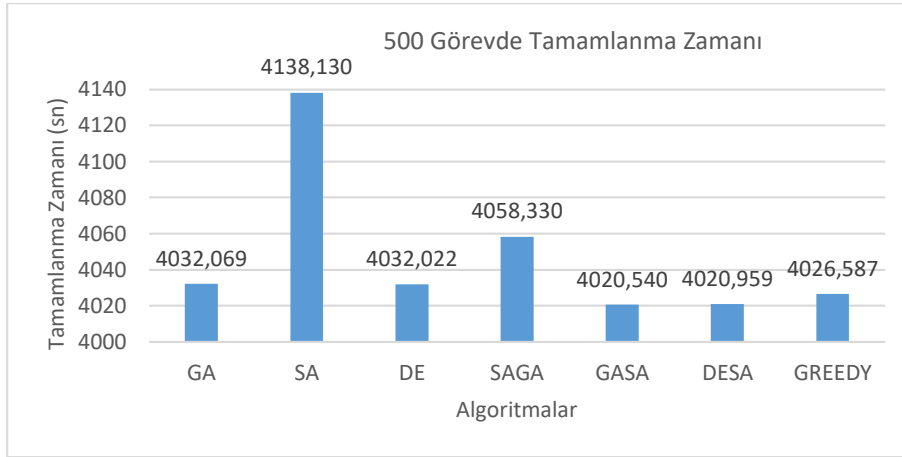


Şekil 5.18: 300 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği



Şekil 5.19: 400 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği

Şekil 5.20’ de belirtilen grafiğe göre 500 adet görevden oluşan deneyde en düşük sonuçları sırasıyla GASA, DESA ve Greedy vermektedir.



Şekil 5.20: 500 adet görev üzerinden algoritmaların *tamamlanma zamanı* değerleri grafiği.

5.3.2. Minimum tamamlanma zamanı

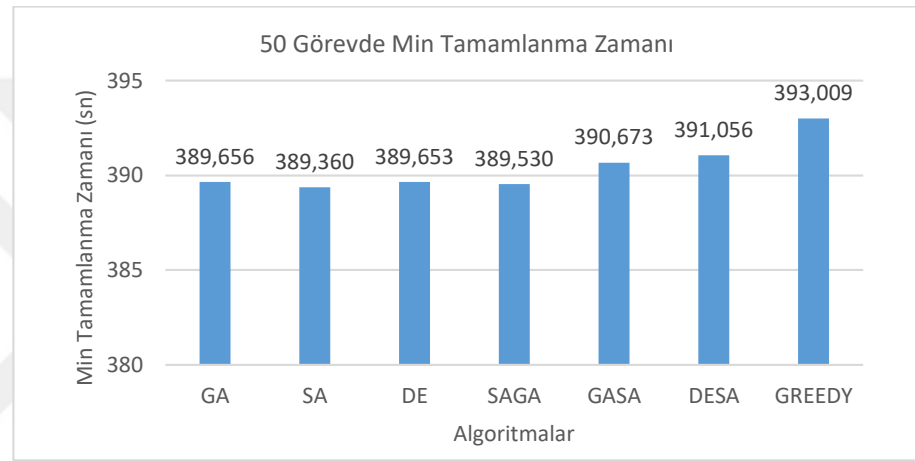
Çizelge 5.8’ de değişen görev sayılarında algoritmaların minimum tamamlanma zamanları gösterilmiştir. Takip eden şekillerde bu tablo, grafiklere dönüştürülmüştür. Çizelgede minimum teorik limit belirtilmiştir.

Çizelge 5.8: Değişen görev sayıları üzerinden algoritmalarla göre *minimum tamamlanma zamanı* değerleri tablosu.

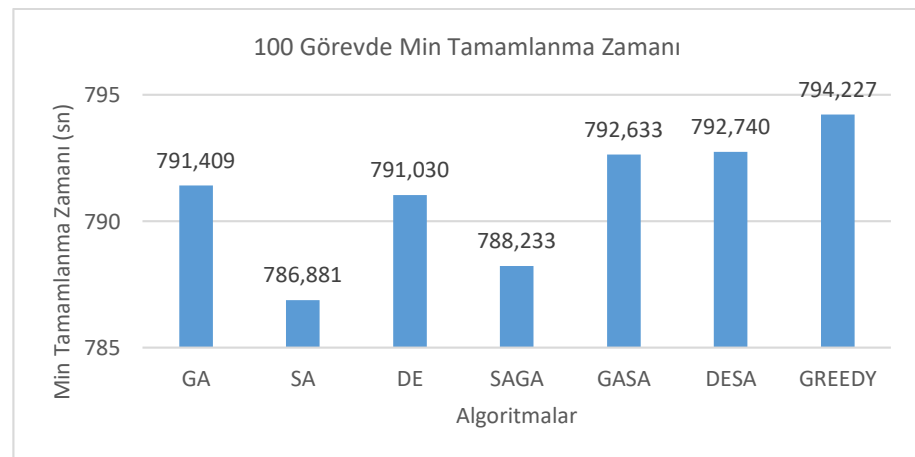
		GA	SA	DE	SAGA	GASA	DESA	GREEDY	Teorik Limit
50 GÖREV	AVG	389,656	389,360	389,653	389,530	390,673	391,056	393,009	403,416
	SD	3,719	5,993	4,472	4,747	2,910	2,466	0,000	
100 GÖREV	AVG	791,409	786,881	791,030	788,233	792,633	792,740	794,227	803,497
	SD	7,472	11,206	6,972	7,889	7,788	7,032	0,000	
200 GÖREV	AVG	1579,195	1557,978	1581,462	1566,300	1589,221	1589,034	1592,668	1601,180
	SD	11,892	24,409	11,006	20,537	7,243	7,638	0,000	
300 GÖREV	AVG	2374,289	2330,685	2373,584	2349,439	2385,075	2387,803	2386,960	2399,672
	SD	16,269	36,194	15,468	33,581	13,301	9,617	0,000	
400 GÖREV	AVG	3188,394	3095,669	3186,354	3153,754	3201,210	3206,905	3207,690	3218,351
	SD	17,212	50,982	17,851	39,508	13,373	7,263	0,000	
500 GÖREV	AVG	3981,418	3878,838	3980,671	3949,618	3994,566	3997,549	3998,609	4007,696
	SD	16,596	51,521	18,079	34,795	11,096	8,913	0,000	

Şekil 5.21’ de 50 adet görevden oluşan deney, Şekil 5.22’ de 100 adet görevden oluşan deney gösterilmiştir. 50, 100, 300, 400 ve 500 görevden oluşan deneylerde minimum tamamlanma zamanı ölçütüne göre en düşük değerleri veren üç algoritma sırasıyla: SA- SAGA- DE dir. 200 görevden oluşan deneyde minimum tamamlanma zamanı ölçütüne göre en düşük değerleri veren üç algoritma sırasıyla : SA- SAGA- GA’ dır.

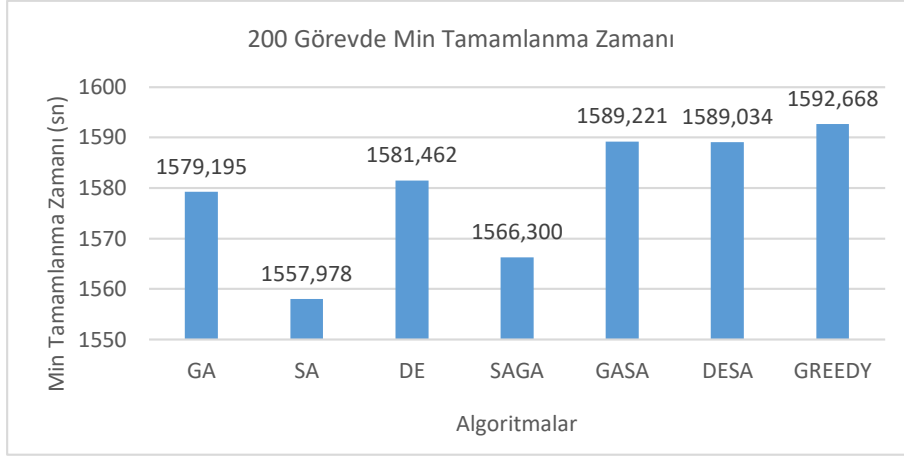
Genel olarak değerlendirildiğinde tüm görev sayılarında yapılan deneylerde en düşük değerli ilk iki sonuç sırasıyla SA ve SAGA olmuştur.



Şekil 5.21: 50 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

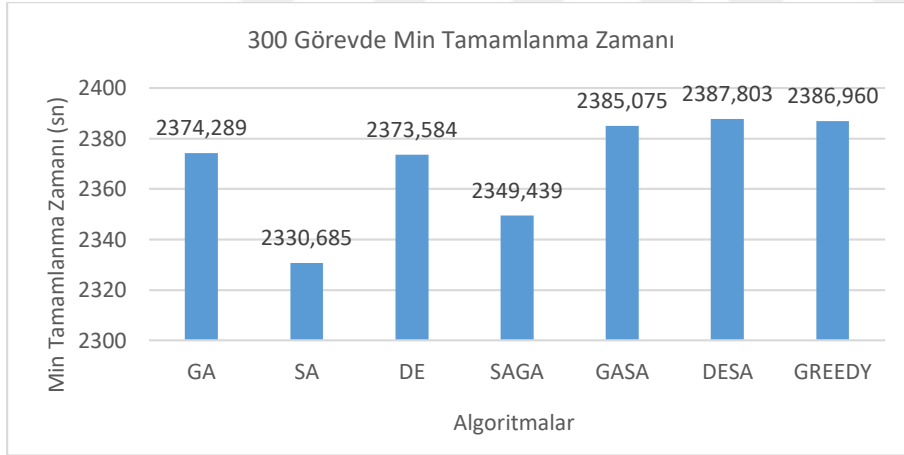


Şekil 5.22: 100 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

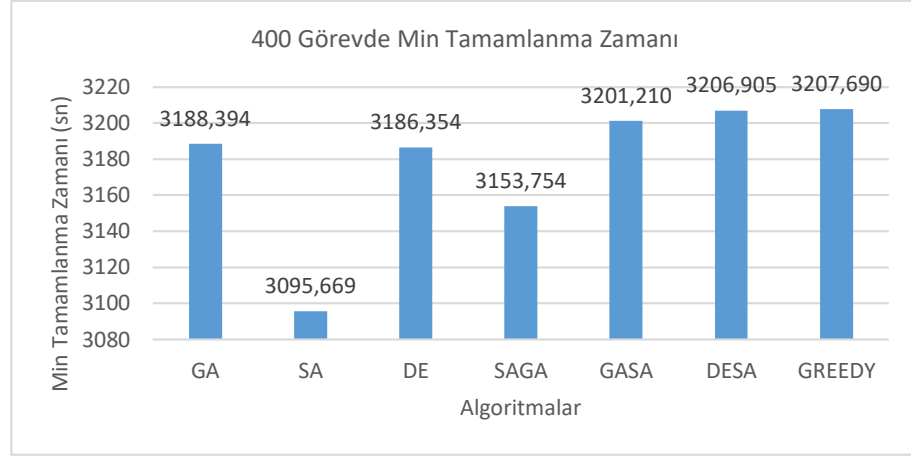


Şekil 5.23: 200 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.

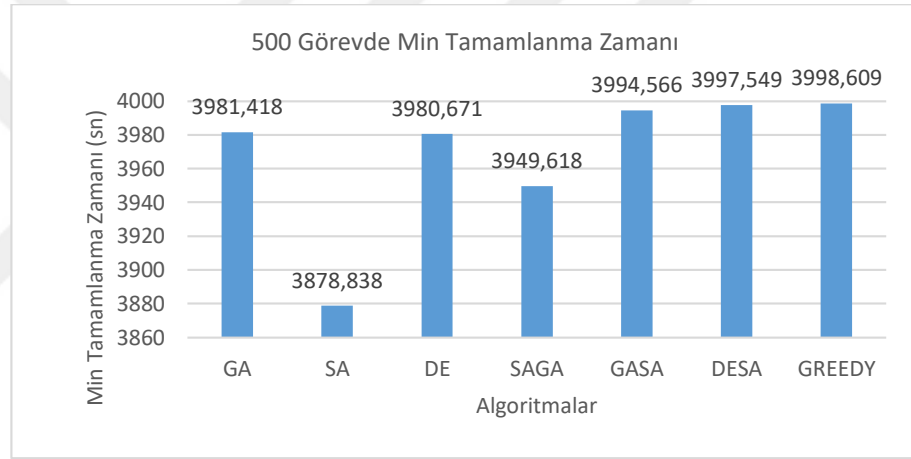
Şekil 5.23’ te 200 adet görevden oluşan deney gösterilmiştir. Şekil 5.24’ te 300 adet görevden oluşan deney, Şekil 5.25’ de 400 adet görevden oluşan deney, Şekil 5.26’ da 500 adet görevden oluşan deney gösterilmiştir.



Şekil 5.24: 300 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği.



Şekil 5.25: 400 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği



Şekil 5.26: 500 adet görev üzerinden algoritmaların *minimum tamamlanma zamanı* değerleri grafiği

5.3.3. Sanal makine yüklerinin standart sapması

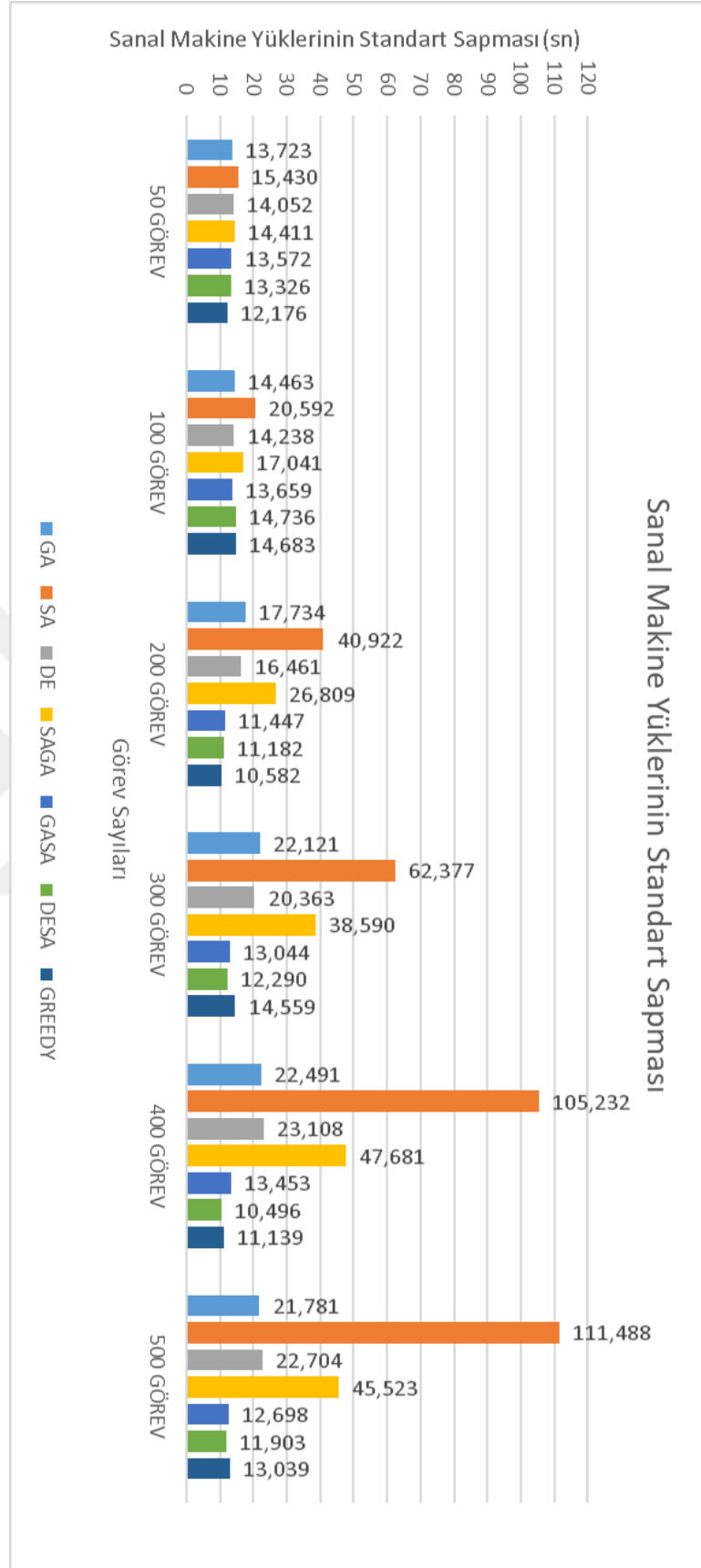
Çizelge 5.9' da sanal makine yüklerinin standart sapması değerleri gösterilmiştir.

50 ve 200 görevden oluşan deneyde en düşük sonucu Greedy algoritması vermiştir. 100 görevden oluşan deneyde en düşük sonucu GASA vermiştir. 300, 400 ve 500 görevden oluşan deneylerde en düşük sonucu DESA vermiştir.

Çizelge 5.9: Değişen görev sayıları üzerinden algoritmalara göre *sanal makine yüklerinin standart sapma* değerleri tablosu.

		GA	SA	DE	SAGA	GASA	DESA	GREEDY
50 GÖREV	AVG	13,723	15,430	14,052	14,411	13,572	13,326	12,176
	SD	2,773	4,960	3,024	2,894	2,008	1,700	0,000
100 GÖREV	AVG	14,463	20,592	14,238	17,041	13,659	14,736	14,683
	SD	4,905	10,449	4,833	5,849	6,058	6,026	0,000
200 GÖREV	AVG	17,734	40,922	16,461	26,809	11,447	11,182	10,582
	SD	7,265	24,239	7,225	13,184	5,377	5,390	0,000
300 GÖREV	AVG	22,121	62,377	20,363	38,590	13,044	12,290	14,559
	SD	10,425	31,440	8,378	20,024	7,077	6,291	0,000
400 GÖREV	AVG	22,491	105,232	23,108	47,681	13,453	10,496	11,139
	SD	9,843	40,956	9,728	25,061	8,071	5,193	0,000
500 GÖREV	AVG	21,781	111,488	22,704	45,523	12,698	11,903	13,039
	SD	10,185	44,687	10,076	19,885	6,390	5,985	0,000

50 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: Greedy, DESA ve GASA' dır. 100 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: GASA, DE ve GA' dır. 200 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: Greedy, DESA ve GASA' dır. 300 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: DESA, GASA ve Greedy'dir. 400 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: DESA, Greedy ve GASA' dır. 500 görevden oluşan deneyin en düşük sonuçları veren ilk üç algoritması: DESA, GASA ve Greedy' dir. Çizelge 5.9' daki verilere göre Şekil 5.27 oluşturulmuştur.



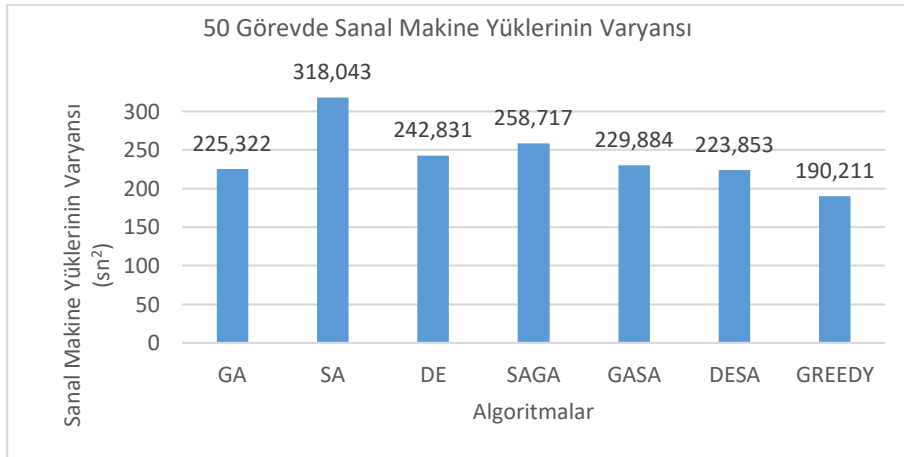
Şekil 5.27: Dinamik çizelgelemede tüm görev sayıları üzerinden algoritmalarla göre sanal makine yüklerinin standart sapma değerleri grafiği.

5.3.4. Sanal makine yüklerinin varyansı

Çizelge 5.10’ da tüm görev sayıları için algoritmalara göre sanal makine yüklerinin varyansı gösterilmiştir. Takip eden grafiklerde bu tablodaki veriler kullanılmıştır. Bu ölçütün düşük olması, sanal makine yüklerin birbirine yakın olduğunun göstergesidir. Şekil 5.28’ de gösterilen 50 adet görevle yapılan deneydeki en düşük sonuçları veren üç algoritma Greedy, DESA ve GA’ dır.

Çizelge 5.10: Değişen görev sayıları üzerinden algoritmalara göre *sanal makine yüklerinin varyans* değerleri tablosu.

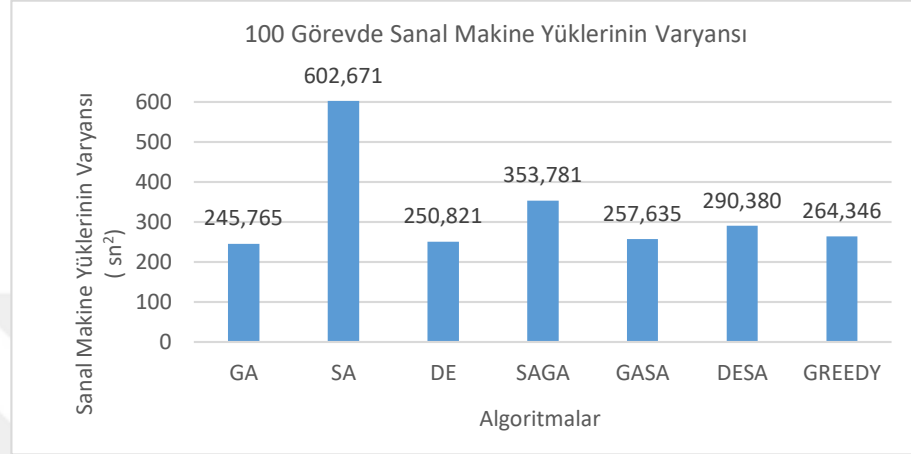
		GA	SA	DE	SAGA	GASA	DESA	GREEDY
50 GÖREV	AVG	225,322	318,043	242,831	258,717	229,884	223,853	190,211
	SD	68,192	180,081	86,694	92,838	45,486	36,124	0,000
100 GÖREV	AVG	245,765	602,671	250,821	353,781	257,635	290,380	264,346
	SD	168,395	576,828	149,061	209,988	184,087	191,439	0,000
200 GÖREV	AVG	384,108	2675,568	338,198	1002,395	177,799	170,935	147,316
	SD	303,632	2761,925	274,644	897,367	131,874	138,163	0,000
300 GÖREV	AVG	627,295	5587,912	553,240	2014,054	252,103	226,311	248,995
	SD	568,494	4736,011	424,590	1902,437	229,349	192,254	0,000
400 GÖREV	AVG	630,385	13320,137	642,340	3007,547	273,115	147,321	149,130
	SD	501,889	8990,471	498,962	2714,774	294,669	122,270	0,000
500 GÖREV	AVG	609,793	15239,351	681,411	2632,021	219,355	189,375	205,326
	SD	560,915	11103,664	589,578	2231,532	200,652	168,561	0,000



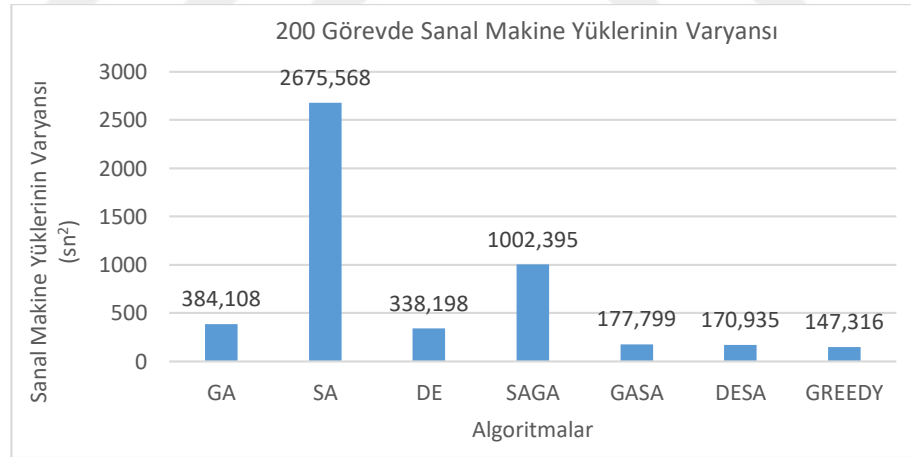
Şekil 5.28: 50 adet görev üzerinden algoritmalara göre *sanal makine yüklerinin varyans* değerleri grafiği.

Şekil 5.29' da gösterilen 100 adet görevle yapılan deneydeki en düşük sonuçları veren üç algoritma GA, DE ve GASA' dır.

Şekil 5.30' da gösterilen 200 adet görevle yapılan deneydeki en düşük sonuçları veren üç algoritma Greedy, DESA ve GASA' dır.

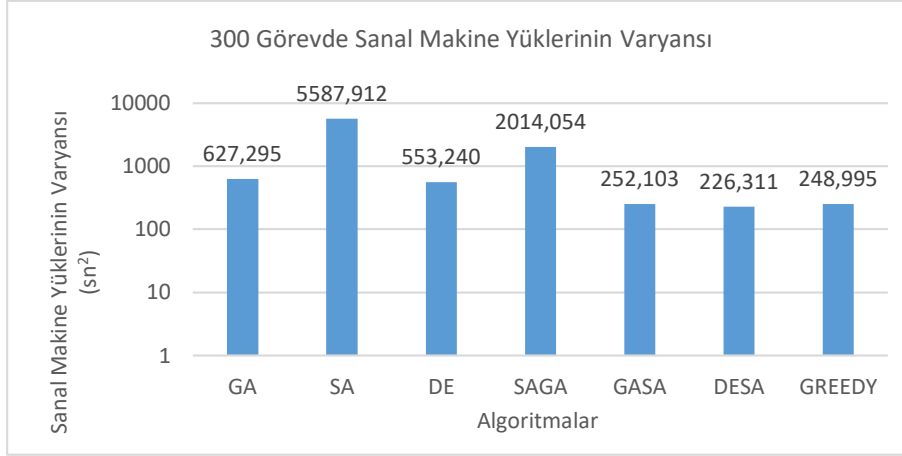


Şekil 5.29: 100 adet görev üzerinden algoritmalarla göre *sanal makine yüklerinin varyans* değerleri grafiği.

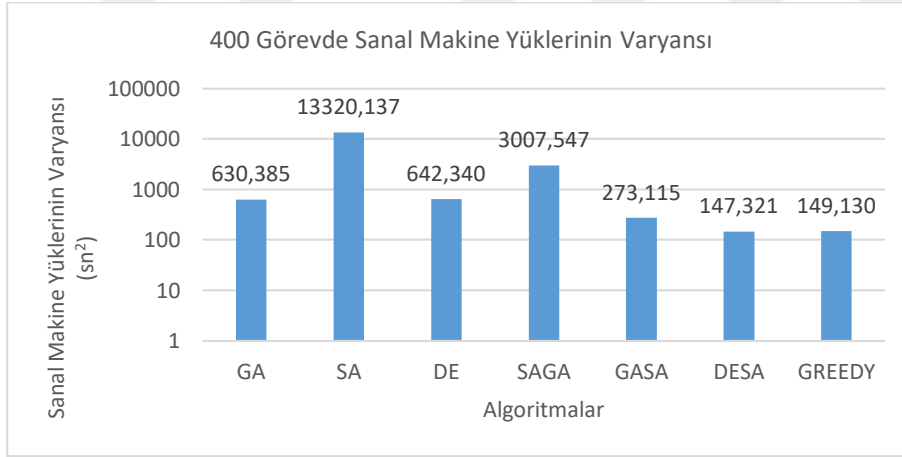


Şekil 5.30: 200 adet görev üzerinden algoritmalarla göre *sanal makine yüklerinin varyans* değerleri grafiği.

Şekil 5.31' de 300 adet görevle yapılan deney, Şekil 5.32' de 400 adet görevle yapılan deney gösterilmiştir. 300-400-500 adet görevle yapılan deneylerdeki en düşük sonuçları veren üç algoritma DESA, Greedy ve GASA' dır.

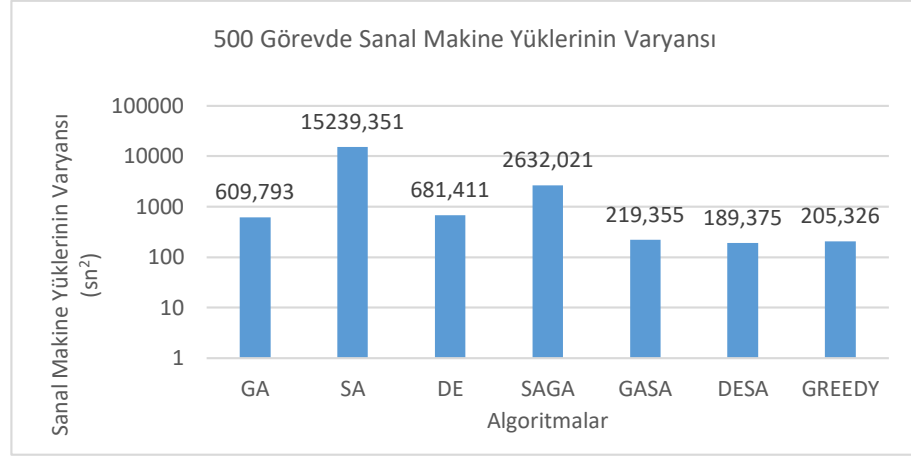


Şekil 5.31: 300 adet görev üzerinden algoritmalarla göre *sanal makine yüklerinin varyans* değerleri grafiği.



Şekil 5.32: 400 adet görev üzerinden algoritmalarla göre *sanal makine yüklerinin varyans* değerleri grafiği.

Şekil 5.33' de 500 görev için en düşük varyans değerlerini sırasıyla DESA, Greedy ve GASA vermiştir.



Şekil 5.33: 500 adet görev üzerinden algoritmalarla göre *sanal makine yüklerinin varyans* değerleri grafiği.

6. SONUÇLAR

Bu tez çalışmasında meta sezgisel yöntemlerle bulut ortamında çalışan görevler için çizelgeleme algoritmaları geliştirilmiştir. Literatürde var olan GA, SA, Greedy gibi algoritmalar kullanılmıştır, bunlara ek olarak farklı yöntemlerin birlikte kullanıldığı DE ve hibrit olarak SAGA, GASA ve DESA algoritmaları geliştirilmiştir. Ayrıca, dinamik çizelgeleme yöntemi önerilmiştir. Bu yedi algoritma, değişen görev sayılarında literatürde mevcut statik çizelgeleme ve tezde önerilen dinamik çizelgeleme yöntemiyle uygulanmış ve tamamlanma zamanının azaltılması ve sanal makineler arasındaki yük dengelenmesinin sağlanması amaçlanmıştır.

Sonuçlar tamamlanma zamanı, minimum tamamlanma zamanı, sanal makine yüklerinin standart sapması ve sanal makine yüklerinin varyansı ölçütleri üzerinden değerlendirilmiştir.

Statik çizelgelemede; en düşük tamamlanma zamanını Greedy' nin verdiği, minimum tamamlanma zamanında DE, GA, DESA'nın en düşük sonuçları gösterdiği, sanal makine yüklerinin standart sapması ve varyansında Greedy'nin en düşük sonucu gösterdiği gözlemlenmiştir.

Dinamik çizelgelemede, statik çizelgelemeye göre daha değişken sonuçlarla karşılaşmıştır. En düşük tamamlanma zamanını Greedy, GASA ve DESA'nın verdiği, en düşük minimum tamamlanma zamanını SA'nın verdiği, en düşük sanal makine yüklerinin standart sapmasını Greedy, GASA ve DESA'nın verdiği, en düşük sanal makine yüklerinin varyansını Greedy, GA ve DESA'nın verdiği gözlemlenmiştir.

Statik çizelgelemede tamamlanma zamanı ve sanal makine yüklerinin standart sapması ölçütlerine bakıldığında Greedy'den sonra ikinci olarak DESA ve SA'nın en düşük sonuçları verdiği görülmektedir.

Dinamik çizelgelemede tamamlanma zamanı ve sanal makine yüklerinin standart sapması ölçütlerine bakıldığında görev sayısı arttıkça DESA ve GASA' nın en düşük sonucu verdiği görülmüştür.

Dikkat çeken bir nokta olarak SA' nın dinamik çizelgelemede minimum tamamlanma zamanında tüm görev grupları için en düşük sonucu vermesine karşın tamamlanma zamanında, yük dengelemedeki iki ölçüt olan standart sapma ve varyansta tüm görev gruplarında en yüksek sonucu vermiştir

Bu tez çalışmasının katkıları şöyledir: Literatürde mevcut statik çizelgeleme yöntemi üzerine GA ve SA adapte edilmiştir, Greedy uygulanmıştır. Yeni geliştirilen DE, SAGA, GASA ve DESA uygulanmıştır. Yeni bir dinamik çizelgeleme yöntemi önerilmiştir. Bu dinamik çizelgeleme yöntemi üzerine, statik çizelgelemede olduğu gibi GA ve SA adapte edilmiştir, Greedy uygulanmıştır. Yeni geliştirilen DE, SAGA, GASA ve DESA uygulanmıştır.

KAYNAKLAR DİZİNİ

- Ahn, C. W. and Ramakrishna, R. S.,** 2003, Elitism-based compact genetic algorithms, *IEEE Transactions on Evolutionary Computation*, 7(4), 367-385p.
- Alla, H. B., Alla, S. B., Touhafi, A. and Ezzati, A.,** 2018, A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment, *Cluster Computing*, 21(4), 1797-1820p.
- Amalarethinam, D. G. and Beena, T. L. A.,** 2014, Cloud scheduling-a survey, *International Journal of Computer Applications*, 97(13).
- Babu, D. and Krishna, P.V.,** 2013, Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing*, 2292-2303p.
- Bittencourt, L. F., Goldman, A., Madeira, E. R., da Fonseca, N. L. and Sakellariou, R.,** 2018, Scheduling in distributed systems: A cloud computing perspective, *Computer Science Review*, 30, 31-54p.
- Calheiros, R. N., Ranjan, R., De Rose, C. A. and Buyya, R.,** 2009, Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services, *GRIDS-TR-2009-1 Technical Report*.
- Cheng, W., Shi, H., Yin, X. and Li, D.,** 2011, An elitism strategy based genetic algorithm for streaming pattern discovery in wireless sensor networks, *IEEE Communications Letters*, 15(4), 419-421p.
- CloudSim,** <http://www.cloudbus.org/cloudsim/> (Erişim tarihi: 08.05.2019).
- Ebadifard, F. and Babamir, S. M.,** 2018, A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment, *Concurrency and Computation: Practice and Experience*, 30(12), e4368.
- Engelbrecht, A. P. and Pampara, G.,** 2007, Binary differential evolution strategies, *2007 IEEE Congress on Evolutionary Computation*, 1942-1947p.

KAYNAKLAR DİZİNİ (devam)

- Fan, Z., Shen, H., Wu, Y. and Li, Y.,** 2013, Simulated-annealing load balancing for resource allocation in cloud environments. In *2013 International Conference on Parallel and Distributed Computing, Applications and Technologies*, 1-6p.
- Gabi, D., Ismail, A. S., Zainal, A., Zakaria, Z. and Al-Khasawneh, A.,** 2018, Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing enviroment. *Journal of ICT*, 17(3), 435-467p.
- Gan, G. N., Huang, T. L. and Gao, S.,** 2010, Genetic simulated annealing algorithm for task scheduling based on cloud computing environment, *International Conference on Intelligent Computing and Integrated Systems IEEE*, 60-63p.
- Gawali, M. B. and Shinde, S. K.,** 2018, Task scheduling and resource allocation in cloud computing using a heuristic approach, *Journal of Cloud Computing*, 7(1).
- Ge, J., Cai, Y. and Fang, Y.,** 2018, Cloud computing task scheduling strategy based on differential evolution and ant colony optimization, *AIP Conference Proceedings*.
- Ge, J., He, Q. and Fang, Y.,** 2017, Cloud computing task scheduling strategy based on improved differential evolution algorithm. *AIP Conference Proceedings*.
- Goldberg, D.,** 1989, Genetic algorithms in search, optimization, and machine learning, *Addison- Wesley*.
- Hwang, K., Dongarra, J. and Fox, G. C.,** 2013, Distributed and cloud computing: from parallel processing to the internet of things, *Morgan Kaufmann*.
- Ibrahim, E., El-Bahnasawy, N. A. and Omara, F. A.,** 2017, Dynamic task scheduling in cloud computing based on the availability level of resources, *International Journal of Grid and Distributed Computing*, 10(8), 21-35p.
- Kalra, M. and Singh, S.,** 2015, A review of metaheuristic scheduling techniques in cloud computing, *Egyptian Informatics Journal*, 16(3), 275-295p.

KAYNAKLAR DİZİNİ (devam)

- Kamalinia, A. and Ghaffari, A.,** 2017, Hybrid task scheduling method for cloud computing by genetic and DE algorithms, *Wireless Personal Communications*, 97(4), 6301-6323p.
- Karaboğa, D. and Ökdem, S.,** 2004, A simple and global optimization algorithm for engineering problems: differential evolution algorithm, *Turkish Journal of Electrical Engineering & Computer Sciences*, 12(1), 53-60p.
- Keskintürk, T.,** 2006, Diferansiyel gelişim algoritması, *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, 85-89s.
- Li, K., Xu, G., Zhao, G., Dong, Y. and Wang, D.,** 2011, Cloud task scheduling based on load balancing ant colony optimization, *2011 Sixth Annual China Grid Conference IEEE*, 3-9p.
- Luo, X. and Fu, X.,** 2018, Configuration optimization method of Hadoop system performance based on genetic simulated annealing algorithm, *Cluster Computing*, 1-9p.
- Mell, P. and Grance, T.,** 2011, The NIST definition of cloud computing, *NIST Special Publication* 800-145.
- Microsoft,** “What is Saas?”, <https://azure.microsoft.com/tr-tr/overview/what-is-saas/>, (Erişim tarihi: 13.05.2019).
- Miller, B. L. and Goldberg, D. E.,** 1995, Genetic algorithms, tournament selection and the effects of noise, *Complex systems*, 9(3), 193-212p.
- Nzanywayingoma, F. and Yang, Y.,** 2017, Effective task scheduling and dynamic resource optimization based on heuristic algorithms in cloud computing environment, *TIIS*, 11(12), 5780-5802p.
- Onwubolu, G. and Davendra, D.,** 2006, Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research*, 171(2), 674-692p.

KAYNAKLAR DİZİNİ (devam)

- Pico, C. G. and Wainwright, R. L.,** 1994, Dynamic scheduling of computer tasks using genetic algorithms, *Proceedings of the First IEEE Conference on Evolutionary Computation IEEE World Congress on Computational Intelligence*, 829-833p.
- Ramadan, H. H. and Kashyap, D.,** 2017, Quality of service (QoS) in cloud computing, *International Journal of Computer Science and Information Technologies*, 8(3), 318-320p.
- Russell, S. and Norvig, P.,** 2010, Artificial intelligence: a modern approach, Global Edition. Harlow, Essex, England: Pearson Education.
- Saleh, H., Nashaat, H., Saber, W. and Harb, H. M.,** 2019, IPSO task scheduling algorithm for large scale data in cloud computing environment. *IEEE Access*, 7, 5412-5420p.
- Sastry, K., Goldberg, D. and Kendall, G.,** 2005, Genetic algorithms, In Search Methodologies, *Springer*, Boston, MA, 97-125p.
- Singh, P., Dutta, M. and Aggarwal, N.,** 2017, A review of task scheduling based on meta-heuristics approach in cloud computing, *Knowledge And Information Systems*, 52, 1-51p.
- Singh, R., Paul, S. and Kumar, A.,** 2014, Task scheduling in cloud computing: review, *International Journal of Computer Science and Information Technologies*, 7940-7944p.
- Sudheer, M. S., Reddy, K. G., Sree, P. K. and Raju, V. P.,** 2017, An effective analysis on various scheduling algorithms in cloud computing, *2017 International Conference on Inventive Computing and Informatics (ICICI) IEEE*, 931-936p.
- Tsai, C. W., Huang, W. C., Chiang, M. H., Chiang, M. C. and Yang, C. S.,** 2014, A hyper-heuristic scheduling algorithm for cloud, *IEEE Transactions on Cloud Computing*, 2(2), 236-250p.

KAYNAKLAR DİZİNİ (devam)

- Wadhonkar, A. and Theng, D.,** 2016, A survey on different scheduling algorithms in cloud computing, *2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Chennai, India.
- Wang, T., Liu, Z., Chen, Y., Xu, Y. and Dai, X.,** 2014, Load balancing task scheduling based on genetic algorithm in cloud computing, *IEEE 12th International Conference on Dependable, Autonomic and Secure Computing*, 146-152p.
- Zomaya, A.Y. and Teh, Y.H.,** 2001, Observations on using genetic algorithms for dynamic load-balancing, *IEEE Transactions On Parallel And Distributed Systems*, 12(9), 899-911p.

TEŞEKKÜR

Tez çalışması boyunca kıymetli bilgi, birikim ve tecrübeleri ile bana yol gösteren, hiçbir zaman desteğini esirgemeyerek tezin tamamlanmasını sağlayan danışmanım Doç. Dr. Hasan Bulut'a ve her zaman bana destek olan aileme en içten teşekkürlerimi sunarım.

30 / 07 / 2019

İmzası

Merve Nur AKTAN

ÖZGEÇMİŞ

MERVE NUR AKTAN

İLETİŞİM BİLGİLERİ

- Adnan Menderes Üniversitesi, Bilgisayar Mühendisliği Bölümü, Efeler/AYDIN.
- E posta: merve.nur.aktan@adu.edu.tr

EĞİTİM BİLGİLERİ

- Lisans, Dokuz Eylül Üniversitesi, Bilgisayar Mühendisliği Bölümü, 2011-2016.
- Yüksek Lisans, Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 2016-devam ediyor.

AKADEMİK DENEYİM

- Araştırma Görevlisi, Adnan Menderes Üniversitesi, Bilgisayar Mühendisliği Bölümü, 2019.

EKLER

Ek A Türkçe-İngilizce Terimler Sözlüğü

Ek B İngilizce-Türkçe Terimler Sözlüğü



Ek A Türkçe-İngilizce Terimler Sözlüğü

Doğrudan Döngüsüz Graf:	Direct acyclic graph
Kabul Edilebilir Kuyruklar:	Acceptable queues
Kayan Pencere:	Sliding window
Ortalama kaynak kullanımı:	Average resource utilization
Tabu Araması:	Tabu search
Tamamlanma Zamanı:	Makespan
Tepe Tırmanışı:	Hill climbing
Turnuva Seçilimi:	Tournament selection
Yük Dengeleme:	Load balancing
Yüksek Seviyeli Dil:	High level language
Yürütme Süresi:	Execution Time

Ek B İngilizce-Türkçe Terimler Sözlüğü

- Backward Transformation:** Geri dönüşüm
- Cloudlet:** Cloudsim ortamındaki görevler
- Cycle Crossover:** Döngülü Çaprazlama
- Datacenter:** Veri merkezi
- Deterministic:** Rastgelelik içermeyen algoritmalar
- Elitism:** GA da elitizm yöntemi
- Event:** Cloudsim ortamındaki olaylar
- Execution List:** Cloudsim ortamında, işlenen görevlerin bulunduğu liste
- Exhaustive Search:** Tam arama
- Fitness:** Uygunluk değeri
- Forward Transformation:** İleri dönüşüm
- Greedy:** Açgözlü yaklaşım
- Heuristics:** Sezgisel algoritmalar
- Host:** Ana bilgisayar
- Makespan:** Sistemdeki en son görevin tamamlanma zamanı
- Min Completion Time:** Minimum tamamlanma zamanı
- Mutant:** Mutasyona uğramış
- Normalized:** Sayının belli değer aralığına çekilmesi
- Order Based Crossover:** Sıralamaya dayalı çaprazlama
- Population:** Popülasyon, nesil
- Random:** Rastgele
- Return Event:** Cloudsim ortamında, görevin işlenmesinin tamamlandığını belirten olay
- Roulette Wheel Selection:** Rulet tekerleği seçilimi
- Scheduling:** Çizelgeleme
- Simulator:** Benzetim ortamı
- Sliding Window:** Kayan pencere
- Submit Event:** Cloudsim ortamında görevin, makineye atıldığı olay
- Swap Mutation:** Kromozomdaki iki genin karşılıklı yer değiştirmesi
- Task:** Görev, iş
- Trial Chromosome:** Deneme kromozomu
- Uniform Distribution:** Tekbiçimli dağılım

Uniform Order Based Crossover: Tekbiçimli sıralamaya dayalı çaprazlama

User: Kullanıcı

Virtual Machine: Sanal Makine

Waiting List: Cloudsim ortamında görevlerin bekleme listesi

