
GAZİ ÜNİVERSİTESİ
Gazi
1926

BULUT BİLİŞİMDE ÇOK AMAÇLI OPTİMİZASYON TABANLI DİNAMİK YÜK DENGELEME

Serap DÖRTERLER

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAYIS 2018

Serap DÖRTERLER tarafından hazırlanan BULUT BİLİŞİMDE ÇOK AMAÇLI OPTİMİZASYON TABANLI DİNAMİK YÜK DENGELEME adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Prof. Dr. Suat ÖZDEMİR

Bilgisayar Mühendisliği Anabilim Dalı, Gazi Üniversitesi ...……

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum.

Başkan: Prof. Dr. Şeref SAĞIROĞLU

Bilgisayar Mühendisliği Anabilim Dalı, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……..

Üye: Dr. Öğr. Üyesi Murat AYDOS

Bilgisayar Mühendisliği Anabilim Dalı, Hacettepe Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……..

Tez Savunma Tarihi: 30/05/2018

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..…….

Prof. Dr. Sena YAŞYERLİ

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Serap DÖRTERLER

30/05/2018

BULUT BİLİŞİMDE ÇOK AMAÇLI OPTİMİZASYON TABANLI DİNAMİK YÜK DENGELEME

(Yüksek Lisans Tezi)

Serap DÖRTERLER

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Mayıs 2018

ÖZET

Bulut bilişim, insanların bulunduğu yerden uzakta bulunan donanım, işletim sistemi, yazılım gibi kaynakları istedikleri zaman internet üzerinden kullanmalalarını sağlamaktadır. Bulut bilişimin vazgeçilmez bir parçası olan sanallaştırma teknolojisi işlemci, bellek, bant genişliği gibi donanımların birden fazla sanal makine arasında paylaştırılarak kullanılmasına imkân sunmaktadır. Sanallaştırma teknolojisi sanal makinelerin fiziksel makineler üzerinde eşzamanlı yürütülmesi esasına dayanmaktadır. M adet sanal makinenin N, M'den küçük olmak üzere N adet fiziksel makineye belirlenen amaca uygun olarak nasıl yerleştirileceği, sanal makinelerin yerleştirilmesinde enerji tüketimi, maliyet yönetimi, kaynak paylaşımı gibi bir veya birkaç amaç esas alınarak fiziksel makineler üzerindeki yükün dengelenmesi hedeflenmektedir. Bu amaçlardan birden fazlasını aynı anda geçekleştirmek için çok amaçlı optimizasyon algoritmaları kullanılabilir. Bu tezde literatürde yaygın kullanımı olan çok amaçlı optimizasyon algoritmalarının sanal makine yerleştirme problemindeki başarımları ele alınmıştır. Hazırlanan benzetim ortamında elde edilen sonuçlar karşılaştırmalı olarak değerlendirilmiştir.

Bilim Kodu	:	92413
Anahtar Kelimeler	:	Bulut bilişim, CloudSim, Çok amaçlı evrimsel algoritmalar, Sanal makine yerleştirme problemi, Yük dengeleme
Sayfa Adedi	:	80
Danışman	:	Prof. Dr. Suat ÖZDEMİR

MULTIOBJECTIVE OPTIMIZATION BASED DYNAMIC LOAD BALANCING IN CLOUD COMPUTING

(M. Sc. Thesis)

Serap DÖRTERLER

GAZI UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

May 2018

ABSTRACT

Cloud computing ensures people to use sources such as hardware, operating system, software wherever they want over internet.Virtualization technology which is an indispensable part of cloud computing gives opportunity to employ resources such as memory, bandwidth, processor among multiple virtual machines. Virtualization technology is based on simultaneous execution of virtual machines on physical machines. Virtualization technology allows placement of *M* virtual machines to *N* physical machines, where *M>N*, by balancing the load on physical machines with respect to one or more goals like energy consumption, cost management, or resource sharing. In order to satisfy more than one goal simultaneously, multiobjective optimization algorithms are shown to be effective solutions in the literature. In this thesis, four well-known multiobjective optimization algorithms are realized to solve virtual machine placement problem under CPU maximization and energy consumption minimization constraints. Extensive simulation results for different performance metrics are comparatively discussed.

Science Code	:	92413
Key Words	:	CloudSim, Cloud computing, Load balancing, Multiobjective evolutionary algorithm, Virtual machine placement problem
Page Number	:	80
Supervisor	:	Prof. Dr. Suat ÖZDEMİR

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren, kıymetli tecrübelerinden faydalandığım danışmanım Prof. Dr. Suat ÖZDEMİR'e, ayrıca çalışmalarımda bana yardımcı olan Öğr. Gör. Dr. Murat DÖRTERLER'e teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

1. GİRİŞ

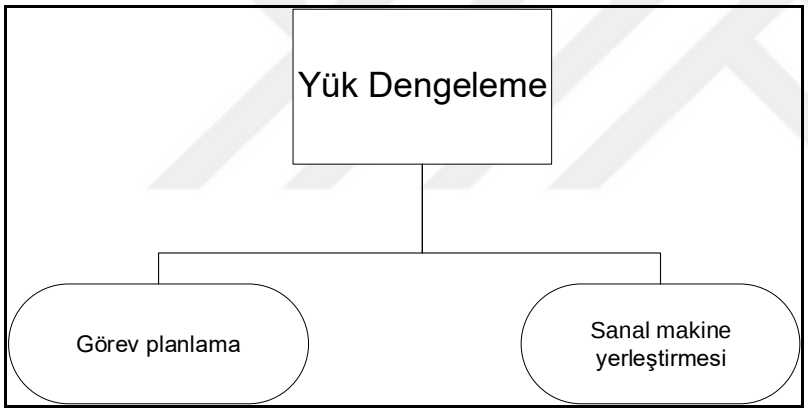
Bulut bilişim ağ, sunucular, depolama, uygulamalar ve hizmetler gibi bilgi işlem kaynaklarının ortak havuzuna talep üzerine ağ erişimi sağlayan bir modeldir. Bulut bilişim mimari olarak hizmet olarak alt yapı (Infrastructure as a service - IaaS), hizmet olarak platform (Platform as a Service - PaaS), hizmet olarak yazılım (Software as a Service - SaaS) isimleri altında üç kategoriye ayrılmaktadır. IaaS diğer "as a Service" modellerinin üzerine inşa edildiği bulut servislerinin temelidir. Bu özel uygulama müşterilere depolama, sunucu, ağ ve işletim sistemleri gibi temel bilgi işlem kaynakları sağlamaktadır. Bu uygulamada müşteriler donanım gibi altta yatan alt yapıyı kontrol edemezler fakat işletim sistemi, uygulamalar ve belki bazı ağ elemanları üzerinde yönetim hakkına sahiptirler. PaaS modeli ise IaaS'nin üzerine kurulum yapılmış halidir. IaaS'de müşteriye tanınan esneklikler PaaS'de önemli ölçüde ortadan kalkmaktadır. Örnek olarak Google App Engine [1], Microsoft Azure [2] gösterilebilir. SaaS'de ise PaaS'de azalan esneklikler bir miktar daha düşmektedir ve müşterilerin yetkileri çok kısıtlı olmaktadır. Örneğin, Google'ın elektronik posta hizmeti bir SaaS'dir. Burada müşterilerin sunucunun donanımı, işletim sistemi hakkında bir söz hakları veya servisin işlevselliğini genişletme yetenekleri bulunmamaktadır [3].

Sanallaştırma teknolojisi bulut bilişimin temelini oluşturmaktadır [4]. Genellikle donanım, sanal makine (Virtual Machine - VM) ve işletim sisteminden oluşmaktadır. Sanal makine, fiziksel kaynaklarla beslenen bilgisayar yazılımıdır. Uygulamalar ve işletim sistemleri tıpkı fiziksel bilgisayarlarda (Physical Machine - PM) olduğu gibi sanal makineler üzerine de kurulabilmektedir. Kaynak koruması ve verimli donanım kullanımıyla kullanıcıların kaynakları daha verimli kullanmalarını sağlamaktadır [5]. İş yükünün bir sunucudan başka bir sunucuya aktarılmasına imkân tanımaktadır. Bu imkân fiziksel makinenin kapasitesi yetmediğinde veya bir arıza durumunda kullanılabilmektedir. Bu durum sistemlerin erişilebilirlik durumunu yükseltmektedir [6].

Sanallaştırmanın sağladığı esneklik yeni yönetim zorlukları getirmektedir. Sanal makine havuzunun öngörülmesi ve yönetilmesi gerekmektedir. Kaynak talebi, maliyet, enerji tüketimi gibi amaçları önceliklerine göre değerlendirerek sanal makinelerin nereye yerleştirileceği ve kaynakların nasıl atanacağı belirlenerek cevap verilmelidir. Veri

merkezinin ölçeği büyüdükçe bu işlemin karmaşıklığı da artış göstermektedir. Bu sebeple bilgisayar destekli karar yapılarından yararlanılmaktadır [7].

Fiziksel makineler üzerinde bulunan farklı sayılardaki makinelerin değişken olabilen farklı miktardaki kaynak gereksinimleri başarımda düşüşe ve hizmet kalitesinde düşüşe neden olabilmektedir. Bu sıkıntıların çözümü için yük dengelemenin yapılması gerekmektedir [8]. Yük dengeleme Şekil 1.1'de görüldüğü gibi görev planlaması ve sanal makine yerleştirmesi (Virtual Machine Placement - VMP) yaklaşımlarıyla yapılabilmektedir. Görev planlaması kullanıcılardan gelen uygulama taleplerinin bir amaca yönelik olarak ilgili sanal makinelere dağıtımıdır. Sanal makine yerleştirilmesi ise sanal makinelerin yine bir amaç dâhilinde fiziksel sunuculara yerleştirilmesidir [9]. Sanal makine yerleştirme problemi çeşitli optimizasyon amaçları ve hedef uygulama merkezine ait özel gereksinimler doğrultusunda ele alınması zorunlu bir araştırma konusu olmuştur [10].



Şekil 1.1. Yük dengeleme çeşitleri

Literatürde sanal makine yerleştirme problemine yönelik çok amaçlı optimizasyon kullanılarak yapılmış çeşitli çalışmalar mevcuttur [11-23]. Bu çalışmalar amaç, kapsam, veri merkezinin ölçeği gibi çeşitli noktalarda farklılık göstermektedir. Bu tez çalışmasında sanal makine yerleştirme probleminin çözümüne yönelik yaygın kullanılan çok amaçlı optimizasyon algoritmalarından Non-Dominated Sorting Genetic Algorithm-II (NSGA-II), steady-state MOEA based on the ϵ-dominance concept (ϵ-MOEA), Pareto Archived Evolution Strategy (PAES) ve Strength-Pareto Evolutionary Algorithm 2 (SPEA2) uygulanarak başarımları kıyaslanmaktadır. Algoritmaların başarımını ölçmek için CloudSim benzetim ortamı ve Multiobjective Evalutionary Algorithm Framework (MOEA Framework) kütüphanesinden yararlanılmaktadır. Problemin çözümünde kullanılan çok

amaçlı optimizasyon algoritmalarında birden fazla amaçta iyileştirme hedeflenmektedir. Bu çalışmadaki amaçlar enerji tüketimini en aza indirmek ve fiziksel makine başına düşen ortalama işlemci (Central Processing Unit - CPU) kullanım miktarını artırmaktır. Bu tez çalışmasının literatüre katkısı bulut ortamında sanal makine optimzasyonunun literatürde daha önce kullanılmamış çok amaçlı optimizasyon algoritmalarıyla yapılarak sonuçların karşılaştırılmasıdır.

Bu tezin ilerleyen kısımları şu şekildedir. İkinci bölümde sanal makine yerleştirme problemi ve bu probleme getirilen çözümle ilgili literatürdeki çalışmalardan bahsedilmiştir. Üçüncü bölümde tez çalışmasında yararlanılan teknolojiler, kavramlar ayrıntılı bir şekilde sunulmuştur. Bulut bilişim, sanallaştırma teknolojisinin yanı sıra uygulamada kullanılan CloudSim ve MOEA Framework açık kaynak kodlu yazılımlar tanıtılmıştır. Problemin çözümü için kullanılan algoritmalar ayrıntılı olarak sunulmuştur. Dördüncü bölümde sanal makine yerleştirme problemi hakkında ayrıntılı bilgi verilmiştir. Beşinci bölümde tez kapsamında yapılan uygulama çalışması ve altıncı bölümde uygulama sonucunda elde edilen sonuçlar paylaşılmış ve değerlendirilmiştir. Yedinci bölümünde ise tez çalışmasının değerlendirilmesi ve geleceğe dönük çalışma önerileri yapılmıştır.

2. LİTERATÜR TARAMASI

Bu bölümde sanal makine yerleştirme problemini çok amaçlı optimizasyon ile ele alan çalışmalar araştırılmıştır. Bu çalışmalar amaç, kapsam, veri merkezinin ölçeği, karşılaştırıldıkları algoritmalar gibi çeşitli noktalarda farklılık göstermektedir.

[11]'de "A multiobjective ant colony system algorithm (VMPACS)" isimli çok amaçlı karınca kolonisi sistem algoritması tasarlanmıştır. Bu algoritmayla toplam kaynak tüketimi ve güç tüketimi en aza indirilerek sanal makineleri yerleştirme problemini çözmek amaçlanmıştır. Elde edilen sonuçlara göre önerilen VMPACS algoritması, çok amaçlı optimizasyon algoritması olan Multiobjective Grouping Genetic Algorithm (MGGA) ve tek amaçlı optimization algoritması olan First Fit Decreasing (FFD) ve Slave Ants Based Ant Colony Optimization (SACO) ile karşılaştırılmıştır. Sonuçta VMPACS'in MGGA, FFD ve SACO'dan daha verimli ve etkili olduğu tespit edilmiştir.

[12]'de Multiobjective Ant Colony Optimization (MACO) yerleştirme algoritması tasarlanmıştır. Bu algoritmayla toplam kaynak tüketimi, güç tüketimi ve ağ elemanları arasındaki iletişim masraflarını en aza indirerek sanal makine yerleştirme problemini çözmek amaçlanmıştır. MACO algoritması Cloudsim benzetim ortamında Micro Genetic Algorithm (MGA), Local Regression (LR), Dynamic Voltage and Frequency Scaling (DVFS) ve FFD algoritmalarıyla karşılaştırılmıştır. Sonuçta MACO'nun enerji tüketimi ve iletişim enerjisi maliyeti konusunda daha başarılı olduğu görülmüştür. Burada MGA çok amaçlı optimizasyon algoritmasıdır ve FFD, DVFS ve LR ise tek amaçlı sanal makine yerleştirme algoritmalarıdır.

[13]'de Cloud Adapted Feedback isimli algoritma sunulmuştur. Bu algoritmayla belirli bir işi daha az zaman ve maliyet ile bitirmek amaçlanmıştır. The Objective Case (Optimal Algorithm), Hystorical Statistical Algorithm ve Gene Exchange And Mutation Algorithm ile karşılaştırma yapılmıştır. Sonuçta sistemdeki ani değişikliklere adapte olmak konusunda en başarılı algoritmanın yazarların sunduğu Cloud Adapted Feedback Algoritması olduğu görülmüştür. Normal durumlar için de bu geçerlidir. Fakat sistem parametrelerindeki keskin değişiklik durumunda sunulan algoritmanın verimli olmadığı görülmüştür.

[14]'de iş yükünü sanal makinelere, sanal makineleri de fiziksel makinelere yerleştiren iki seviyeli kontrol sistemi önerilmiştir. Fuzzy multiobjective değerlendirme ile güçlendirilmiş genetik algoritma (MGGA) sunulmuştur. Kaynak israfını, güç tüketimini ve soğutma maliyetini en aza indirmek hedeflenmiştir. Önerilen MGGA algoritması FFD, BFD VE SGGA ile karşılaştırılmıştır. Sonuçta MGGA algoritmasının diğerlerine göre çakışan parametreler arasındaki dengeyi daha iyi sağladığı görülmüştür.

[15]'de bulut veri merkezlerinde sanal makineleri fiziksel makinelere yerleştirmeyi yöneten TOPSIS (Technique For Order Preference Similarity To Ideal Solution) tabanlı çok amaçlı optimizasyon yaklaşımı sunulmuştur. TOPSIS, farklı amaçlar (objectiveler) arasındaki çakışmayı dengelemek için kullanılan, çok kriterli karar verme tekniklerinden bir tanesidir. Önerilen çalışmada hangi sanal makinelerin nereye, ne zaman yerleştirileceğine karar vermek çözülmek istenen problemdir. Toplam kaynak tüketimini ve güç tüketimini azaltmak ve sunulan hizmet kalitesini artırmak hedeflenmiştir. CloudSim ortamında yapılan benzetimlerle önerilen Multiobjective Optimization Approach Based on TOPSIS (MOA-T), üç adet tek amaçlı optimizasyon - Single Objective Optimization Approaches (SOA) ile ve bir adet çok amaçlı optimizasyon algoritmasıyla karşılaştırılmıştır. Bahsedilen tek amaçlı optimizasyonlar MOA-T'nin ağırlıkları değiştirilerek oluşturulmuştur ve SOA-S, SOA-R, SOA-P isimleri verilmiştir; karşılaştırma yapılan çok amaçlı optimizasyon ise Multi-Objective Optimization Approach Based on Simple Additive (MOA-S) Algoritmasıdır. Sonuçta hizmet düzeyi anlaşması konusunda MOA-T'nin SOA-P ve SOA-R'den daha iyi olduğu, kaynak yükü ve güç tüketimi konusunda SOA-S, SOA-P ve MOA-S'den daha iyi olduğu ve makine taşınması konusunda diğer hepsinden daha iyi olduğu görülmüştür.

[16]'da kaynak kullanımı ve makine taşıma zamanları kriterleri göz önüne alınarak makine yerleştirmesi problemini çözmeyi amaçlayan Improved Multiobjective Particle Swarm Optimization (IMOPSO) algoritması sunulmuştur. İki adet benzetim çalışması yapılmıştır. Birinci çalışmada IMOPSO, Quantum Particle Swarm Optimization (QPSO) ve Particle Swarm Optimization (PSO) ile ikinci çalışmada ise NSGA-II ile karşılaştırma yapılmıştr. Birinci çalışma sonucunda IMPOSO'nun uygulanabilir ve verimli olduğu görülmüştür. Diğer karşılaştırmada ise IMOPSO'nun popüslasyon çeşililiğinde daha iyi olduğu ve hızlı bir şekilde pareto fronta yakınsadığı görülmüştür.

[17]'de "An Improved Evolutionary Multiobjective Optimization Algorithm" (NS-GGA) isimli bir algoritma tasarlanmıştır. Bu algoritmada NSGA-II'nin baskılanamayan sıralama özelliği ve Grouping Genetik Algoritma'nın (GGA) genetik operatörleri kullanılmıştır. Hedeflenen amaçlar aktif fiziksel makine sayısının azaltılması, iletişim trafiğinin azaltılması ve çok boyutlu kaynak kullanımının dengelenmesidir. GGA, BA, Cluster and Cut ve Greedy yöntemleri ile karşılaştırılmıştır. Sonuçta NS-GGA isimli algoritmanın diğer yöntemlerden daha başarılı olduğu görülmüştür.

[18]'de yeni bir algoritma önerilmemiş, varolan GA, NSGA ve NSGA-II algoritmaları karşılaştırılmıştır. Karşılaştırılan algoritmalardaki ortak amaç açık fiziksel makinelerin ortalama kaynak tüketimini artırmak ve yük dengelemeyi çoğaltmak, kaynak israfını en aza indirgemektir. C++ kütüphanesi olan, genetik algoritma parçalarını içeren GALib kütüphanesi kullanmışlardır. Sonuçta NSGA-II'nin diğer iki algoritmadan daha başarılı olduğu görülmüştür.

[19]'da "Multiobjective Biogeography Based Optimization For Virtual Machine Placement" (VMPMBBO) isimli bir algoritma önerilmiştir. Güç tüketimi, kaynak israfı, sunucu düzensizliği, sanal makineler arası trafik ve sanal makine taşıma zamanını en aza indirgemek hedeflenmiştir. VMPMBBO algoritması, MGGA ve VMPASC algoritmalarıyla karşılaştırılmıştır. Sonuçta VMPMBBO algoritmasının daha iyi yakınsama özelliğinin olduğu, hesaplama olarak daha verimli olduğu ve daha güçlü olduğu görülmüştür.

[20]'de "Multiobjective optimization with stabilization" (MOS) isimli bir algoritma önerilmiştir. Sanal makine taşınması, toplam taşınma süresi, toplam güç tüketimi, toplam ısıl ihlal süresi, toplam kaynak kullanım ihlal süresini azaltmak hedeflenmiştir. Testler yapılırken IBMBladeCenter'dan yararlanılmıştır. Sonuçta gereksiz sanal makine taşınmasının %80 oranına kadar azaldığı; dengesiz fiziksel makine seçiminden kaçınıldığı; uygulama performansının %30'a kadar yükseldiği ve kaynak kullanım verimliliğinin de %20'ye kadar yükseldiği görülmüştür.

[21] "Bulut Bilişimde Sanal Makine Yerleştirme Projelerine Genel Bakış" isimli, bulut bilişim ve veri merkezleri için literatürde sunulan sanal makine yerleştirme projelerinin analiz edildiği araştırma makalesidir. Çalışmalar yerleştirme yapılırken kullanılan

algoritmaya göre sınıflandırılmıştır. Makalede etkili yerleştirme algoritmalarının gereksiz makinelerin kapatılmasını sağlayarak enerji tüketiminin azaltılabileceğinden bahsedilmiştir. Literatürde taranan çalışmalarda çoğu projede bulut bilişimde performans ve enerjiyle alakalı konularda iyileştirme yapmanın hedeflendiği; güvenlikle ilgili konuların ihmal edildiği görülmüştür. Günümüzde "VM Escape Attacks, VM Sprawling Attacks, Cloud-Internal Denial Of Service Attacks (CIDos), VM Neighbor Attacks" gibi saldırılar sistemi sanal makine göçlerine zorlayarak sisteme yük getirmektedir. Bunun da bulut bilişimin gelecekteki bilgi sistemlerinde kritik rol oynamasına engel olabileceği belirtilmiştir.

[22]'de sanal makine yerleştirme probleminde kullanılan "Grouping Genetic Algorithm"in çoğu durumda verimli çalışmadığı belirtilmiştir ve bu algoritmayı iyileştirmek amaçlanmıştır. "Vector packing" problemi kullanılarak sanal makine yerleştirme problemi modellenmiştir ve kullanılan makine sayısı azaltılarak enerji tüketimi azaltılmak istenmiştir. Ayrıca kaynak kullanım verimliliği de artırılmaya çalışılmıştır. Improved Grouping Genetic Algorithm (IGGA) kodlama ve çarprazlama metodu sunulmuştur. Yapılan testler sonucunda ortalama enerji tüketimi ve kaynak kullanım verimliliği konularında IGGA, FFD ve Hybrid Grouping Genetic Algorithm (HGGA)'ya göre daha iyi sonuçlar elde edilmiştir.

[5]'de eş zamanlı olarak enerji kullanımını azaltmak, çok boyutlu kaynak kullanımını dengelemek ve veri merkezindeki iletişim trafiğini azaltmayı amaçlayan sanal makine yerleştirme modeli sunulmuştur. Problemi çözmek için yerel sezgisel metot ve seçicilik (elitism) stratejisi ile iyileştirilmiş genetik algoritma geliştirilmiştir. Benzetim sonuçlarına göre sunulan model ve algoritma GREEDY, Genetik Algoritma (Genetic Algorithm - GA), Arı Algoritması (Bees Algorithm - BA) ve CLUSTER ile karşılaştırıldığında kaynak kullanımının arttığı çok boyutlu kaynak kullanımının dengelendiği ve iletişim trafiğinin azaldığı görülmüştür. Yazarlar önerdikleri metodun çözüm hızını artırdığını fakat bunun sanal makine üretim aşamasında tüketildiğini belirtmişlerdir ve bu yüzden de ilerde sanal makine üretme ve yeniden yapılandırma sistemi tasarlamayı planlamışlardır.

[23]'de aile geni (family gene) yaklaşımını kullanan sanal makine yerleştirmesi için yeni bir teknik sunulmuştur. Sunulan yöntemde genetik algoritmanın erken yakınsama (prematüre convergence) ve yüksek işlem zamanı gibi sorunların üstesinden gelmek

hedeflenmiştir. Cloudsim'de yapılan benzetim çalışmasında enerji tüketiminin azaldığı gözlemlenmiştir. Ayrıca host başına düşen hizmet seviyesi anlaşması ihlali (SLAV) zamanı artarken sanal makine göçünün azaldığı görülmüştür.

Çizelge 2.1. Literatürdeki yöntemlerin karşılaştırmalı özet tablosu

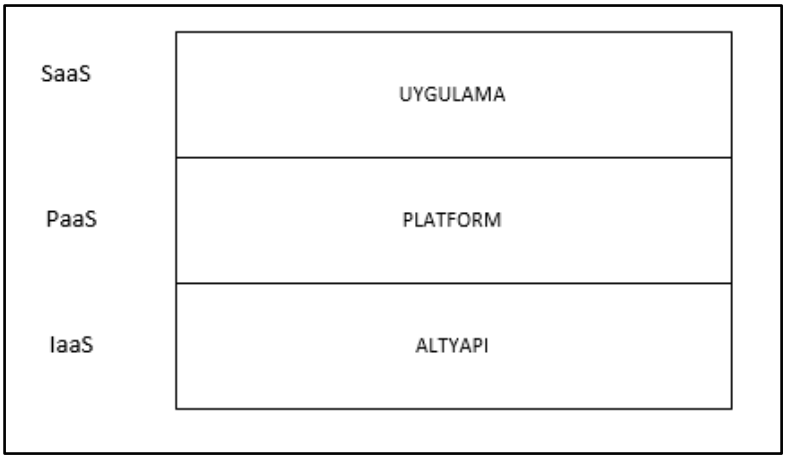
Makale	Yöntem	Amaçlar
[11]	VMPACS, MGGA, SACO, FFD	Toplam kaynak israfını azaltmak, enerji tüketimini azaltmak
[12]	MACO, MGA, DVFS, FFD, LR	Toplam kaynak israfını azaltmak, enerji tüketimini azaltmak, ağ elemanları arasındaki enerji tüketim maliyeti
[13]	Cloud Adopted Feedback Algorithm, Optimal Case Algorithm, Historical Statistical A., Genetic Algorithm	İş tamamlama süresini azaltmak, maliyeti azaltmak
[14]	MGGA, FFD, BFD, SGGA	Toplam kaynak israfını azaltmak, enerji tüketimini azaltmak ısı kaybını azaltmak
[15]	MOA-T, SOA-S, SOA-R, SOA-P, MOA-S	Toplam kaynak gerilimini azaltmak, enerji tüketimini azaltmak, hizmet kalitesini artırmak
[16]	IMOPSO, NSGA-II, QPSO, PSO	Kaynak kullanımını artırmak, Taşınma zamanlarını azaltmak
[17]	NS-GGA, NSGA-II ve GGA	Aktif PM sayısını azaltmak, iletişim trafiğini azaltmak, çok boyutlu kaynak kullanımını dengelemek
[18]	GA, NSGA, NSGAII	Ortalama kaynak kullanımını artırmak, yük dengelemeyi artırmak, kaynak israfını azaltmak
[19]	VMPMBBO, VMPACS, MGGA	Kaynak israfını azaltmak, enerji tüketimini azaltmak, sunucular arasındaki yük dengesini sağlamak, VM'ler arası ağ trafiğini azaltmak, depolama diski trafiğini azaltmak, VM taşınma maliyetini azaltmak
[20]	MOS, MONS, SOC, SOI, SOF	VM taşınmasını azaltmak, toplam taşınma zamanını azaltmak, enerji tüketimini azaltmak, toplam ısı ihlal süresini azaltmak, toplam kaynak kullanım ihlali süresini azaltmak
[21]	Araştırma Makalesi	Enerji tüketimini azaltmak, kaynak kullanımı verimini artırmak
[22]	IGGA, FFD, HGGA	Enerji tüketimini azaltmak, kaynak kullanımı verim
[5]	Greedy, GA, BA, CLUSTER	Enerji tüketimini azaltmak, çok boyutlu kaynak kullanımı dengelemek
[23]	FGA, THR, LR, LRR, IQR, MAD	Fiziksel kaynak kullanımını (CPU, RAM, BW) artırmak
[Bu tez çalışması]	NSGA-II, SPEA2, ϵ -MOEA, PAES	Aktif PM başına düşen ortalama CPU kullanımını artırmak, enerji tüketimini azaltmak

3. KULLANILAN YÖNTEMLER VE METODOLOJİ

Bu bölümde tez çalışmasında kullanılan teknolojiler olan bulut bilişim, bulut bilişimin vazgeçilmez parçası olan sanallaştırma teknolojisi, sanal makine yerleştirme probleminin çözümünde kullanılan çok amaçlı evrimsel optimizasyon algoritmaları (Multiobjective Optimization Evaluation Algorithms - MOEA) ve bu algoritmalardan NSGA-II, SPEA2, PAES ve ϵ-MOEA açıklanmıştır. Çalışma yapılırken veri merkezinin oluşturulduğu benzetim ortamı olan CloudSim yazılımı ve yukarıda bahsedilen algoritmaların çalıştırılması için kullanılan MOEA Framework aracı da anlatılmaktadır.

3.1. Bulut Bilişim

Bulut bilişim ağ, sunucu, depolama, uygulama ve servis gibi bilişim kaynaklarından oluşan bir havuza her zaman, her yerden, talep halinde ağ üzerinden erişim sağlayan bir modeldir [52]. Bulut ortamı, veri merkezinin donanımı ve yazılımından oluşmaktadır. Bulut ortamları özel ve genel olmak üzere kullanıcılarına göre ikiye ayrılmaktadır. Özel bulut ortamı, kurum ve kuruluşların kendilerinin oluşturduğu ve sadece kendi içinde kullandığı ortama denilmektedir. Genel bulut ise kaynaklarını herkesin ödediği bedel karşılığında kullanabildiği ortamdır [53]. Bulut bilişim dağıtım modellerine göre gruplandığında Şekil 3.1'de görüldüğü gibi SaaS, PaaS, IaaS olmak üzere üçe ayrılmaktadır[51]. IaaS'de bilgisayar donanım alt yapısı kullanıcının hizmetine sunulmaktadır. PaaS'de yazılım kullanıcıya lisanslanır ve kullanıcı istediği zaman o yazılımı kullanma hakkına sahip olmaktadır. Veri tabanı yönetim yazılımı, uygulama geliştirme ortamı buna bir örnek olarak verilebilir. SaaS ise web uygulalarının kişilerin kullanımına sunulmasıdır. Google'ın eposta hizmeti buna örnek olarak verilebilir.



Şekil 3.1. Bulut bilişimin dağıtım modellerine göre katmanları

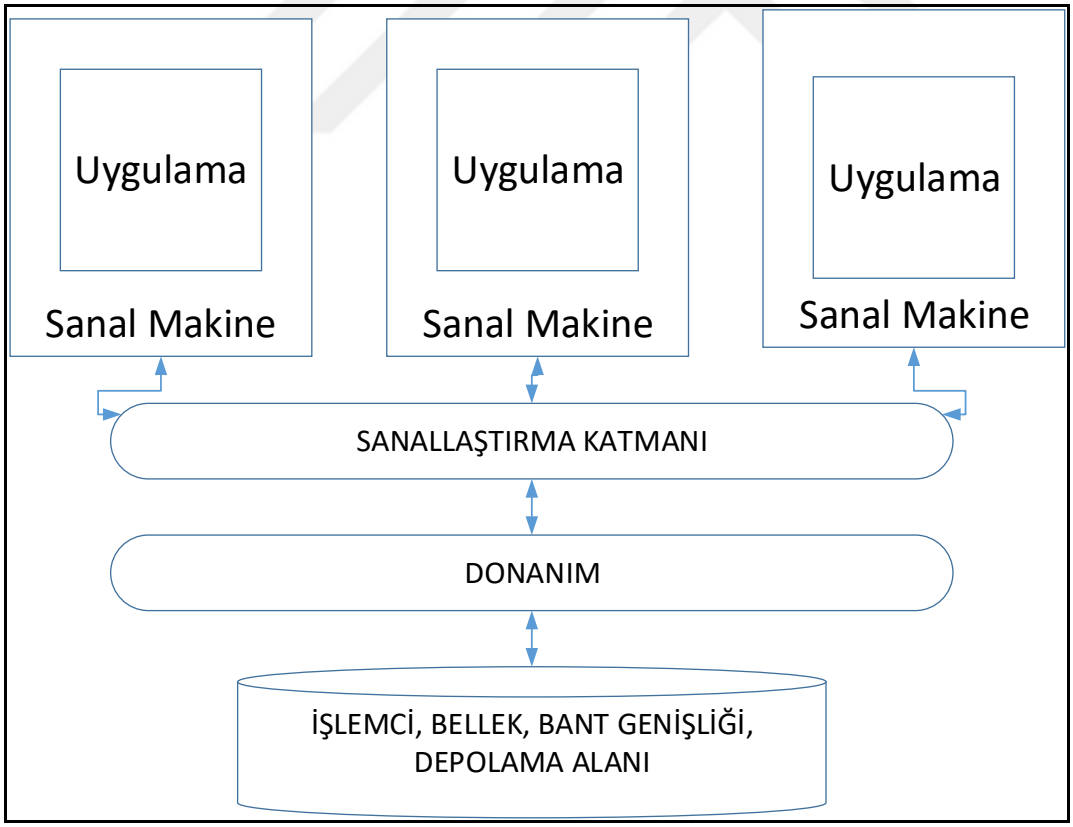
Kuruluşların gün geçtikçe bilişim sistemlerine bağlılığı artmaktadır. Bulut ortamlarının kurulumu ve yönetilmesi de insan kaynağı ve yüksek maliyet gerektirmektedir. Bilişim hizmetlerinden yararlanmak isteyen kişi veya kuruluşlar bulut bilişimden yararlanarak altyapı kurulumu, yeni personel eğitimi ve yazılım lisansı alma gibi yönetimsel ve mali zorluklardan kurtulabilmektedirler [55]. Örneğin bulut bilişim sayesinde yenilikçi bir internet hizmeti fikri fiziksel kaynak satın alınmadan bulut üzerine kurularak insanların kullanımına sunulabilmektedir [54]. Bulut bilişim hızlı, esnek, düşük maliyet gibi avantajlar sağlamaktadır fakat güvenilirliği çok yüksek değildir. Birçok kullanıcının verilerinin aynı yerde bulunması bir risktir. Bunun dışında kullanıcı verileri bulut bilişim sağlayıcılarının denetimine bırakıldığından veriler de risk altında bulunmaktadır.

3.2. Sanallaştırma

Geleneksel yöntemlerde uygulamalar fiziksel makineler üzerine kurulmaktadır ve bu da enerji israfı, yer israfı, düşük kaynak kullanımı ve önemli yönetim giderleri gerektirmektedir. Bunlar da maliyeti artırmaktadır. Günümüzde kullanılan sanallaştırma teknolojisi ise daha esnek, güvenli ve isteğe bağlı olarak gerektiğinde tahsis edilerek kaynak ayrımında esneklik sağlamaktadır. Sanallaştırma teknolojisinin kullanıldığı bulut veri merkezinde hesaplama (sunucular), depolama ve ağ cihazları bulunmaktadır ve bunlar ağ üzerinde dağıtık halde bulunabilmektedir. Fiziksel sunucular coğrafi olarak ayrılmış olarak yani farklı veri merkezlerinde de tutulabilmektedirler. Fiziksel sunucu bilgisayar, CPU, bellek (Random Access Memory- RAM), depolama gibi elemanlarıyla hesaplama yetisine sahiptir [49]. Veri merkezinde bulunan diğer düğümlerin de CPU, bellek ve ağ

bant genişliği (Bandwitdh - bw) gibi özellikleri vardır. Veri merkezi bünyesinde barındırdığı çok sayıda fiziksel sunucuyu yönetmektedir ve bu fiziksel sunuculara yerleştirme politikasına göre bir veya birden fazla sanal makine atanabilmektedir. Bulut veri merkezlerinin yönetiminde kaynak planlaması işinin önemli bir rolü bulunmaktadır. Bu kaynak planlaması yapılırken amaca göre karar verilmektedir. Örneğin yük dengeleme ve enerji tasarrufu farklı yerleştirme politikaları gerektirebilmektedir. Sanal makinelerin yerleştirme, taşınma işlemlerine karar vermek zorlayıcı planlama problemlerindendir.

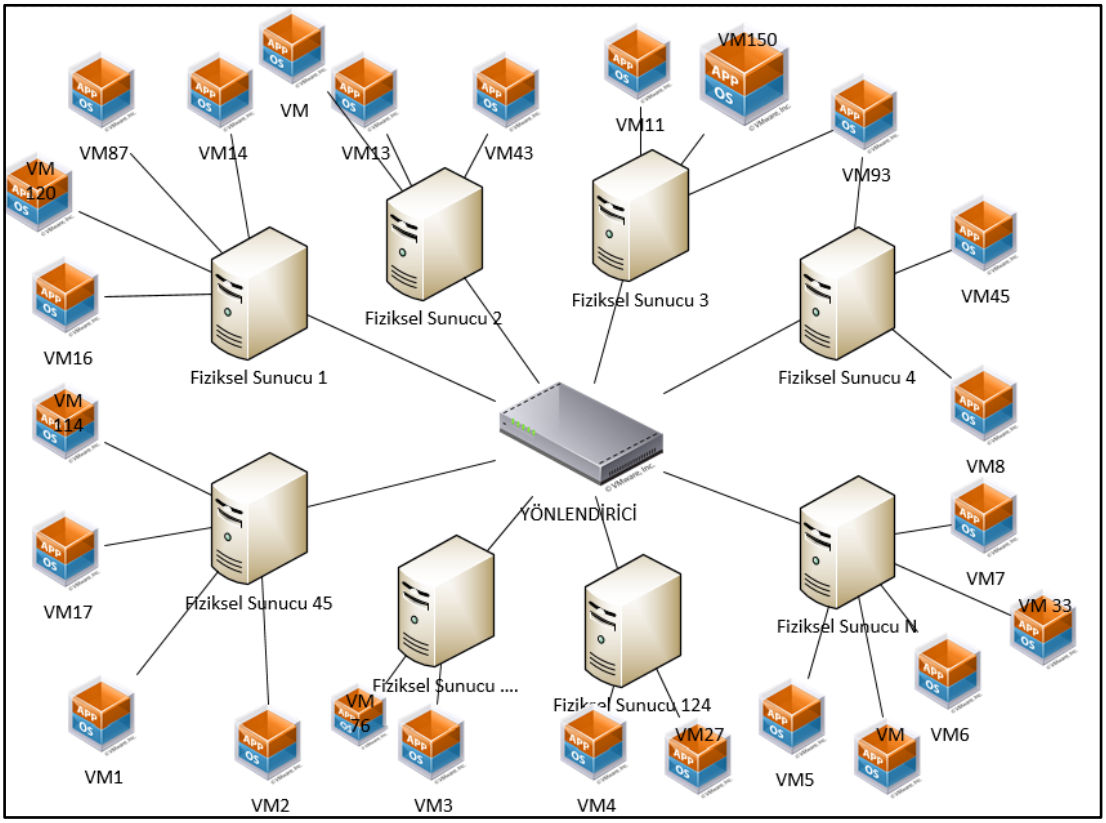
Sanallaştırma teknolojisinin mimari yapısı Şekil 3.2'de görülmektedir. Sanal makineler, üzerinde bulunduğu fiziksel makinenin kaynaklarını kullanmaktadır [26]. En popüler sanallaştırma yazılımları VMware [63], Microsoft [64] ve Citrix [65] tarafından üretilen yazılımlardır. Sanallaştırma enerji tasarrufu, maliyet azaltma sağlamaktadır. Sanallaştırma katmanı hipervisor olarak da bilinmektedir. Sanallaştırma katmanı sanal makinelere işlemci, bellek ve ağ bant genişliği sağlamaktadır [26].



Şekil 3.2. Sanallaştırma teknolojisinin mimari yapısı [26]

Sanallaştırma teknolojisinin sağladığı sanal makinelerin dinamik yerleştirilmesi kaynakları etkili kullanmayı ve enerji tüketimini azaltmayı sağlamaktadır. Atıl durumdaki sunucular

üzerlerindeki sanal makineler alınarak pasif duruma (düşük enerji modu veya uyku modu) geçirilerek enerji tüketimi azaltılabilmektedir. Sanallaştırma teknolojisinin sağladığı bir imkân olan canlı taşıma, sanal makinelerin hizmet kesintisi olmadan bir fiziksel sunucudan başka bir fiziksel sunucuya taşınmasıdır. Eğer bir uygulamanın kaynak gereksinimleri (işlemci, bellek vb.) karşılanamazsa uygulamanın cevap zamanı gecikme hatası veya hizmet kesintisi gibi sorunlarla karşılaşılabilmektedir. Bu yüzden bulut sağlayıcılar enerji tüketimini azaltırken performansın düşmemesi için performans enerji arasında denge kurmalıdırlar [29].



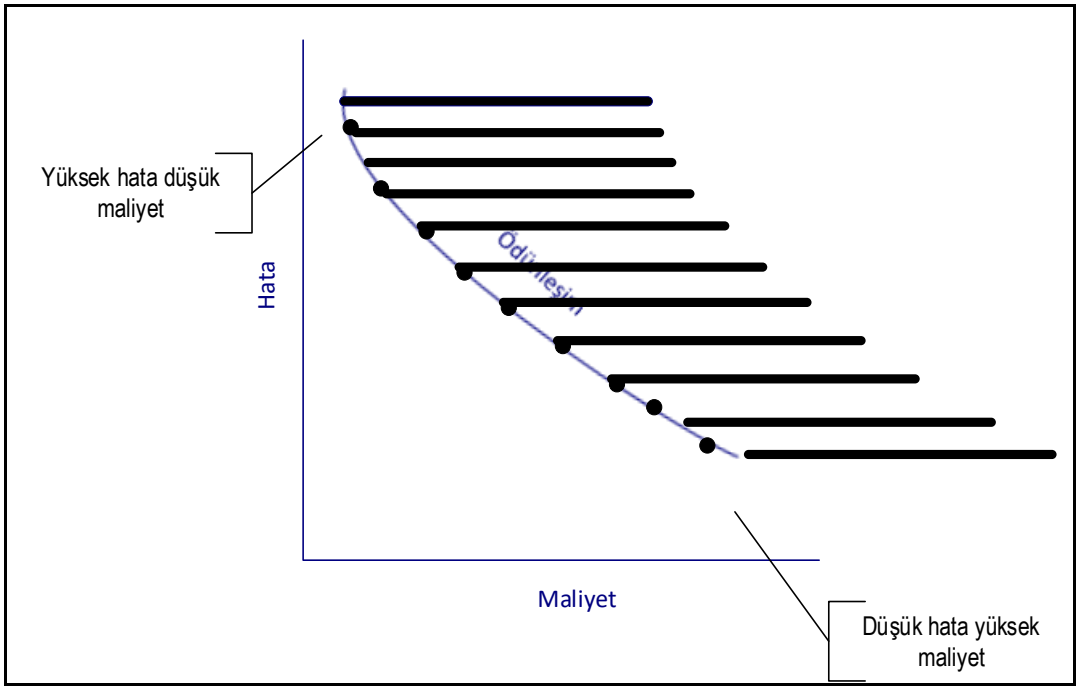
Şekil 3.3. Sanallaştırma ortamındaki fiziksel ve sanal makineler

3.3. Çok Amaçlı Optimizasyon

Optimizasyon, bütün mühendislik alanlarında kaşılaşılan matematiksel problemlere getirilen bir çözüm yöntemi olup bir veya birden fazla amacın sınır değerleri arasında kalan uygulanabilir çözümleri bulmaktır [34,56]. Birden fazla amaç fonksiyonunu sistemli ve eşzamanlı olarak optimize etme işine çok amaçlı optimizasyon veya vektör optimizasyonu denilmektedir [57]. Çok amaçlı algoritmalar üç gruba ayrılmaktadır. NSGAII [30], SPEA2 [45] pareto tabanlı, IBEA [46], SMS-MOEA [41] gösterge tabanlı

ve MOEA/D [42], MOEA/D-IR [43] dekompozisyon tabanlı çok amaçlı optimizasyon algoritmalarıdır [44]. Çoğu çok amaçlı optimizasyon algoritması çoğul amaçları tek amaca çevirmeye çalışmaktadır. Tek amaçlı ve çoğul amaçlı optimizasyon arasında temel farklılıklar bulunmaktadır. Tek amaçlı optimizasyonda bir tek optimum çözüm sunulurken; çok amaçlı optimizasyonda birden fazla optimum çözüm sunulmaktadır [50]. Tek amaçlı optimizasyonda tek bir kritere odaklanılmaktadır ve diğer kriterlere hiçbir şekilde bakılmamaktadır. Örneğin bir alıcı bir ev alırken fiyata göre karar verirse gidip en ucuzunu alabilmektedir, bu durumda aldığı ev şehir merkezinden çok uzakta olacaktır. Eğer ev seçerken kriterlerine hem şehir merkezine yakınlığı hem de fiyatını katarsa biraz maliyetten biraz da şehir merkezine yakınlığından ödün verebilmektedir.

Çok amaçlı optimizasyonda birden fazla en iyi çözüm olmasının sebebi amaçların birbiriyle çakışmasıdır [34]. Çok amaçlı optimizasyonlar çözüm olarak farklı avantaj ve dezavantajları olan birden fazla seçenek sunmaktadır. Bu çözüm seçeneklerinden oluşan kümeye Pareto Optimal Küme (Pareto Optimal Set - POS) denilmektedir. Bu çözümlerin grafik üzerinde birleştirilmesiyle oluşan, Şekil 3.4'de görülen eğriye ise Pareto Optimal Front (POF) denilmektedir [30].

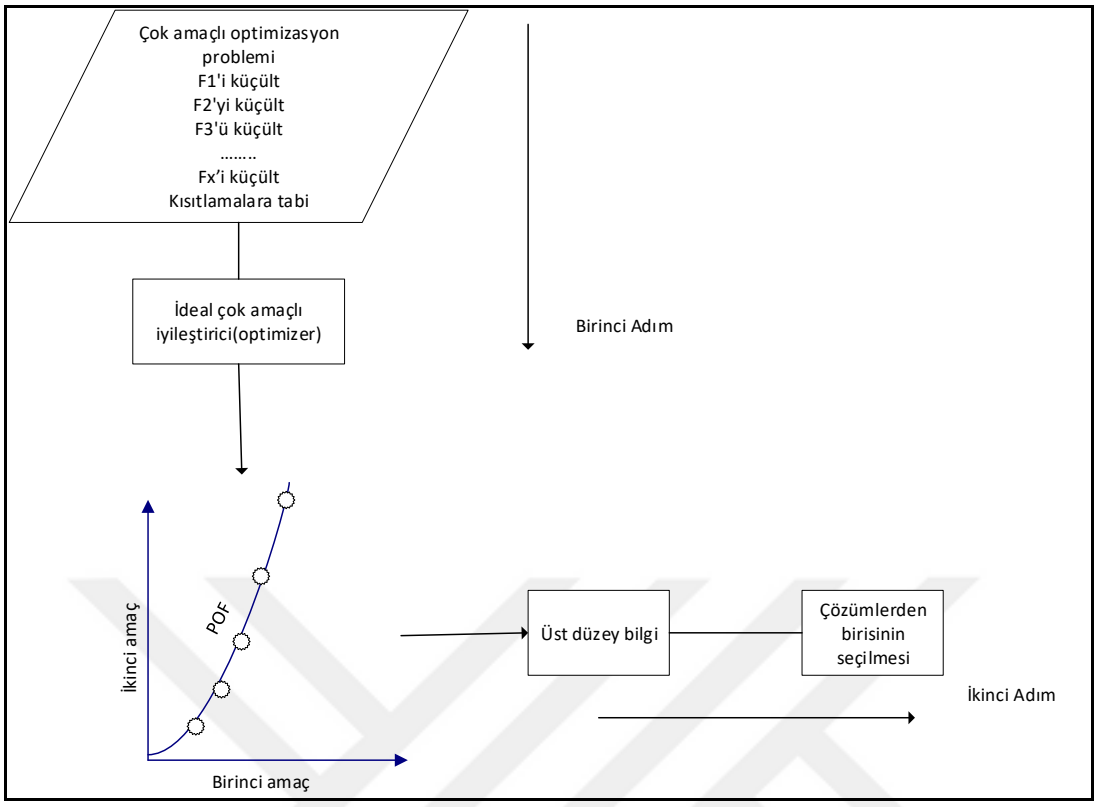


Şekil 3.4. Pareto Optimal Front eğrisi [28]

Çok amaçlı optimizasyonda amaçlar arasında ödünleşim bulunmaktadır (bknz. Şekil 3.4). Çizgili alanda kalan çözümler farazi probleme aday çözümlerdir. Sol üst alan aday çözümlerden düşük maliyetli ve yüksek hatalı olanları göstermektedir. Tek bir en iyi çözüm olmadığından hedefler arasında değişen miktarda ödünleşim olan çok sayıda potansiyel çözüm bulunmaktadır. Karar vericiler bu potansiyel çözüm kümesinden keşif yapmak ve uygulanacak olan çözüm veya çözümleri seçmekle sorumludur. Optimizasyon araçları bu karar verme sürecinde yardımcı olabilmektedir. Örneğin sonuçları azaltmakta fayda sağlayabilmektedir. Maliyet çok yükseldiğinde hatada sıra dışı bir azalma olması beklenmektedir. Bu türden bir analiz yapabilmek için soncul optimizasyon olarak bilinen numaralandırma ve ödünleşim tahmini yapılmaktadır [28].

Çok amaçlı optimizasyon sonucunda sunulan çözümlerden hangisinin seçileceği kişinin insiyatifindedir. Örneğin ev örneğinde ortaya çıkan en iyi çözümlerden maliyeti en düşük olan ya da şehir merkezine en yakın olan seçilebilmektedir. Bunun dışında her bir kritere bir ağırlık değeri verilerek tek amaçlı optimizasyona çevrilebilmektedir.

Çok amaçlı optimizasyon yönteminde eşzamanlı olarak birden fazla amaç optimize edilmeye çalışıldığından ve amaçlar birbiriyle çeliştiğinden problem zorlaşmaktadır. Bu tür problemlerin tamamında birden fazla amaç fonksiyonu bulunmaktadır ve bazılarında kısıtlar da bulunmaktadır. Şekil 3.5’de görüldüğü gibi birinci adımda çok amaçlı optimizasyon problemleri çözümünde elde edilen POS elemanlarından birisi ikinci adımda tercih edilmelidir.



Şekil 3.5. Çok amaçlı optimizasyon adımları [34]

3.3.1. Evrimsel çok amaçlı optimizasyon

Evrimsel çok amaçlı optimizasyon son yıllarda uygulama ve araştırma alanında popüler ve kullanışlı olmaktadır. Evrimsel optimizasyon algoritmaları popülasyon tabanlı yaklaşım içermektedir. Popülasyon, her bir döngüye katılan birden fazla çözüme denilmektedir. Herbir döngüde yeni bir popülasyon oluşturulmaktadır.

Evrimsel optimizasyon algoritmalarının popüler olma sebepleri çeşitlidir. Birincisi, evrimsel algoritmalar türetilmiş işlenmiş bilgiye gereksinim duymamaktadır. İkinci olarak evrimsel algoritmaların göreceli olarak uygulanması kolaydır. Üçüncüsü ise evrimsel algoritmalar esnektir ve yaygın bir uygulanabilirliğe sahiptir.

Evrimsel optimizasyon başlangıç adımından sonra sonlandırma şartı sağlanıncaya kadar şu adımları tekrarlayarak eldeki popülasyonu sürekli güncellemektedir:

1) Seçme (Selection)
2) Çarprazlama (Crossover)
3) Mutasyon (Mutation)

4) Seçkinleri Koruma (Elite preservation)

Başlangıç prosedürü genellikle rastgele çözümler oluşturulmasıyla başlar. Başlangıç popülasyonu rasgele çözümler üretilerek oluşturulabilmektedir fakat başlangıç popülasyonunda iyi özellikli çözümler olması tavsiye edilir. Bu, sonuca daha hızlı ulaşılmasını sağlamaktadır. Popülasyon üyelerinin uygunluk değerleri ölçüldükten sonra seçme operatörü vasıtasıyla seçilen bireyler ara eşleme havuzuna atılmaktadır.

Varyasyon operatörü birden fazla sayıdaki operatörün (örn. Çarprazlama, mutasyon) toplamıdır. Bunlar değiştirilmiş popülasyon oluşturmak için kullanılır. Çarprazlama operatörünün amacı eşleme ara havuzundan iki veya daha fazla ebeveyn seçmek ve ebeveynler arasında bilgi alışverişi yaparak bir veya birden fazla çözüm üretmektir. Ebeveyn bireylerin çarprazlamaya ne kadar katılacağını belirlemek için çaprazlama olasılığı kullanılmaktadır ($P_c \epsilon\, [0,1]$). Bireylerin kalan parçası (1-P_c) çocuk bireylere kopyalanmaktadır.

Mutasyon operatörü çeşitliliği sağlamak amacıyla bazı çözümleri değiştirmektedir. Mustasyon rasgele değişikliklere dayanmaktadır. Mutasyon operatörünün değiştirme gücüne mutasyon oranı denilmektedir. Herbir çözümün değiştirilme olasılığı P_m'dir. n değişken sayısı olmak üzere genellikle $P_m = \frac{1}{n}$ eşitliği ile hesaplanmaktadır. Böylece genellikle sadece bir değişken değiştirilmektedir. Bu operatör sayesinde mevcut çözümlerden çok uzaklaşılmamakla birlikte birbirinden farklı çözümler üretilmektedir.

Elitizm operatörü mevcut ve yeni popülasyonlarda bulunan çözümler arasından iyi olanların seçilmesini sağlamaktadır. Bu operatör de performansın düşmemesine katkıda bulunur.

Evrimsel optimizasyonu sonlandırmak için sonlandırma kriteri belirlenmelidir. Genellikle daha önceden belirlenen sayıda nesil (generation) üretildiğinde ya da belirlenen hedefe ulaşıldığında optimizasyon sonlandırılmaktadır.

Çok amaçlı optimizasyon probleminde en aza indirgenmesi veya en büyük değere çıkarılması gereken birden fazla amaç fonksiyonu bulunmaktadır. Çok amaçlı optimizasyonun matematiksel eşitliği aşağıdaki gibidir:

$$X = (x_1, x_2, x_3, \ldots . x_n)^T \quad (3.1)$$

Eş. 3.1 çözüm vektörünü ifade etmektedir.

Minimize/maksimize $f_m(X)$ $m = 1,2,3, \ldots . . M;$ (3.2)

Kısıtlar

$$g_j(X) \geq 0 \quad j = 1,2,3, \ldots . . J; \quad (3.3)$$

$$h_k(X) = 0 \quad k = 1,2,3, \ldots . . K; \quad (3.4)$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad i = 1,2,3, \ldots . . n \quad (3.5)$$

Eş. 3.1'deki X, popülasyonda bulunan bir çözümü temsil etmektedir. Her bir çözüm n adet karar değişkeninin (X_n) oluşturduğu bir vektördür. Eş. 3.2 M adet amaç fonksiyonunu ifade etmektedir. Bu fonksiyonlar minimize ya da maksimize edilebilmektedir. Eş. 3.3'deki $g_j(x)$ ve Eş. 3.4'deki $h_k(x)$ kısıt fonksiyonlarıdır. Buna benzer olarak çözümün geçerliliğini etkileyen diğer Eş. 3.5'teki değişkenlerin sınırları kısıtları ifade etmektedir. Herbir x, $x_i^{(L)}$ ve $x_i^{(U)}$ arasında olmalıdır [34].

3.3.2. Baskılanamayan çözümler (Non-dominated Solutions)

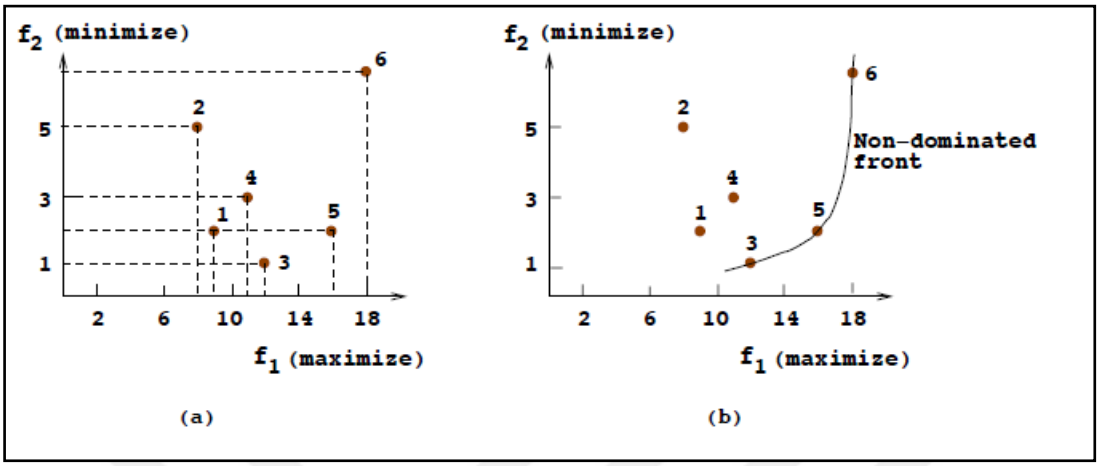
Çok amaçlı optimizasyonda çözümlerin birbirine baskınlığı söz konusudur. İki çözüm arasındaki baskınlık ilişkisi aşağıdaki gibi tanımlanır.

Eğer şu şartlar sağlanıyorsa "x_1 çözümü x_2 çözümüne baskındır" denir:

1) x_1 çözümü hiçbir amaç yönünden x_2'den daha kötü değildir. Bu karşılaştırma yapılırken amaç fonksiyonunun değerine veya grafikteki yerine bakılır.
2) x_1 çözümü en az bir amaç bakımından x_2'ye göre daha üstündür.

Popülasyonda bulunan her bir eleman diğer elemanlarla yukarda anlatıldığı şekilde karşılaştırılmaktadır. Herhangi bir x_1 elemanı x_2 elemanını baskılıyorsa, bunun tersi de doğrudur yani x_2 elemanı x_1'i baskılayamaz. Herhangi bir x_1 elemanı x_2 elemanını

baskılamıyorsa tersi doğru olmayabilir yani bu x_2 elemanının x_1 elemanını baskıladığı anlamına gelmez.



Şekil 3.6. Baskılanamayan çözümler kullanılarak çizilen "Non-dominated front" eğrisi [34]

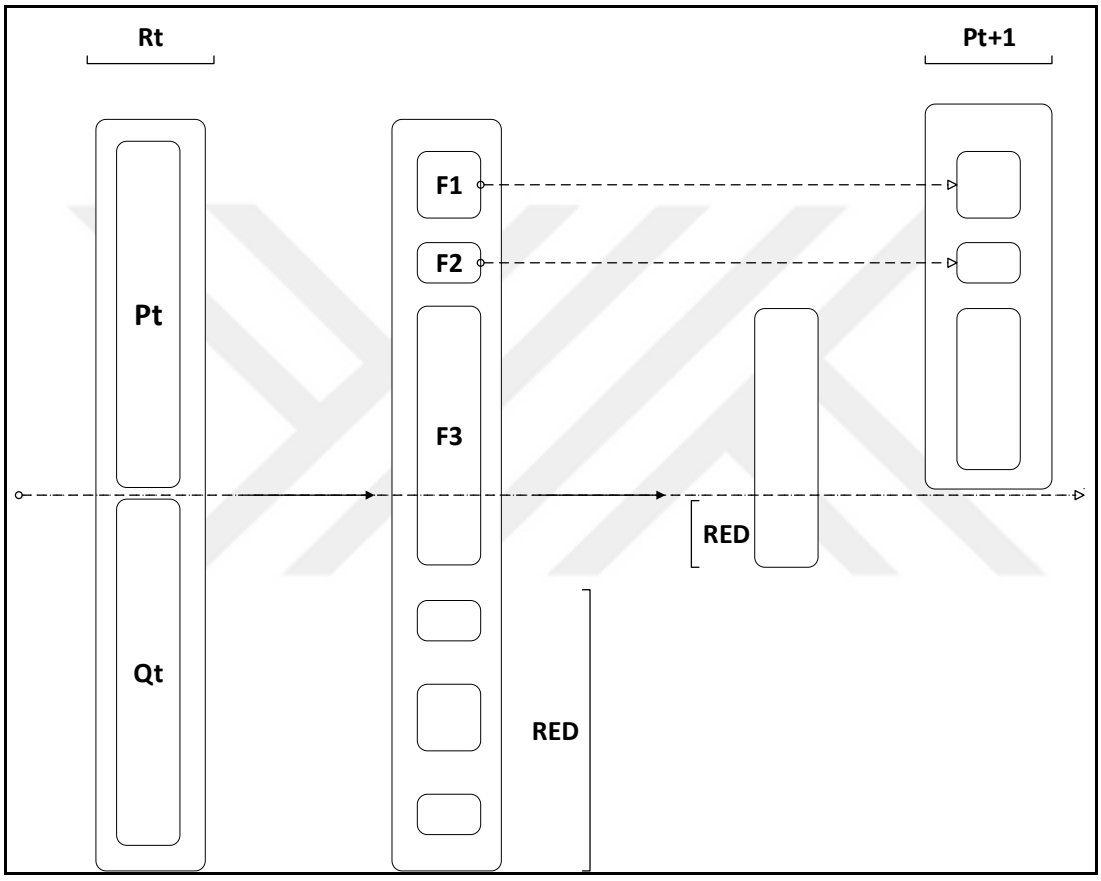
Çözüm kümesindeki elemanlar grafik alanında nokta ile ifade edilmektedir. Çözüm kümesinde bulunan herbir eleman bir diğeriyle tek tek karşılaştırılmaktadır, hangi çözümün diğerine baskın olduğu ve hangi çözümlerin birbirini baskılayamadığı belirlenmektedir. Daha sonra öyle bir küme oluşturulur ki bu kümedeki elemanların hiçbirisi bir diğerini baskılayamamaktadır. Bu kümeye baskılanamayan küme denilmektedir. Bu kümede bulunan elemanlar, kümeye ait olmayan elemanları baskılamaktadır. Baskılanamayan elemanları grafikte ifade eden noktalar kullanılarak bir eğri çizildiğinde bu eğriye Non-Dominated Front veya Pareto Optimal Front denilmektedir. Bu eğri Şekil 3.6'da örneklenmektedir. Kullanıcı, çözüm kümesi bulunduktan sonra çözümlerden hangisinin seçileceği konusunda bir ikileme düşmektedir (bknz. Şekil 3.5). Bu seçimi yapmak gerçekte teknik bilgi değil deneyim gerektiren bir durumdur. Çok amaçlı optimizasyonda amaç sayısı çok artırıldığında örneğin 3'ten 10'a artırıldığında %10 olan baskılanamayan eleman sayısı %90'a çıkmaktadır [34].

3.4. Kullanılan Algoritmalar

Yapılan çalışmada NSGA-II, ϵ-MOEA, PAES ve SPEA2 algoritmaları ile sanal makine yerleştirme benzetimi gerçekleştirilerek elde edilen sonuçlar karşılaştırılmıştır. Bu bölümde kullanılan algoritmaların açıklamaları yapılmaktadır.

3.4.1. NSGA-II

Kalyonmoy Deb ve arkadaşları NSGA-II [30] ismini verdikleri algoritmayı 2002 yılında geliştirmişlerdir. NSGA-II, NSGA [35]'nın iyileştirilmiş sürümüdür [36]. Evrimsel ve arşiv popülasyonu olmak üzere iki adet popülasyon kullanılmaktadır. Evrimsel popülasyon P_t ve arşiv popülasyonu ise Q_t ile gösterilmektedir.



Şekil 3.7. NSGAII algoritmasının görselleştirilmesi [30]

Başlangıçta rasgele ebeveyn popülasyonu P_0 oluşturulmaktadır. Daha sonra P_0 popülasyonu bireylerine İkili turnuva, çarprazlama ve mutasyon operatörleri uygulanarak N boyutundaki Q_0 arşiv popülasyonu oluşturulmaktadır. P_0 ve Q_0 popülasyonları oluşturulduktan sonra P_t ve Q_t popülasyonları birleştirilerek $2N$ boyutundaki R_t popülasyonu oluşturulmaktadır.

R popülasyonundaki bireyler seviye seviye oluşturulan pareto optimal kümelerine atanmaktadır. Bu kümelere Şekil 3.4'de görüldüğü gibi F_1, F_2,… F_N isimleri verilmektedir. Elemanların uygunluk değerleri de kaçıncı pareto optimal kümede olduklarına göre

belirlenmektedir. Örneğin F_1 pareto optimal kümesinde bulunan elemanların uygunluk değerleri 1 olmaktadır. F_1 pareto optimal kümesinde bulunan elemanlar en iyi elemanlardır. Bu durumda uygunluk değeri düşük olan elemanların daha iyi olduğu kabul edilmektedir. Daha sonra R_t popülasyonu elemanları uygunluk değerlerine göre sıralanmaktadır.

Şekil 3.7'de görüldüğü üzere eğer F_1'in eleman sayısı ebeveyn popülasyonu boyutu olan N'den küçükse F_1'in bütün elemanları yeni popülasyon olan P_{t+1}'e seçilmektedir. Eğer P_{t+1}'de boş kalan yerler varsa seviyelerine göre diğer kümelerin elemanlarıyla doldurulmaktadır. Önce F_2 kümesinin elemanları, sonra F_3 kümesinin elemanları seçilmektedir ve böyle devam etmektedir. Yerleştirme yapılan son pareto optimal kümesi örneğin F_3 olursa F_3'ün elemanları yoğunluk karşılaştırma operatörüyle azalan bir şekilde sıralanmaktadır. Bu sıralamadaki en iyi elemanlar kullanılarak yeni popülasyon P_{t+1}'de boş kalan yerler doldurularak yeni popülasyon oluşturma işlemi tamamlanmaktadır. Pareto optimal küme elemanları arasında sıralama yapılırken çözümün etrafındaki çözüm yoğunluğuna bakılmaktadır, yoğunluğu en az olan eleman seçilmektedir. Çözümlerin bulunduğu alanın yoğunluk değeri kalabalığa uzaklık tekniği ile bulunmaktadır. Kalabalığa uzaklık, çözümlerin bulunduğu alanın yoğunluk değeridir. Bu yöntem sayesinde çeşitlilik sağlanmaktadır.

İlk döngüde kullanılan Q_0 popülasyonu oluşturulduktan sonra diğer döngülerde kullanılan Q_{t+1} popülasyonu oluşturulurken ikili turnuva seçme yönteminin yerini yoğunluk karşılaştırmaya dayalı yöntem almaktadır. Bu yöntem sıralama ve kalabalığa uzaklık bilgileri kullanmaktadır. Bu bilgiler P_{t+1} oluşturulurken zaten hesaplandığı için yeni bir hesaplama maliyeti getirmemektedir [30].

3.4.2. ϵ-MOEA

ϵ-baskınlık kavramına [58] ve verimli ebeveyn ve arşiv güncelleme yönemlerine dayalı ϵ-MOEA [39] isminde bir algoritma Kalyonmoy Deb ve arkadaşları tarafından sunulmuştur.

Algoritmada iki türde popülasyon bulunmaktadır. Evrimsel popülasyon P_t ile, arşiv popülasyonu ise Q_t ile gösterilmektedir. Burada t, döngü sayısını ifade etmektedir.

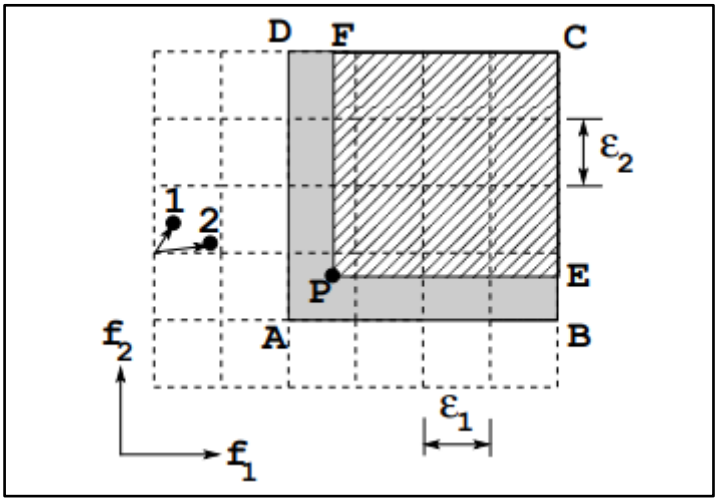
Başlangıçta Q_0 popülasyonu, P_0'ın ϵ-baskılanamayan çözümleri atanarak oluşturulmaktadır.

P_t'den rasgele iki birey seçilerek, bu iki bireyden diğerine baskın olan eleman eşleşme için seçilmektedir. Eğer birbirlerine baskın gelemezlerse rasgele birisi seçilmektedir. Seçilen elemana p denilmektedir. Q_t arşiv popülasyonundan da rasgele bir e bireyi seçilmektedir. p ve e çarprazlanarak çocuk birey oluşturulmaktadır.

Yeni oluşturulan çocuk bireyin kabul edilip edilmeyeceğinin belirlenmesi için Q_t ve P_t popülasyonundaki bireylerle karşılaştırılmaktadır. Çocuğun arşivdeki herbir elemanla ϵ-dominantlık karşılaştırması yapılmaktadır. Arşivdeki herbir çözüm B dizisiyle ifade edilmektedir. M amaç sayısı olmak üzere B vektörü $B = (B_1, B_2, \ldots . B_M)^T$ şeklinde ifade edilmektedir. B dizisi Eş. 3.6'ya göre hesaplanmaktadır.

$$B_j(f) = \begin{cases} \lfloor f_j - f_j^{min}/\epsilon_j \rfloor, & f_j' yi\ minimize\ etmek\ için \\ \lfloor f_j - f_j^{min}/\epsilon_j \rfloor, & f_j' yi\ maksimize\ etmek\ için \end{cases} \tag{3.6}$$

B tanımlama dizisi bütün amaç uzayını kutucuklara bölmektedir. Şekil 3.5'de görüldüğü gibi her bir kutucuk amaç numarası j'ye göre değişen ϵ_j boyutuna sahiptir.



Şekil 3.8. ϵ - dominantlık kavramı [39]

Arşivdeki herhangi bir eleman α ile, çocuk eleman ise c_t ile ifade edilmektedir.

1) α elemanı c_t'ye ϵ -baskınsa c_t elenmektedir.

2) c_t herhangi bir α'ya baskınsa α silinip yerine c_t koyulmaktadır.

3) 1. ve 2. şartların herikiside sağlanmıyorsa c_i arşiv elemanı tarafından ϵ-baskılanamayandır denilmektedir. Bu durumda

a. Aynı kutudalarsa yani aynı B vektörünü paylaşıyorlarsa klasik baskılanmazlık kontrolü yapılmaktadır. Eğer c_t, α'yı baskılarsa veya α, c_t'yi baskılayamazsa fakat B'ye α'dan daha yakınsa α silinip yerine c_t kabul edilmektedir.
b. Aynı kutuda değillerse c_t kabul edilmektedir.

Arşivin eleman sayısı sabit değildir. "Herbir kutuda sadece 1 eleman olacaktır" kuralı çözümlerin iyi dağıtılması ve final arşiv boyutunun pareto optimal çözümlerin toplam sayı boyutuyla sınırlanmasını ve çeşitliliği sağlamaktadır.

Çocuğun popülasyona kabul edilip edilmeyeceğini belirlemek için çocuk eleman popülasyondaki bütün çözümlerle karşılaştırılır. Eğer çocuk popülasyondaki bir elemanı baskılıyorsa onun yerine geçer, eğer çocuk birden fazla çözümü baskılıyorsa onlardan rasgele bir tanesi seçilerek onun yerine koyulur. Eğer çocuk eleman popülasyondaki herhangi bir elemana baskılanıyorsa popülasyona kabul edilmez. Eğer bu iki durum da sağlanmıyorsa çocuk eleman popülasyondan rasgele seçilen bir elemanla yer değiştirilmektedir. Böylece popülasyonun boyutu sabit kalmış olmaktadır.

3.4.3. PAES

Knowles ve Corne 2000 yılında evrimsel strateji kullanan çok amaçlı evrimsel algoritma olan PAES algoritmasını geliştirmişlerdir. PAES'de İlk önce rasgele çözüm p_0 yeni ebeveyni seçilmektedir. Sonra bu çözüm mutasyona uğratılmaktadır. Mutasyona uğramış ebeveyne çocuk denilmektedir ve c_t ile ifade edilmektedir. Başlangıçta yapılan p_t ve c_t karşılaştırması 3 senaryoda gerçekleştirilmektedir. Eğer p_t, c_t'ye baskınsa çocuk c_t elenmektedir ve yeni bir mutasyona uğramış çocuk ilerdeki süreçler için oluşturulmaktadır. Eğer c_t, p_t'ye baskınsa çocuk ebeveynden daha iyidir, bu durumda c_t bir sonraki nesile ebeveyn olarak kabul edilmektedir ve kopyası arşive kaydedilmektedir**.** Arşiv bu şekilde doldurulmaktadır. Arşivin boyutu PAES tarafından sürekli güncellenmektedir. p_t ve c_t birbirine baskın değilse karışıklık ortaya çıkmaktadır. Bu durumda çocuk o andaki arşivle karşılaştırılır (arşivde o ana kadar bulunan baskılanamayan çözümleri tutar). Burada 3 durum mümkündür:

1. Arşiv üyesi çocuğa baskındır. Çocuk, arşive alınmaz, ebeveyn p_t ilerki süreçlerde kullanmak için bir çocuk bulmak amacıyla mutasyona uğratılmaktadır.
2. Çocuk, arşivin bir üyesine baskındır, bu da çocuk, arşivin bazı üyelerinden daha iyi demektir. Arşivin baskılanmış üyeleri silinmektedir, onların yerine çocuk kabul edilmektedir. Çocuk bir sonraki neslin ebeveyni olmaktadır.
3. Arşivdeki hiçbir eleman çocuğu baskılayamamaktadır ve çocuk da arşivin hiçbir elemanını baskılayamamaktadır. Bu durumda çocuk arşiv çözümlerinin ait olduğu baskılanamayan cepheye aittir. Bu durumda, eğer boşluk varsa bir sonraki nesilde çocuk arşive alınabilmektedir. Çocuğun bir sonraki nesil için ebeveyn olup olamayacağına karar vermek için çevredeki çözümlerin yoğunluğuna bakılmaktadır. Çünkü p_t ve c_t'nin her ikisi de arşivin üyesidir. En az kalabalık bölgede bulunan birey ebeveyn olarak seçilmektedir. Eğer arşiv tamamen dolu ise ebeveyn veya çocuğun hangisinin arşivde kalacağını belirlerken yoğunluk tabanlı karşılaştırma uygulanmaktadır. Eğer çocuk arşiv üyeleri amaç uzayındaki en az kalabalık alanda bulunuyorsa ebeveyn olarak kabul edilmektedir ve bir kopyası arşive eklenmektedir. Kalabalık, bütün arama uzayı d^n alt alana bölünerek ve alt alanları dinamik olarak güncellenerek düzenlenebilmektedir. Burada d derinlik parametresi, n karar değişkenlerinin sayısıdır.

Her bir t jenerasyonunda (her bir döngüde) p_t ve c_t'ye ek olarak PAES o ana kadar bulunan en iyi çözümlerin tutulduğu arşivi de iyileştirmektedir. Başlangıçta bu arşiv boş olmaktadır, nesil ilerledikçe iyi çözümler arşive eklenmektedir ve güncellenmektedir [30].

3.4.4. SPEA2

SPEA2 algoritması Eckart Zitzler ve arkadaşları tarafından 2001 yılında sunulmuştur. SPEA2 algoritmasında N, popülasyondaki birey sayısını; $\overline{N}$ arşiv popülasyonundaki birey sayısını; T döngünün kaç defa tekrarlanacağını ve A baskılanamayan kümenin eleman sayısını belirtmektedir.

Evrimsel P_t ve arşiv popülasyonu Q_t olmak üzere iki adet popülasyonu bulunmaktadır. Başlangıçta arşiv popülasyonu boştur. P_t ve Q_t'deki herbir i bireyine güç değeri atanır. $S(i)$ değeri Eş. 3.7'de olduğu gibi i bireyinin baskıladığı çözümlerin sayısını göstermektedir.

$$S(i) = |\{j \mid j \in P_t + Q_t \ \wedge \ i \succ j\}| \tag{3.7}$$

Eş. 3.7'de $|.|$ simgesi eleman sayısını, $+$ simgesi birleştirmeyi, $\succ$ simgesi pareto baskınlık ilişkisi'ni ifade eder. Hesaplanan güç değeri uygunluk değeri hesaplarında kullanılmak üzere ham uygunluk değeri hesaplanırken kullanılmaktadır.

$$R(i) = \sum_{j \in P_t + Q_t, j \succ i} S(j) \tag{3.8}$$

Eş. 3.8'de görülen $R(i)$ fonksiyonu i bireyinin ham uygunluk değeridir ve i bireyini baskılayan bireylerin her birinin baskıladığı eleman sayıları toplamı ile hesaplanmaktadır. Burada i'yi baskılayan elemanların ne kadar güçlü olduğu bulunmaya çalışılmaktadır. Diğer bir ifadeyle ham uygunluk, arşiv ve popülasyondaki i bireyini baskılayan elemanların güç değerlerinin toplamıdır. Burada uygunluk değerinin az olması daha makbuldür. Örneğin baskılanamayan elemanların ham uygunluk değeri $R(i) = 0$'dır. $R(i)$ değerinin yüksek olması i bireyinin birçok birey tarafından baskılandığını gösterir. Ham uygunluk değeri belirleme işi bireylerin çoğu birbirine baskın değilse başarısız olabilmektedir. Bundan dolayı aynı ham uygunluk değerine sahip bireyleri ayırt edebilmek için yoğunluk bilgisiyle birleştirilmektedir. SPEA2'de en yakın k. komşu metodunun uyarlaması olan yoğunluk tahmin tekniği kullanılmaktadır, k. en yakın komşuya olan uzaklığın bir fonksiyonu kullanılmaktadır. Yoğunluk tahmini olarak k. en yakın komşuya olan uzaklığın tersi alınmaktadır. Daha açık bir ifadeyle her bir i bireyinin arşiv ve popülasyondaki her bir j bireyine uzaklığı hesaplanıp bir listede tutulmaktadır. Liste artan sırada sıralanırsa k. eleman aranan mesafeyi vermektedir. Bu mesafe değeri σ_i^k ile ifade edilmektedir. Ortak bir ayar olarak $k = \sqrt{N + \bar{N}}$ eşitliğiyle hesaplanmaktadır. Daha sonra i'ye bağlı yoğunluk değeri hesabı Eş. 3.9'da gösterilmektedir.

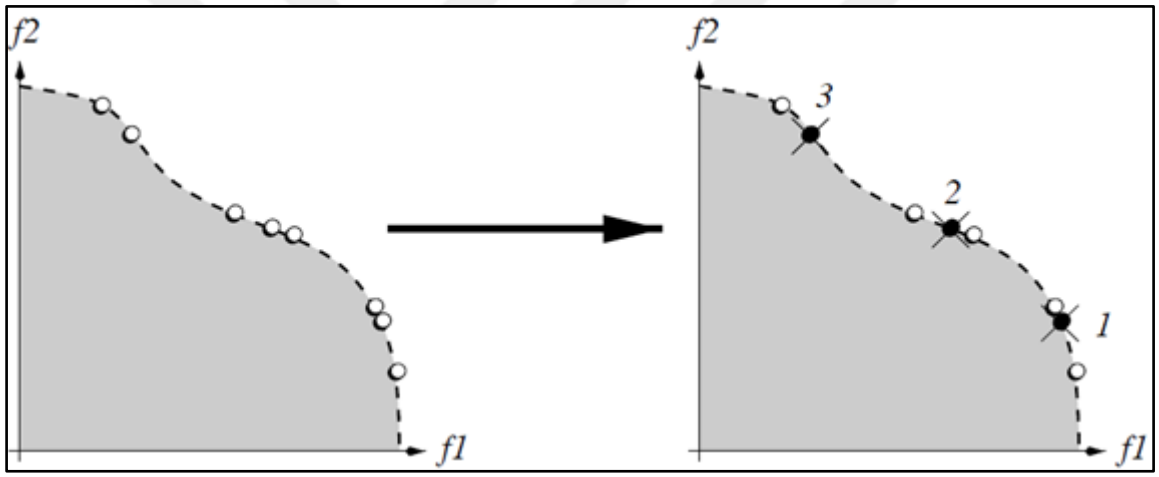
$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{3.9}$$

Eş. 3.9'da paydada 2 olmasının sebebi paydanın 0'dan büyük olmasını ve $D(i) < 1$ sağlamaktır. En son Eş. 3.10'da $D(i)$'ye $R(i)$ eklenerek uygunluk değeri $F(i)$ hesaplanmaktadır.

$$F(i) = R(i) + D(i) \tag{3.10}$$

Uygunluk değerlerine göre yeni arşiv popülasyonu oluşturulur. P_t ve Q_t'deki bütün baskılanamayan bireyler Q_{t+1} arşiv popülasyonuna kopyalanmaktadır. Eğer Q_{t+1} boyutu $\overline{N}$ sabit değerini aşıyorsa, arşiv kesme yöntemiyle bazı elemanlar elenmektedir. Arşiv boyutu $\overline{N}$ olana kadar döngü içinde, k. komuşsuna uzaklığı en düşük olan eleman elenmektedir. Eğer Q_{t+1} boyutu $\overline{N}$ sabit değerinden düşük ise bir önceki arşiv ve popülasyondaki baskılanan bireylerden en iyilerle doldurulmaktadır. Değerin düşük olması makbul olduğundan P_t ve Q_t'nin elemanları uygunluk değerine göre küçükten büyüğe doğru sıralanıp $F(i) \geq 1$ olanlardan en üsttekilerle doldurulmaktadır.

İkili turnuva yöntemi ile seçim yapıldıktan sonra çarprazlama ve mutasyon teknikleri kullanılarak yeni bireyler ile yeni popülasyon oluşturulmaktadır [45].



Şekil 3.9. Arşivin boyutunu azaltma yöntemi [45]

3.5. MOEA Framework

MOEA Framework [66], çok amaçlı evrimsel algoritmalar ve diğer genel amaçlı optimizasyon algoritmalarını geliştirmek ve deneyimlemek için açık kaynak kodlu ve ücretsiz Java kütüphanesidir. MOEA Framework 24 adet çok amaçlı optmimizasyon algoritmasını desteklemektedir. Örnek olarak NSGA-II, ϵ-MOEA, e-NSGA-II, PAES, PESA2, SPEA2, IBEA, SMS-EMOA, GDE3, SMPSO, OMOPSO, SMA-ES ve MOEA/D algoritmaları sayılabilir [28].

MOEA Framework'de üç adet ana sınıf bulunmaktadır. Bunlar *Executer*, I*nstrumenter* ve *Analyzer* sınıflarıdır. *Executer* sınıfı algoritmaları çalıştırırken kullanılır ve üç adet

parametre alır. Bu parametrelerden birincisi problem, ikincisi problemi çözmek için kullanılan algoritma, üçüncüsü ise iterasyon sayısıdır. *Instrumenter, executer* ile *executer*'ın verisini toplamak için el ele çalışmaktadır. *Executer* algortirmayı ayarlamak ve çalıştırmaktan sorumludur aynı zamanda algoritma çalışırken *instrumenter*ın gerekli veriyi kaydetmesine de izin vermektedir. *Executer*'ın çalıştırdığı algoritmaların dokümantasyonu *instrumenter* tarafından yapılmaktadır. *Analyzer* sınıfı ise çalışma bitiminin analizini sağlamaktadır. Bu analiz, pareto yaklaşım kümesine odaklanmaktadır ve bilinen bir referans kümesi ile karşılaştırmaktadır. *Analyzer* algoritmaların bulduğu sonuçları veya bir algoritmanın farklı parametrelerle bulduğu sonuçları karşılaştırmaktadır [28].

MOEA Framework'deki problemler *problem* ara yüzünü uygulamaktadır. *Problem* ara yüzü problemi özelleştirmek için metotları tanımlamaktadır. Yeni problem oluşturulurken *Problem* sınıfının uygulanması zorunlu değildir, istenirse *AbstractProblem* sınıfı genişletilerek yeni *problem* sınıfı oluşturulabilmektedir. *Problem* sınıfı oluşturulurken, *solution* sınıfının da oluşturulması gerekmektedir. Bu sınıfa parametre olarak karar değişkenlerinin sayısı ve amaçlar verilmektedir. Bunun dışında sınıfta *Evaluate* metodu vardır ve parametre olarak *solution* sınıfını almaktadır. Burada karar değişkenleri bir diziye atanabilmektedir. Sonra karar değişkeni sayısınca döngü içerisinde fonksiyon sonuçları hesaplanmaktadır. Bu değerler de çözüm sınıfına amaç olarak atanmaktadır. *Encodingutils* sınıfı çözüm içinden karar değişkenleri çıkarılırken kullanılmaktadır. Sonra bu karar değişkenleri problemi değerlendirmek için kullanılmaktadır. Bu aşamalardan sonra problem tanımlanmış olmaktadır ve MOEA tarafından kullanılabilir hale gelmektedir [28].

3.6. CloudSim

Gerçek ortamda sanal makine yerleştirme problemine çözüm araştırmak oldukça zorlu bir süreçtir. Ağ durumlarını tahmin etmek veya kontrol etmek neredeyse imkânsızdır. Ayrıca bir veri merkezi kurmak ya da test için kullanmak çok büyük maliyet gerektirmektedir. Bu tür araştırmaları yapmak için benzetim ortamınları daha elverişlidir. Benzetim uygulamaları; kaynak miktarı, veri merkezi, kullanıcıların sayısı ve kullanıcı bilgilerini de içeren iş yükü hakkında bilgi vermektedir. Benzetimler, planlanan çözümlerin gerçek sisteme uygulamadan önce test edilmesini sağlamaktadır. Gerçek ortamlara göre çok daha hızlıdır. Örneğin gerçek ortamda bir yıl sürecek bir süreci benzetim araçları birkaç dakikada tamamlayabilmektedir [27].

Bulut bilişimde kaynak kullanımını iyileştirmek, yük dengelemek, enerji tüketimini azaltmak, maliyetleri düşürmek araştırmacıların hedefleri arasındadır. Açık kaynak kodlu bulut benzetim ortamlarının kullanımı kodları inceleme, yeni algoritmalar geliştirme ve gerektiğinde iyileştirme yapmaya imkân tanımalıdır. Cloudsim, Icancloud, Greencloud, Cloudsched bu amaca yönelik geliştirilmiş örnek açık kaynak kodlu benzetim ortamlarıdır. Mimari, modelleme elementleri ve benzetim süreci benzetim araçlarının ortak özellikleridir [48].

Cloudsim yaygın kullanımı olan benzetim araçlarından birisidir, kolaylıkla geliştirilebilmektedir. Fakat paralel deneyimler ve sanal makinelerin yaşam döngüsünü dikkate almaması zayıf yönünü oluşturmaktadır. Cloudsim dışındaki benzetim araçları sonuçları bir ara yüzle göstermektedirler, Cloudsim programının ara yüzü bulunmamaktadır. Icancloud paralel deneyimi uygulamaktadır fakat enerji tüketimi ve sanal makine taşınmasını dikkate almamaktadır. Greencloud, farklı fiziksel bileşenler için enerji tüketimini detaylı bir şekilde modellemektedir. Cloudsched isteklerin yaşam döngüsünü modellemektedir ve yük dengeleme, enerji verimliliği ve kullanımı gibi farklı metrikler sağlamaktadır. Bahsi geçen açık kaynak kodlu benzetim araçlarının tamamı farklı bir katmana odaklanmaktadır, bütün bulut katmanlarını modelleyen bir araç henüz üretilmemiştir. Genel olarak benzetim süreci 4 bölüme ayrılabilmektedir [48]:

- Müşteri taleplerini oluşturmak
- Veri merkezini oluşturmak
- Yerleştirme politikasını belirlemek
- Sonuçları toplamak ve çıktı almak

Bu çalışmada benzetim ortamı olarak CloudSim aracı kullanılmaktadır. CloudSim Projesi bulut bilişim için geliştirilmiş Java tabanlı bir bulut benzetim ortamıdır. Melbourne Üniversitesi'nde bulunan CLOUDS Laboratuvarı'nda geliştirilmiştir. Geniş boyutlu veri merkezlerini (sunucu bilgisayarlarını, enerji farkında bilişim kaynaklarını) modelleyerek benzetimi yapılabilmektedir. Sanal makine yerleştirme probleminde kullanıcı tanımlı politikaları desteklemektedir [27].

3.6.1. CloudSim sistem modelinde enerji tasarrufu duyarlı dinamik sanal makine yerleştirmesi

CloudSim sistem modelinde dinamik sanal makine yerleştirmesi 4 parçaya ayrılmıştır:

1. Fiziksel makinenin ne zaman aşırı yüklü sayılacağını belirlemek (üzerindeki sanal makine yükünü azaltmak amacıyla),
2. Fiziksel makinenin ne zaman az yüklü sayılacağını belirlemek(uyku moduna geçirilmek amacıyla),
3. Aşırı yüklü fiziksel makine üzerindeki hangi sanal makinelerin taşınacağını belirlemek,
4. Az yüklü veya aşırı yüklü fiziksel makineler üzerinden seçilen sanal makinelerin yeni yerini belirlemektir.

Dinamik olmayan yük dengelemede sabit eşik değeri kullanılabilir. Sabit eşik dinamik ortamlar için iyi değildir. Anton ve arkadaşları [47] eski verileri kullanarak otomatik eşik belirleyen bir sistem geliştirmişlerdir. CPU kullanım sapması arttıkça CPU kullanımı yüzde yüze yaklaşmaktadır ve hizmet anlaşması ihlaline sebep olmaktadır. Sanal makine seçme sürecinde aşırı yüklü fiziksel makine üzerinden bir adet sanal makine seçildikten sonra o fiziksel makinenin hala aşırı yüklü olup olmadığına bakılmaktadır, eğer öyleyse tekrar bir sanal makine seçilmektedir. Minimum taşıma zaman politikasına göre taşınması en kısa süren sanal makine önce taşınmaktadır. Taşıma süresi, taşınacak olan VM'nin RAM'i fiziksel makinenin erişebildiği boş ağ bant genişliğine bölünerek tahmin edilmektedir.

Fiziksel makinelerin yük tespiti;

1. Önce aşırı yüklü hostlar bulunur,
2. Taşınacak sanal makineler hedef fiziksel makinelere yerleştirilir,
3. Diğer hostlara göre daha az yükü olan fiziksel makine bulunur ve bu fiziksel makine üzerindeki sanal makineler diğer fiziksel makinelere onları aşırı yüklü yapmadan taşımayı dener. Duruma göre hostu kapatır veya açık bırakır.
4. Bu işlem bütün hostlar için onların aşırı yüklü olduğu düşünülmeyene kadar devam eder.

3.6.2. CloudSim yazılımında bulunan bazı sınıflar

Java diliyle geliştirilmiş olan CloudSim oluşturulurken GridSim'den [40], GridSim [37] oluşturulurken de SimJava'dan yararlanılmıştır. CloudSim açık kaynak kodlu benzetim aracında "datacenter", "SANStorage", "virtualmachine", "cloudlet", "cloudcoordinator",

“bwprovisioner”, “memoryprovisioner”, “vmprovisioner”, “VMMAllocation” isimli sınıflar bulunmaktadır [24].

Datacenter sınıfı

Bu sınıf içerisinde donanım kaynaklarını barındırmaktadır. Depolama, bellek, işlemci kapasitesine göre donanımlar homojen ya da heterojen olabilmektedir [24]. İsim, fiziksel makine listesi (hostlist), işlemci gücü (MIPs), işlemci listesi (peList) gibi parametreleri almaktadır.

Datacenterbroker (Cloudbroker) sınıfı

Bu sınıf, kullanıcıların servis kalitesi (Quality of Service - QoS) gereksinimlerine göre kullanıcılar ve servis sağlayıcılar arasında aracılık eden nesnelerden oluşmaktadır [24].

DatacenterCharacteristics sınıfı

Bu sınıf veri merkezinde bulunan kaynakların özelliklerini tutmaktadır. Bunlar işlemci, bellek, depolama alanı, sanal makine yerleştirme politikası, bellek atama ve bant genişliği atama politikaları gibi özelliklerdir [24]. Kaynakların mimarisi, işletim sistemi, sanal makine yönetimi (Virtual machine management - VMM), fiziksel makine listesi, saat dilimi aldığı parametrelerden bazılarıdır.

SANStorage sınıfı

Veri merkezindeki geniş verilerin tutulduğu depo alan ağını modellemektedir [24].

Virtualmachine sınıfı

Bu sınıf, bir sanal makineyi modellemektedir. Bu sanal makinenin yaşam döngüsü, üzerinde bulunduğu fiziksel sunucunun sorumluluğundadır. Bir fiziksel sunucu birden fazla sanal makine içerebilmektedir ve işlemci çekirdekleri yer paylaşımlı ve zaman paylaşımlı olarak daha önceden belirlenen işlemci paylaşım politikalarına göre paylaştırılmaktadır. Her sanal makine kendisi ile ilgili bellek, işlemci ve depolama alanına

erişebilmektedir [24]. İşlemci sayısı, işlemci gücü, bellek, depolama alanı, bant genişliği vb. parametreleri alır [38].

Cloudlet sınıfı

Bu sınıf, bulut tabanlı uygulama servislerini modellemektedir [24]. Özelleştirilmiş görevleri ifade etmektedir. Boyut, dosya boyutu ve çıktı boyutu gibi parametrelere sahiptir [38]. Boyut, MI cinsinden işlemcinin işleyeceği komutun boyutunu ifade etmektedir. Dosya boyutu işlemciye giren verinin boyutunu, çıktı boyutu da işlemciden çıkan veri boyutunu göstermektedir. Hesaplama yoğunluklu, web sunucu ve veritabanı tiplerinde görevler oluşturulabilmektedir [27].

Cloudcoordinator sınıfı

Bu sınıf, veri merkezindeki kaynakların durumunu gözlemlemektedir ve yük azaltma kararlarını vermektedir [24].

BWProvisioner sınıfı

Bant genişliğini sanal makineler arasında paylaştırmaktadır [24].

MemoryProvisioner sınıfı

Sanal makinelere bellek atamaktadır [24].

VmProvisioner sınıfı

Sanal makineleri hostlara yerleştirmektedir [24].

VMMAllocationPolicy sınıfı

Bu sınıf sanal makineyi işlemci, bellek ve depolama durumuna göre uygun olan fiziksel sunucuya yerleştirmektedir [24].

CloudletSchedulerPolicy sınıfı

Sanal makinenin sahip olduğu işlemci gücünü üzerinde bulunan *cloudlet*ler arasında paylaştırır. Zaman paylaşımlı ve yer paylaşımlı olmak üzere iki türü bulunmaktadır [24].

VmSchedulerPolicy sınıfı

Hostlar tarafından uygulanır. Zaman paylaşımlı ve yer paylaşımlı gibi türleri bulunmaktadır. İşlemci gücünü sanal makineler arasında paylaştırır [24].

Host sınıfı

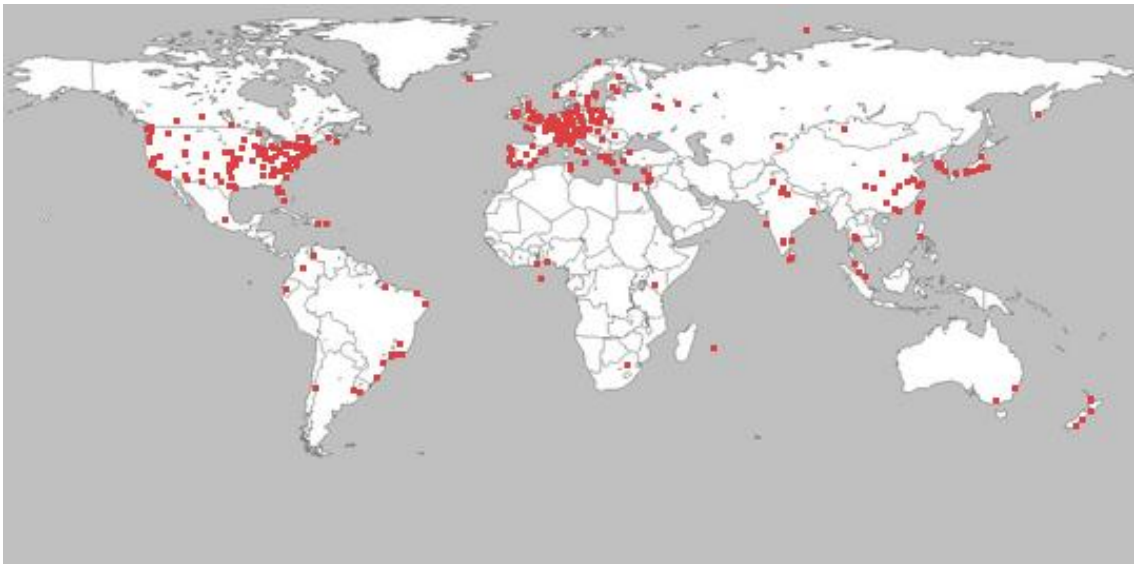
Fiziksel makineyi ifade etmektedir. Fiziksel makine numarası, işlemci çekirdeği sayısı, işlemci gücü, bellek, depolama alanı, bant genişliği, işlemci paylaşım stratejisi parametrelerini almaktadır [38].

3.7. Kullanılan Veriseti

Bu çalışmada gerçek dünya verisi olan PlanetLab [67] veriseti kullanılmıştır. PlanetLab, yeni ağ servislerinin geliştirilmesini sağlayan araştırma ağıdır. 2003'ün başından beri 1000'den fazla araştırmacı ve labaratuvar araştırma kuruluşu dağıtık depolama, ağ haritalama, uçtan uca sistemler, sorgu işleme vb. ile ilgili yeni teknolojiler geliştirmek için PlanetLab kullanmaktadır [47]. İsteyen araştırmacılar Şekil 3.10'da görüldüğü gibi dünyanın heryerinden PlanetLab uygulamasına kayıt olup bir veya birden fazla fiziksel makineye sahip olabilmektedir. Hizmet sağlayıcı kullanıcıların uygulamalarıyla ilgili bilgiye sahip değildir, iş yükleri heterojendir. Verimerkezinde iki tür fiziksel makine bulunmaktadır. Bunların yarısı 2 çekirdekli 1860 MHz intel Xeon 3040 işlemciye sahip HP Proliant G4 sunucudur ve diğer yarısı 2 çekirdekli 2660 MHz Intel Xeon 3075 işlemciye sahip HP Proliant ML 11065 sunucudur. G4 tipindeki sunucu 1860 MIPS değerinde, G5 tipindeki sunucu 2660 MIPS değerinde işlemciye sahiptir. Verisetinde bulunan VM'lerin toplam CPU miktarı verisetinde bulunan PM'lerin toplam CPU miktarının %12,31'dir [47]. Verisetindeki kayıtlar sunucuların 5 dakika aralıklarla CPU kullanım miktarlarınının kaydını içermektedir. Bu kayıtlar CoMon projesi aracılığıyla elde edilmektedir. CoMon PlanetLab veri merkezinde bulunan sunucuların sağlıklı olup olmadığını; CPU, RAM, bw gibi kaynak tüketimini ve sunucuların zamana göre davranışlarını takip eden yazılımdır.

PlanetLab ortamında bir hesabın sahip olduğu sunucuların bulunduğu alana *slice* denir. Bu alan içinde bir veya birden fazla fiziksel sunucu bulunabilmektedir. Fiziksel sunucuların barındırdığı sanal makinelere ise *sliver* denir. Eğer bir sanal sunucu %0.1 oranında CPU kullanımına sahipse buna canlı *sliver* denir. Bir *slice* en az bir canlı *sliver* içeriyorsa canlı *slice* denir. CoMon'a cevap veren fiziksel makinelere ise canlı düğüm denir.

Planetlab veriseti CloudSim projesinde kullanılmaktadır. PlanetLab veriseti 1052 adet sanal makine ve 800 adet fiziksel makineden oluşmaktadır.



Şekil 3.10. PlanetLab'ın sahip olduğu 717 bölgedeki 1353 düğüm [47]

4. SANAL MAKİNE YERLEŞTİRME PROBLEMİ

Bulut bilişim, depolama, hesaplama, ağ gibi bilişim kaynaklarına her an her yerden ağ üzerinden erişim sağlayan bir modeldir. Günümüzde yaygın olarak kullanılan bulut bilişimin temelinde sanallaştırma teknolojisi yatmaktadır. Sanallaştırma teknolojisi ile bir fiziksel makine üzerinde birden fazla sanal makine yürütülerek kaynakların ortak kullanımı sağlanmaktadır. Kaynakların kullanıma sunumunda kaynakların aşırı kullanımı engellenirken, diğer taraftan verimli kullanımı sağlanmalıdır. Kaynakların aşırı kullanımı hizmet kesintisi, performans kaybı gibi sorunlara sebep olmaktadır. Kaynakların verimsiz kullanımı ise maliyetlerin artmasına yol açmaktadır. Sistemdeki toplam yüke bağlı olarak kaynak tahsisinde bu iki unsur arasındaki dengenin sürekli olarak gözetilmesi gerekmektedir. Bulut bilişim kapsamındaki veri merkezlerinde yük dengesini sağlamak üzere sanal makineler fiziksel makineler üzerinde dinamik olarak taşınmaktadır. Hangi sanal makinenin hangi fiziksel makine üzerinde çalışacağı sorusu sanal makine yerleştirme problemi olarak tanımlanmaktadır. Bu kapsamda, problemin çok amaçlı optimizasyon yöntemleriyle çözümleri araştırılmıştır.

4.1. Sanal Makine Yerleştirme Probleminin Kısıtları

Yönetim, kaynak ve performans gibi unsurlara bağlı kısıtlar dikkate alınarak sanal makinelerin fiziksel makineler üzerine yerleştirilmesi karmaşık bir problemdir [59]. Yönetime bağlı kısıtlar bir sanal sunucunun belirli bir fiziksel makinede tutulması gerekliliği veya belirli iki sanal sunucunun farklı fiziksel makinelerde bulunması gerekliliği olabilmektedir.

Kaynağa bağlı kısıtlar sanal makinelerin ihtiyaç duyduğu disk, RAM, CPU ve ağ bant genişliği gereksinimlerinin karşılanmasıdır. Bir fiziksel makinenin kaynak kapasitesinin, üzerinde bulunan sanal makinelerin toplam kapasitesini geçmemesi gerekmektedir. Burada kaynak türlerinden birisinin yetersiz kalması diğer kaynak türlerinin atıl kalmasına sebep olabilmektedir. Örneğin eğer bir fiziksel makine üzerinde bulunan 10 adet sanal makinenin RAM kapasitesini ancak karşılayabiliyorsa üzerinde boş CPU miktarı bulunsa bile bu CPU kullanılamamaktadır. Fiziksel makinelerin üzerinde bulunan sanal makinelerin kaynak ihtiyacını karşılaması gerekliliğinin yanında yük dengelemenin de sağlanması gerekmektedir. Eğer sanal makine yükleri belirli makinelere toplanır ve bazı makineler az

yükle kalırsa aşırı yük altında ezilen fiziksel makineler beklenen performansı sağlayamayabilirler hatta hizmet kesintisi durumu yaşanabilmektedir. Bunu çözmek için sanal makinelerin fiziksel makineler arasında dengeli dağıtılmalarını sağlamak gerekmektedir.

Enerji tasarrufu sanal makine yerleştirme probleminde dikkat edilmesi gereken etkenlerden birisidir. Sanal makineler performansı korumak kaydıyla bazı fiziksel makineler üzerine toplanıp, bazı fiziksel makineler boşaltılabimektedir ve boş kalan bu makineler kapatılarak enerji tasarrufu sağlanabilmektedir. Bütün bunların yanında makinelerin kullanım durumlarının dinamik olması sanal makine yerleştirme problemini zorlaştırmaktadır.

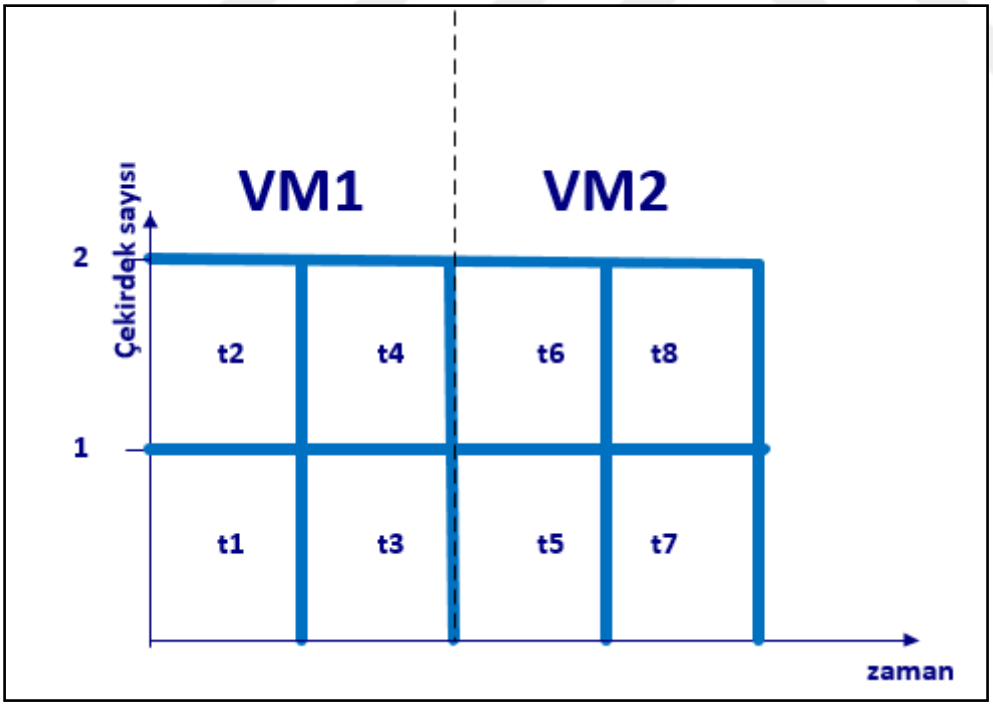
4.2. Sanal Ortamlarda İşlemci Kullanımı

Bir sanal makinenin işlemci gücü görevler arasında paylaştırılmaktadır. İşlemci gücü paylaşımında kullanılan iki tür politika vardır. Bunlar yer paylaşımlı ve zaman paylaşımlı politikalar olarak adlandırılmaktadır. Bir sanal makinenin çalıştırması gereken görev varsa bunları sırayla çalıştırırsa buna yer paylaşımlı komut programlama denir. Eğer bu komutlar eşzamanlı olarak iki işlemci üzerinde dönüşümlü olarak çalıştırılırsa buna zaman paylaşımlı komut programlama denilmektedir. Birinci kaynak sağlama politikası olan yer paylaşımlı yöntemde sadece bir sanal makine belirlenen çekirdeği, başlanan görev bitene kadar kullanabilmektedir. Örneğin bir fiziksel makine üzerinde bir çekirdek varsa ve iki sanal makine varsa bu sanal makinelerden biri çekirdeği kullandıktan sonra diğeri kullanmaktadır. Yani sırayla birer birer kullanılmaktadır. Aynı senaryoda zaman paylaşımlı programda ise belirlenen zaman diliminde çekirdeği bir süre bir sanal makine, bir süre diğer sanal makine kullanmaktadır. Bir görev işlemciyi kullanmaya başladıktan sonra belirli bir süre kullanabilmektedir, belirlenen süre sonunda görev tamamlanmamış olsa bile işlemci diğer görevin kullanımı için bırakılmaktadır. Yani sanal makineler arasında işlemci çekirdeği dönüşümlü olarak kullanılmaktadır [40].

Bir fiziksel makine üzerindeki sanal makinelere ulaştırılan kaynak miktarı fiziksel makinenin toplam işlemci gücüyle sınırlıdır. Bu kritik etken, yerleştirme sürecinde göz önüne alınmalıdır. Fiziksel makine seviyesinde, her bir fiziksel makinedeki her bir çekirdeğin işlemci gücünün ne kadarının her bir sanal makineye nasıl atanacağı belirlenmelidir. Sanal makine seviyesinde ise sanal makinenin elinde bulunan işlemci

gücünün kendi görevlerine nasıl atanacağı belirlenmelidir. Sanallaştırma ortamlarının kullandığı yer paylaşımlı ve zaman paylaşımlı politikalar arasındaki farkı ve uygulama performansı üzerindeki etkilerini görselleştirmek için Şekil 4.1, 4.2, 4.3, 4.4 incelenebilir. Bu şekillerde iki adet CPU'su bulunan bir fiziksel makine üzerinde iki adet sanal makine çalışması betimlenmektedir. Bu sanal makineler VM1 ve VM2 ile ifade edilmektedir. Bu sanal makinelerin her biri iki çekirdeğe ihtiyaç duymaktadır ve VM1'in t1, t2, t3, t4 görevlerini; VM2'nin ise t5, t6, t7, t8 görevlerini çalıştırması gerekmektedir.

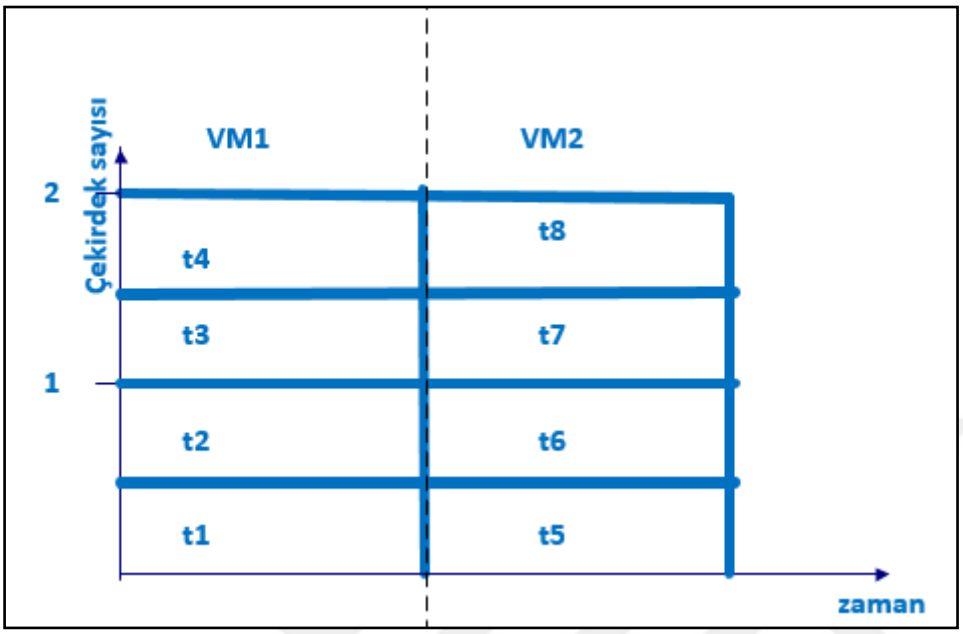
Şekil 4.1'de hem sanal makine hem de görevler için yer paylaşımlı provisyon politikası kullanılmaktadır. Her bir sanal makine iki çekirdeğe ihtiyaç duyduğu için bir zaman diliminde sadece bir sanal makine çalışabilmektedir. Bu sebepten VM2, VM1 görevlerini bitirdikten sonra çalışabilmektedir. Aynı durum sanal makineler üzerinde bulunan görevler için de geçerlidir. Her bir görev sadece bir çekirdeğe ihtiyaç duyduğundan iki görev eşzamanlı olarak çalışabilmektedir. İki görev (t1, t2) çalıştırıldıktan sonra diğer iki görev (t3, t4) çalıştırılmaktadır.



Şekil 4.1. VM'ler ve görevler için yer paylaşımlı CPU kullanımı [40]

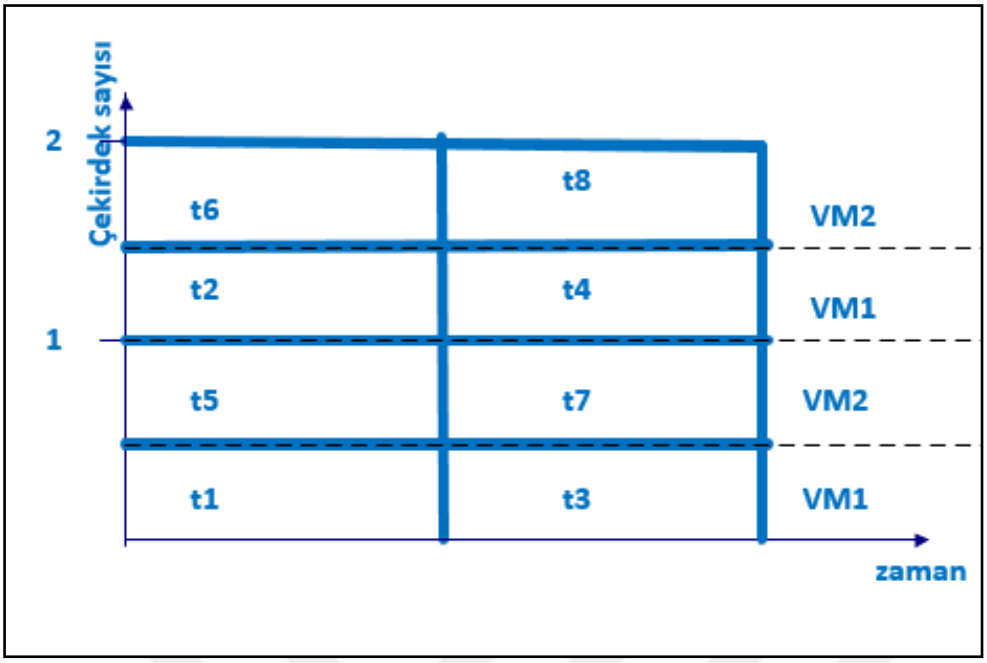
Şekil 4.2'de sanal makineleri yerleştirmek için yer paylaşımlı politika, sanal makine üzerindeki görevleri yerleştirmek için zaman paylaşımlı politika kullanılmaktadır. Bir

çekirdek belirli zamanda belirli sanal makine tarafından kullanılmaktadır. Sanal makinenin sahip olduğu çekirdek ise aynı anda birden fazla görev için kullanılmaktadır.



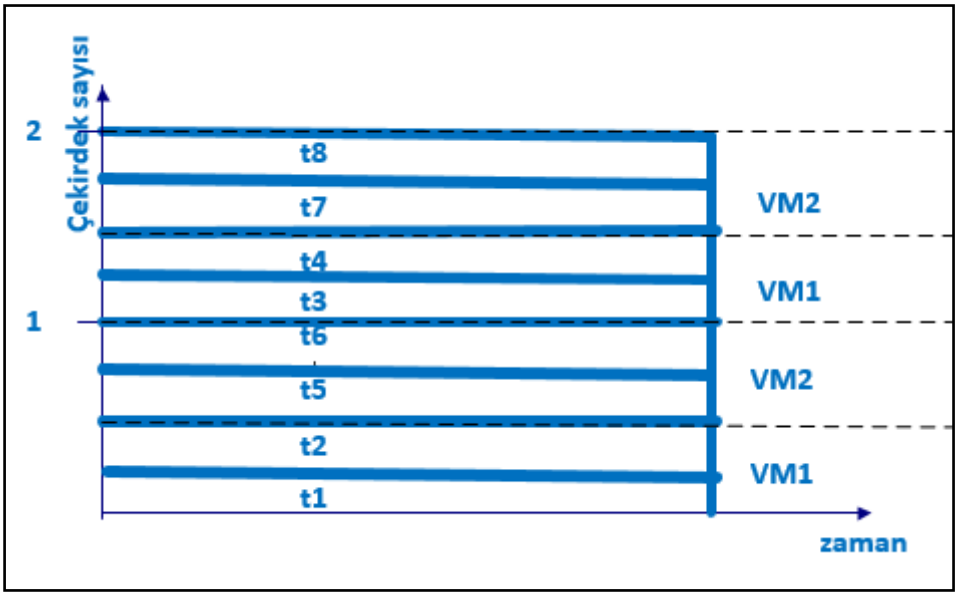
Şekil 4.2. VM'ler için yer ve görevler için zaman paylaşımlı CPU kulanımı [40]

Şekil 4.3'de sanal makineleri yerleştirmek için zaman paylaşımlı politika, sanal makine üzerindeki görevleri yerleştirmek için yer paylaşımlı politika kullanılmaktadır. Her bir sanal makine belirli zaman diliminde çekirdeği kullanmaktadır. Çekirdek aynı anda diğer sanal makine tarafından da kullanıldığından daha önce belirtilen senaryolarda kullanılan çekirdek gücüne göre birim zamanda kullanılan çekirdek gücü daha azdır. *Görev* atamaları yer paylaşımlı olduğundan bir sanal makine her bir çekirdeğe sadece bir görev atayabilmektedir, diğer görevler daha sonra yapılmak üzere sıraya sokulur.



Şekil 4.3. VM'ler için zaman paylaşımlı, görevler için yer paylaşımlı CPU kulanımı [40]

Şekil 4.4'de, bir çekirdek aynı anda iki sanal makine tarafından kullanıldığından zaman paylaşımlı provizyon yöntemi kullanılmaktadır. Bir sanal makine kendisine atanmış olan çekirdeği bir zaman diliminde birden fazla göreve kullandığından zaman paylaşımlı provizyon yöntemini kullanılmıştır [24].



Şekil 4.4. VM'ler ve görevler için zaman paylaşımlı CPU kullanımı [40]

4.3. Hizmet Seviyesi anlaşması ve Enerji Tüketimi

Tüketiciler geleneksel yöntemler yerine bulut bilişimi kullanmaya başladıkça tüketiciler ve sağlayıcılar arasındaki anlaşmalar, yani hizmet seviyesi anlaşmasının - Service Level Agreements (SLA) önemi artmaya başlamıştır. Servis sağlayıcı ve müşteriler arasında varılan anlaşmalarda servis kalitesinin sağlanması zorunludur. QoS, cevap zamanı ve çıktı (throughput) gibi özelliklere sahiptir. Çıktı berli zaman aralığında iletilen verinin miktarıdır [60]. Bu gereksinimleri sağlamadığı zaman hizmet sağlayıcı müşterisine ceza ödemek zorunda kalabilmektedir. Uygulamanın kaynak gereksinimleri karşılanmazsa uygulama cevap zamanı gecikmesi, hizmet kesintisi gibi sorunlarla kaşılaşılabilmektedir. Servis sağlayıcı hizmet anlaşması gereksinimlerini karşılarken enerji tüketimini da gözardı etmemelidir. Enerji tüketimi işlemci, disk, güç kaynağı, soğutma sistemlerinin harcadığı enerjiyle alakalıdır.

4.4. Performans Metrikleri

Bulut ortamları için geliştirilen benzetim yazılımları test edilen algoritmaların performansını değerlendirmek için kullanılan ölçütlere performans metrikleri denilmektedir. Bu bölümde bazı performans metriklerinin kısa açıklamaları yapılmaktadır.

Yük değişimi ve kullanımının standart sapması

Bu iki metriğin her ikisi de kaynakların ortalama kullanımından sapmayı gösterir. Değerlendirmesi kolay olduğu için yaygın bir şekilde kullanılmaktadır [38]. Fakat kaynak kullanımından çok zaman kısıtına odaklanan algoritmalar için uygun değildir.

Tamamlanma süresi

Tamamlanma süresi, bir işin başlangıcından bitirilişine kadar geçen süredir. Yük dengesi sağlandığı zaman tamamlanma süresi de düşer ve daha düşük tamamlanma süresi bir planlama algoritmasının birincil amacıdır. Bu metrik zaman kısıtının önemsendiği algoritmalarda önemlidir.

Aşırı yüklü fiziksel sunucu sayısı

Bulutta ne kadar aşırı yüklü fiziksel sunucu olduğunu ölçmektedir ve sistem durumu hakkında genel bilgi edinmeyi sağlamaktadır. Yük dengeleme algoritmaları aşırı yüklü fiziksel sunucuları azaltmayı amaçlamaktadır. Bu metrik yük dengelemeyle ilgili net bir görüş sağlamaktadır. Fakat bu metrik yükün dağıtımıyla ilgili çok ayrıntılı bilgi vermemektedir.

Çıktı

Dengesiz dağılmış yük, sistemin performansını düşürebileceğinden bu metrik fiziksel sunucuların istekleri ne kadar hızlı işleyeceğini değerlendirmektedir. Bu yüzden yüksek değerdeki çıktı, yükü daha iyi dengelenmiş sistem demektir. Cevaplama zamanının önemsendiği durumlar için kullanımı tavsiye edilebilmektedir. Bu metrik genelde taşıma sayısı gibi metriklerle birlikte değerlendirilmektedir.

Bağlantıların standart sapması

Bu metrik network hassasiyeti olan algoritmalarda kullanılabilmektedir. Bağlantıların dengeli olup olmadığını denetlemektedir.

Ortalama dengesizlik seviyesi

Popüler bir metrik olan kaynak kullanımının standart sapması sadece bir kaynağın kullanımını hesaplarken; bu metrik CPU, RAM, bant genişliği gibi birden fazla çeşitte kaynağı bir arada göz önüne almaktadır.

Sanal makine taşıma sayısı

Yardımcı metrik olan taşıma sayısı, yük dengelemeyi değerlendirmek için tek başına kullanılabilecek bir metrik değildir. Çünkü taşıma sayısı artırılarak yük dengesi daha iyi sağlanabilmektedir. Fakat sanal makinelerin taşınması sisteme ilave yük getirmekte, bu da performansı düşürmektedir. Performans ve yük dengesi arasında bir ödünleşim bulunmaktadır.

Hizmet seviyesi anlaşması İhlali

Yardımcı metrik olan Hizmet seviyesi anlaşması ihlali tek başına kullanılabilecek bir metrik değildir. Hizmet seviyesi anlaşması ihlali sanal makinenin yeterli kaynağı alamamış olmasıdır. Bu metriğin değerinin fazla olması fiziksel sunucuların iyi dengelenmemiş olduğunu gösterir; düşük olması iyidir. Diğer metriklerle bir arada değerlendirilmelidir [38].

4.5. Performans Metriklerinin Hesaplanması

Bu tez çalışmasında kullanılan benzetim yazılımı olan CloudSim perforformans metriklerinden bazılarının matematiksel hesapları ayrıntılı olarak anlatılmaktadır. Bu metrikler [47]'de yapılan çalışmadan yararlanılarak detaylı olarak açıklanmıştır.

4.5.1. Hizmet seviyesi anlaşması ihlali

SLAV hesabının belirlenmesinde etkili olan iki değer vardır. Bunlar aktif fiziksel makine başına düşen ortalama SLAV zamanı (SLAV Time per Active Host - SLATAH) ve Taşımadan kaynaklanan performans düşüşü (Performance Degreetion per Migration - PDM) değerleridir. Diğer bir metrik ise sanal makine yerleştirme adaptasyonu süresince yapılan VM taşınma sayısıdır. Hizmet seviyesi anlaşması ihlali olan SLAV değeri SLATAH (%) ve PDM (%) değerlerinin çarpılmasıyla elde edilmektedir. SLATAH, aktif PM başına düşen SLAV değerini; PDM ise VM taşınmasından kaynaklanan performans düşüşünü ifade etmektedir. Bulut bilişim ortamaları için QoS gereksinimlerini karşılamak çok önemlidir. QoS gereksinimleri SLA formunda formülleştirilmektedir. Minimum çıktı ve maksimum cevap zamanı gibi açılardan belirlenebilmektedir. Bu özellikler uygulamadan uygulamaya değiştiğinden hizmet kalitesini değerlendirirken kullanılmamaktadır. İş yükünden bağımsız bir metriği, IaaS'deki herhangi bir sanal makineye dağıtılan SLAV'ı değerlendirmek için tanımlamak gerekmektedir [47].

PDM ve SLATAH'ın her ikisi de aynı öneme sahip olduğundan yazarlar SLAV isimli bir metrik sunmuşlardır. Bu metrik hem fiziksel makinenin aşırı yüklenmenin sebep olduğu performans düşüşünü hem de sanal makine taşımasının sebep olduğu performans düşüşünü

göstermektedir. SLAV, SLATAH ve PDM'nin birleşimini ifade etmektedir ve aralarındaki ilişki Eş. 4.1'de görülmektedir.

$$SLAV = SLATAH.PDM \tag{4.1}$$

4.5.2. Sanal makine taşınmasından kaynaklanan performans düşüşü

Anton B.'nin modellediği çok çekirlekli CPU mimarisine göre n çekirdek varsa, her bir çekirdek m MIPS içermektedir, yani çekirdek 1 saniyede m milyon komut işleyebilmektedir. Bir tek CPU'nun toplam kapasitesi $n \times m$ MIPS'dir. Canlı taşıma durdurmasız ve çok kısa bir kesinti ile yapılabilmektedir. Fakat yine de taşıma sırasında uygulama performansına olumsuz etki yapmaktadır. Bir VM'in taşınma uzunluğu o VM'in kullandığı RAM ve erişilebilir ağ bant genişliğine bağlıdır. Canlı taşımayı sağlamak için ayarlamalar ona göre yapılmaktadır. VM'in imajı ve verisi ağ erişimli depolama biriminde tutulmaktadır, böylece sanal makinenin depolama birimlerini kopyalamak ve taşımak gerekmemektedir. Yazarların deneyimlerine göre bir j sanal makinesinin taşıma zamanı Eş. 4.2 ve performans azalması Eş. 4.3'de tanımlanmaktadır.

$$T_{m_j} = \frac{M_j(vm\ j'nin\ kullandığı\ ram)}{B_j(ulaşılabilir\ ağ\ bant\ genişliği)} \tag{4.2}$$

$$U_{d_j} = 0.1. \int_{t_0}^{t_0+T_{m_j}} U_j(t)dt \tag{4.3}$$

U_{d_j}, sanal makine j'nin sebep olduğu toplam performans düşüşünü, T_{m_j}, taşımanın ne kadar sürdüğünü, $U_j(t)$, CPU kullanımını, t_0, taşıma başlangıç zamanını ifade etmektedir [47].

PDM, Sanal makine taşımasının sebep olduğu performans düşüşüdür. Eşitliği 4.4'de gösterilmektedir.

$$PDM = \frac{1}{M}\sum_{j=1}^{M}\frac{C_{dj}}{C_{rj}} \tag{4.4}$$

C_{dj},göç kaynaklı performans düşüşünü, C_{rj} sanal makinenin talep ettiği CPU kapasitesini ifade etmektedir [47].

4.5.3. Aktif fiziksel makine başına düşen SLAV zamanı

Aktif fiziksel makine başına düşen SLAV zamanı kısaca SLATAH olarak ifade edilmektedir. Aktif fiziksel makinelerin ne kadar süre boyunca CPU kullanımlarının %100 olduğunun ortalması alınarak hesaplanır. Bir uygulamanın bulunduğu fiziksel makinenin CPU kullanımı %100 ise o uygulamanın perofmansı o fiziksel makinenin kapasitesiyle sınırlı olduğundan gerekli performans seviyesi sağlanamamaktadır.

Yazarların önerdiği metriklerden SLATAH (SLAV Time per Active Host) Hangi aktif fiziksel makinenin CPU kullanımının %100 olduğunun ne kadar sürdüğünün yüzdesi bulunarak hesaplanmaktadır. SLATAH mantığı şudur ki bir uygulamanın bulunduğu fiziksel makinenin CPU kullanımı %100 ise o ugulamanın performansı o fiziksel makinenin kapasitesiyle sınırlıdır ve gerekli performans seviyesi sağlanamamaktadır.

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{si}}{T_{ai}} \tag{4.5}$$

T_{si}, i fiziksel makinesinin CPU'sunun %100 kullanıldığı zaman; T_{ai} i fiziksel makinesinin aktif olarak çalıştığı süreyi ifade etmek üzere eşitliği 4.5'de gösterildiği şekildedir [47].

4.5.4. Enerji hesabı

Veri merkezleri çok büyük miktarda enerji tüketmektedir. Örneğin Google Veri Merkezi, nüfusu 2010 yılında 805,193 olan San Fransisco şehrinde tüketilen enerji kadar enerji tüketmektedir [24, 62]. Bu yüzden veri merkezlerindeki enerji israfını önlemek çok önemlidir. Veri merkezlerinde fiziksel makineler belirli sıcaklık aralığında çalıştığından enerji sadece fiziksel makinelerin çalışmasında değil aynı zamanda bu makinelerin oluşturduğu sıcak ortamı soğutmak için de kulanılmaktadır. Ne kadar az fiziksel makine çalışırsa o kadar az soğutma masrafı ortaya çıkacaktır. Güç ve enerji birbiriyle ilişkili kavramlardır. Gücün eşitliği Eş. 4.6'da verildiği gibidir. Enerji ise Eş. 4.7'de görüldüğü gibi güç değeri kullanılarak hesaplanmaktadır.

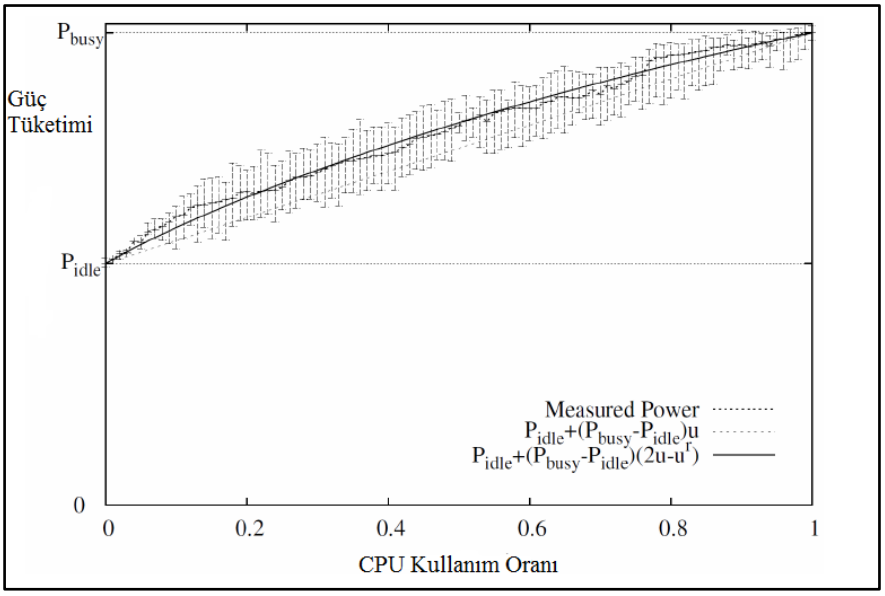
$$P = W/T \tag{4.6}$$

$$E = P.T \tag{4.7}$$

Eş. 4.6 ve Eş. 4.7'de görülen P, güç; T, zaman periyodunu; E enerjiyi ifade etmektedir. Güç ve enerji faklı şeylerdir. Güç tüketimini azaltmak her zaman enerji tüketimini azaltmamaktadır. Güç tüketimi CPU performansı azaltılarak azaltılabilir, örneğin frekansı düşürülebilir fakat programın çalışma süresi uzayacağından iş bitiminde harcanan enerji miktarı aynıdır.

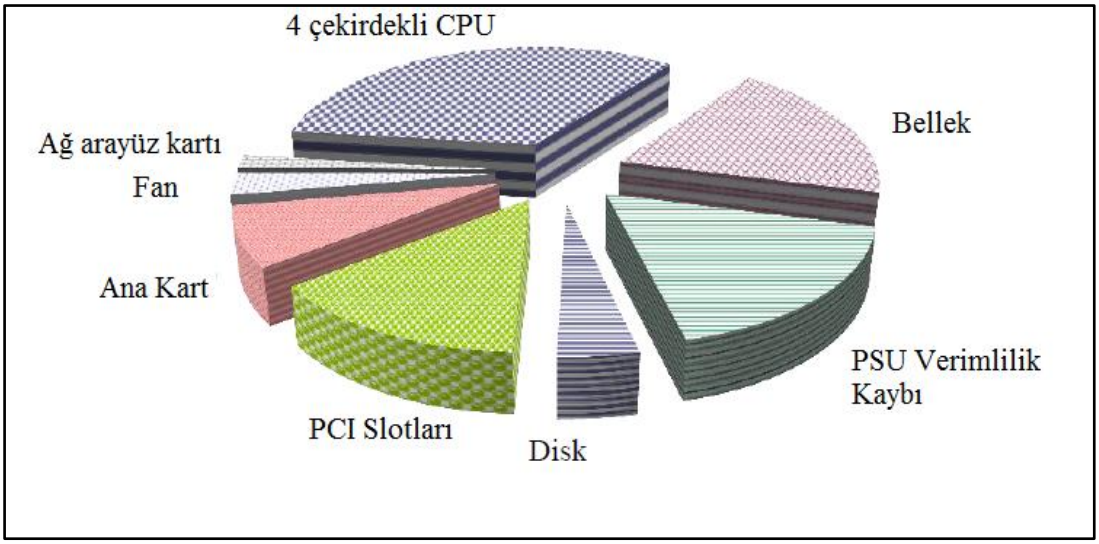
Bir sunucunun harcadığı enerji miktarı kullanılan CPU miktarıyla yakından ilişkilidir [61]. Fiziksel makine üzerinde hiç iş yükü olmadığında kullanılan güç P_{idle} ile, makine CPU'su %100 kullanıldığı durumda güç tüketimi P_{busy} ile ifade edilmektedir. Bir sunucunun kullandığı güç miktarı Eş. 4.8'de hesaplanmaktadır. u, kullanılan CPU oranını ifade etmektedir. Şekil 4.5'de gerçekte tüketilen güç ile hesaplanan tahmini güç tüketiminin değerleri görülmektedir.

$$P(u) = P_{idle} + \left(P_{busy} - P_{idle}\right) * u \tag{4.8}$$



Şekil 4.5. Bir sunucunun CPU kullanımı ve güç tüketimi arasındaki ilişki [47]

Şekil 4.6'da görüldüğü üzere sistemde bulunan diğer bileşenler (örn. I/O, bellek) de güç tüketmektedir fakat bu bileşenlerin güç tüketimi de CPU kullanımıyla orantılıdır.



Şekil 4.6. Sunucu bileşenleri tarafından tüketilen güç miktarları [47]

PlanetLab verisetinde bulunan sunucuların CPU kullanım oranlarına göre Watt cinsinden güç tüketimleri SPECpower benchmark'dan alınan değerler Çizelge 4.1'de görülmektedir [47].

Çizelge 4.1. Sunucuların CPU kullanım oranlarına göre Watt cinsinden güç tüketim miktarları [47]

Sunucu - CPU Kullanım oranı	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP-ProLiant-G4	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP-ProLiant-G5	93.7	97	101	105	110	116	121	125	129	133	135

4.5.5. Aktif fiziksel makine başına düşen ortalama CPU kullanım oranı hesaplama

Bir fiziksel makinenin kaynak kapasitesi üzerinde barındırdığı sanal makinelere dağıtılmaktadır. Fiziksel makine üzerindeki kaynağın kullanılmayan kısmı atıl duran kaynaktır. Bir kaynak ne kadar yüksek oranda kullanılırsa o kadar verimli kullanılıyordur. Diğer taraftan bir fiziksel makine üzerinde taşıdığı sanal makinelerin bütün kaynak ihtiyaçlarını karşılıyor olması gerekmektedir. Örneğin bir fiziksel makinede sanal makineye yetecek miktarda RAM bulunmazsa bu makine üzerinde CPU miktarı çok fazla olsa bile sanal makinenin başka bir makineye taşınması gerekmektedir. Kaynak kullanımında performansı ve enerji tüketimini etkileyen en önemli kaynak olan CPU'nun kullanım miktarları ölçülmüştür. Aktif fiziksel makine başına düşen CPU kullanım oranı

hesaplanırken, aktif olan fiziksel makinelerin CPU kullanım oranlarının ortalaması alınmaktadır.

4.5.6. Benzetim Ortamında Maliyet Hesabı

Bulut ortamlarında harcanan enerjiyi minimuma indirmek doğanın korunması açısından ve bulut sağlayıcının maliyetini azaltmak açısından çok önemlidir. Bu bölümde bulut bilişim benzetim araçlarından birisi olan CloudSim temel alınarak maliyet hesabı anlatılmaktadır.

Tekli sanal makine taşıma probleminde maliyet hesabı

Bir fiziksel makine ve birden fazla sanal makinenin olduğu bir sistemde maliyet hesabı probleminde zaman N parçaya ayrılmaktadır. Her bir zaman diliminin 1 saniye olduğu kabul edilmektedir. Kaynak sağlayıcı kuruluş, fiziksel makinenin harcadığı enerji maliyetini karşılamaktadır.

$$C_T = C_p . t_p \tag{4.9}$$

Burada t_p, zaman periyodu ve C_p, 1 dilimlik zamanda harcanan enerji olmak üzere toplam maliyet Eş. 4.9'da hesaplanmaktadır.

Kaynak kullanımı CPU performansı ile ifade edilmektedir. Talep edilen CPU miktarı erişilebilir kapasiteyi aşarsa SLAV oluşmaktadır. Bu durumda müşteriye kaynak sağlayıcı tarafından ödeme yapılmaktadır. Bunun maliyeti ise $C_v t_v$ eşitliği ile hesaplanmaktadır. Burada C_v birim zamana düşen SLAV maliyeti ve t_v SLAV'nın sürdüğü zaman miktarıdır. Anton B. ve arkadaşları bu birimleri Eş. 4.10'de görüldüğü şekilde tanımlamışlardır.

$$C_p = 1,\ \ C_v = s \quad \text{veya} \quad C_p = \frac{1}{s},\ \ C_v = 1 \tag{4.10}$$

$$r = n - v \tag{4.11}$$

Eş. 4.11'de n, SLAV'nın bitme anı; v, SLAV'nın başlama anı, r SLAV devam etme süresi, m sanal makinenin başka bir fiziksel makineye taşınmaya başladığı andır. Sanal makine taşınması T kadar sürmektedir. Bir sanal makineyi başka bir fiziksel makineye

taşımak için bir fiziksel makineye daha ihtiyaç bulunduğundan taşıma sırasında harcanan enerji $2C_pT$'dir. Sanal makinenin ne zaman taşınması gerektiği m zamanını belirlerken şunlar göz önünde bulundurulmaktadır:

1. Harcanan toplam enerji maliyetini azaltmak
2. SLAV nın sebep olduğu maliyeti azaltmak

Maliyet Fonksiyonu

Maliyet problemini analiz etmek için 4.12 numaralı maliyet fonksiyonu tanımlanmıştır. Toplam maliyet, SLAV'ın sebep olduğu maliyeti ve fazladan enerji tüketim maliyetini de içermektedir. Maliyet fonksiyonunda SLAV başlayana kadar harcanan enerjinin maliyeti hesaplanmaz çünkü algoritmalar bu andan sonra devreye girer ve algoritmalar arasında karşılaştırma yapmak için bu enerjiye ihtiyaç duyulmamaktadır. Enerji tüketimi sanal makinenin bulunduğu ve taşınacağı fiziksel makinenin tükettiği enerji ve SLAV başladıktan sonraki ilk bulunduğu fiziksel makinenin harcadığı enerji Eş. 4.12'de $C(v,m)$ ile hesaplanmaktadır. Maliyet fonksiyonu hesaplarında m taşıma başlama anını; T taşıma süresini simgelemektedir.

$C(v,m)$: (4.12)

$= (v-m)C_p$ ***eğer*** $m < v \; ve \; v - m \geq T$ C_1

$= (v-m)C_p + 2(m-v+T)C_p + (m-v+T).C_v$ ***eğer*** $m \leq v, v - m < T$ C_2

$= rC_p + (r-m+v)C_p + rC_V$ ***eğer*** $m > v$ C_3

Eş. 4.12, şartlara göre üç bölüme ayrılmaktadır. Birinci bölüm C_1, ikinci bölüm C_2 ve üçüncü bölüm C_3 ile ifade edilmektedir.

Birinci durum (C_1)

Taşıma, SLAV başlamadan başlamaktadır ve SLAV başlamadan ya da başlayacağı anda bitmiş ise maliyet C_1 fonksiyonu ile hesaplanmaktadır. Burada SLAV başlamadan yapılan

sanal makine taşıma maliyeti hesaplanmaktadır. SLAV maliyeti yoktur. $(v - m)$ taşıma süresini ifade etmektedir.

İkinci durum (C_2)

Taşıma, SLAV başladığı anda ya da başlamadan önce başlamıştır fakat taşıma bitmeden önce SLAV başladığından bir miktar SLAV maliyeti bulunmaktadır. Ana fiziksel makinenin v anına kadar harcadığı enerji maliyeti, SLAV süresince 2 fiziksel makinenin harcadığı enerji maliyeti ve SLAV ceza maliyeti toplamı ile hesaplanmaktadır.

Üçüncü Durum (C_3)

Sanal makine taşınması SLAV başladıktan sonra başlamıştır. Ana fiziksel makinenin harcadığı enerji maliyeti, ikinci fiziksel makinenin T süresince harcadığı enerji ve ödenen SLAV cezası toplanarak hesaplanmaktadır [47].

Optimal çevrimdışı algoritma maliyet hesabı

Algoritmanın kalitesi m ve v zamanları arasındaki ilişkiye bağlıdır. Yani taşıma ve SLAV başlaması arasında ne kadar az süre geçtiyse algoritma o kadar kalitelidir. Eğer SLAV başlamadan taşıma başlarsa bu en iyi durumdur. Eş. 4.12'deki değerler Eş. 4.13'deki değerlerle değiştirilmesi durumu aşağıda incelenmektedir.

$$v - m = aT,\ m = v - aT,\ a = (v - m)/T \tag{4.13}$$

Maliyet fonksiyonunda Eş. 4.12'de tanımlanan 3 durum aşağıda açıklanmaktadır.

Birinci Durum

$m < v,\ v - m \geq T\ ,\ aT \geq T$'dir ve $a \geq 1$ 'dir. Tekli sanal makine taşıma probleminde Eş. 4.12'de bulunan C_1 fonksiyonunda $(v - m)C_p$'de m'nin yerine $v - aT$ yazıldığında C_1=$(v - v + aT)C_p = aTC_P$ bulunmaktadır.

İkinci Durum

$m \leq v,\ v - m < T, a \geq 0$ ve $aT < T$ yani $0 \leq a < 1$'dir. Yine tekli sanal makine taşıma problemindeki Eş. 4.12'de $C_2 = (v-m)C_p + 2(m-v+T)C_p + (m-v+T)C_v$ m'nin yerine $v - aT$ yazıldığında $C_2 = aTC_p + 2T(1-a)C_p + T(1-a)C_v$ bulunur.

Üçüncü Durum

$m > v$ bundan dolayı $a < 0,\ a = \frac{v-m}{T}, m = v - aT$ yine tekli sanal makine taşıma probleminde Eş. 4.12'de $C_3 = rC_p + (r-m+v)C_p + rC_v = C_p(2r-m+v) + rC_v$ fonksiyonunda $r = m - v + T$ olduğundan m'nin yerine $v - aT$ ve r'nin yerine de $-aT + T$ yazılırsa $C_3 = C_p(2T - 2aT - v + aT + v) + TC_v - aTC_v$ bulunmaktadır. Sadeleştirildiği zaman $C_3 = T(2-a)C_p + T(1-a)C_v$ bulunur, bu da $C_2(v,a)'$ya eşittir $C_2(v,a)$, $C_3(v,a)$'ya eşit olduğu için eşitlik sadeleştirilerek Eş. 4.14 elde edilmektedir.

$C(a) =$ (4.14)

$T(2-a)C_p + T(1-a)C_v$ ***eğer*** $a < 1$ ise

aTC_p ***eğer*** $a \geq 1$ ise

Problem tanımında $C_p = 1/s$ ve $C_v = 1$ olarak tanımlanmıştı, yukardaki $C(a)$ denkleminde yerine koyulduğunda Eş. 4.15 elde edilmektedir

$C(a) =$ (4.15)

$T(2-a)/s + T(1-a)$ ***eğer*** $a < 1$ ise

aT/s ***eğer*** $a \geq 1$ ise

$a = 1$ olduğunda maliyet minimum değerindedir. Yani $(v-m)/T = 1$ olduğunda, bu da taşımanın SLAV başlamadan başlaması demektir. SLAV başlamadan tahmin edebilen algoritma en iyidir [47].

CloudSim'de optimal çevrimiçi deterministik algoritma maliyet hesabı

Eğer bir problemin girdisi çevrimiçi ise ve çıktısı da çevrimiçi olarak oluşturuluyor ise bu problem çevrimiçi problemdir. Çevrimiçi problemler için kullanılan algoritmalara çevrimiçi algoritma denilmektedir. Çevrimiçi algoritmaların performans ve verimliliğini karşılaştırmanın yollarından birisi de rekabet analizi uygulamaktır. Rekabet analizi yapısında çevrimiçi algoritmaların değerleri gelecek bilgisi olan (gelecek işyükünü verisini bilerek yoğunluk yaşanmadan önlem alan), en iyi performansı sergileyecek muhtemel algoritmalar ile karşılaştırılmaktadır. Gelecek bilgisi olan algoritmalar SLAV başlamadan önlem alarak en az maliyetle işlemleri gerçekleştirmektedir. $ALG(I) \leq c.OPT(I) + a$ eşitliğinde $ALG(I)$, I girişi için algoritmanın harcadığı maliyettir. c, sabit faktördür. $OPT(I)$, Optimal online algoritmasının I girişi için maliyetidir ve a da sabit değerdir. c program fonksiyonlarındaki parametrelere bağlı olabilmektedir fakat I'dan bağımsız bir değer olmalıdır [47].

CloudSim'de dinamik sanal makine yerleştirme probleminde maliyet hesabı

Dinamik sanal makine yerleştirme probleminde, toplam maliyeti azaltmak için ne zaman hangi sanal makinenin hangi fiziksel makineye taşınacağını belirlenmektedir. Eşitliği hazırlanırken şu semboller kullanılmaktadır. A_h, herbir fiziksel makinenin CPU kapasitesini ifade etmektedir. A_v, bir sanal makineye atanabilecek en yüksek CPU kapasitesini ifade etmektedir. $m = \frac{A_h}{A_v}$ ile bir fiziksel makineye atanabilecek en yüksek sanal makine sayısı bulunmaktadır. nm ile toplam VM sayısı hesaplanmaktadır. t_m, taşıma süresini ifade etmektedir. $C_p = 1$ enerji tüketim maliyetini ifade etmektedir. $C_v = s$, SLAV maliyetini ifade etmektedir. Bu değerler $C_p = \frac{1}{s}, C_v = 1$ olarak da ifade edilebilmektedir.

$$C = \sum_{t=t_0}^{T} (C_p \sum_{i=0}^{n} a_{t_i} + C_v \sum_{j=0}^{n} v_{t_j}) \quad (4.16)$$

Eş. 4.16'da t_0 başlangıç zamanını ,T toplam zamanı ifade etmektedir. $a_{t_i} \in \{0,1\}, PM \ \ t$ anında fiziksel makinenin aktif mi pasif mi olduğunu göstermektedir, a_{t_i} değeri 1 veya 0 değerlerini alabilmektedir. $v_{t_j} \in \{0,1\}, \ \ t$ anında sanal makinenin SLAV yaşayıp yaşanmadığını göstermektedir, v_{t_j} değeri 0 veya 1 değerlerini alabilmektedir.

Optimal çevrimiçi (online) deterministik algoritmasında rekabet oranı hesaplama

SLAV bir fiziksel makinede en az $m+1$ adet sanal makine varsa oluşmaktadır ve bu sanal makinelerin kullanabilecekleri en yüksek CPU miktarı A_v'dir. Eş. 4.17'de sanal makinelerin üzerinde bulunduğu fiziksel makineyle CPU kapasitesi ilişkisinin hesabı anlatılmaktadır.

$$mA_v = A_h,\ \ k > m,\ \ kA_v > A_h \tag{4.17}$$

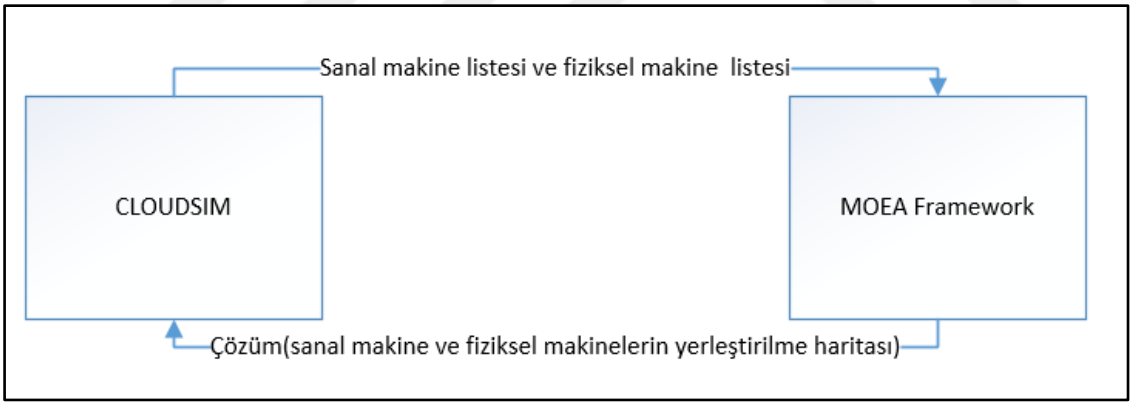
Eşzamanlı olarak SLAV çeken fiziksel makine sayısı n_v ile ifade edilmektedir. $n_v = \frac{nm}{m+1}$ 'dir. SLAV yaşamayan fiziksel makine sayısı ise n_r ile ifade edilmektedir. $n_r = n - n_v$'dir. Herbir zaman periyodu ikiye bölünerek bir zaman peryondu $2t_m$ ile ifade edilmektedir. t_m, taşıma zamanını ifade etmektedir. Bütün fiziksel makineler aktifse ve SLAV yoksa birinci t_m'de maliyet $t_m n C_p$ eşitliği ile hesaplanmaktadır. Bütün fiziksel makineler aktifse, n_v tane fiziksel makine SLAV çekerse ve bazı sanal makineler n_r fiziksel makineye taşınırsa ikinci t_m'de harcanan toplam maliyet $t_m(nC_p + n_v c_v)$ eşitliği ile hesaplanmaktadır. Bir zaman periyodu yani $2t_m$ boyunca harcanan toplam maliyet $C = 2t_m n C_p + t_m n_v c_v$ eşitliği ile hesaplanmaktadır. ALG çevrimiçi algoritmasının sebep olduğu toplam maliyet $ALG(I) = \tau t_m(2nC_p + n_v C_v)$ eşitliği ile hsaplanmaktadır. OPT çevrimdışı algoritmasının sebep olduğu toplam maliyet $OPT(I) = 2\tau t_m n C_p$ eşitliği ile hesaplanmaktadır. Bu iki algoritmanın rekabet oranı $\frac{ALG(I)}{OPT(I)}$=1+$\frac{n_v C_v}{2nC_p}$ bulunmaktadır. C_p'nin yerine $1/s$, C_v'nin yerine 1 yazıldığında $\frac{ALG(I)}{OPT(I)}$=1+$\frac{n_v s}{2n}$ bulunur. $mod\,\frac{nm}{m+1} = 0$ olduğunda, $n_v = \frac{nm}{m+1}$ rekabet oranında yerine koyulursa $\frac{ALG(I)}{OPT(I)}$=$1 + \frac{ms}{2(m+1)}$ bulunmaktadır. Eğer $\frac{nm}{m+1} \neq 0$ ise $\frac{ALG(I)}{OPT(I)}$<$\frac{ms}{2(m+1)}$'dir. Bu iki durum birleştirilirse $\frac{ALG(I)}{OPT(I)}$≤$1 + \frac{ms}{2(m+1)}$ bulunmaktadır.

5. GERÇEKLEŞTİRİLEN ÇALIŞMA

Bu çalışmada, sanallaştırma teknolojisinde karşılaşılan sanal makine yerleştirme probleminin çözümü çok amaçlı optimizasyon algoritmalarıyla ele alınmıştır. Literatürde yaygın kullanımı olan çok amaçlı optimizasyon yöntemleri probleme uygulanmış, başarımları farklı metrikler doğrultusunda değerlendirilmiş ve kıyaslanmıştır.

5.1. Benzetim Ortamında Veri Merkezinin Oluşturulması

Çalışmanın alt yapısında bulut bilişim ortamının benzetimi CloudSim ile gerçekleştirilmiştir. Problemi tanımlamak ve algoritmaları yürütmek için MOEA Framework kütüphanesi Cloudsim ile bütünleştirilmiş ve gerekli uyarlamalar yapılmıştır. Şekil 5.1'de görüldüğü üzere CloudSim'den alınan sanal makine listesi ve fiziksel makine listesi MOEA Framework'e gönderilmekte, MOEA Framework'de elde edilen çözüm tekrar CloudSim'e gönderilmektedir. Elde edilen optimum çözüme göre CloudSim, sanal makine yerleştirme işlemini gerçekleştirmektedir.



Şekil 5.1. Tez çalışmasında kullanılan CloudSim ve MOEA Framework ilişkisi

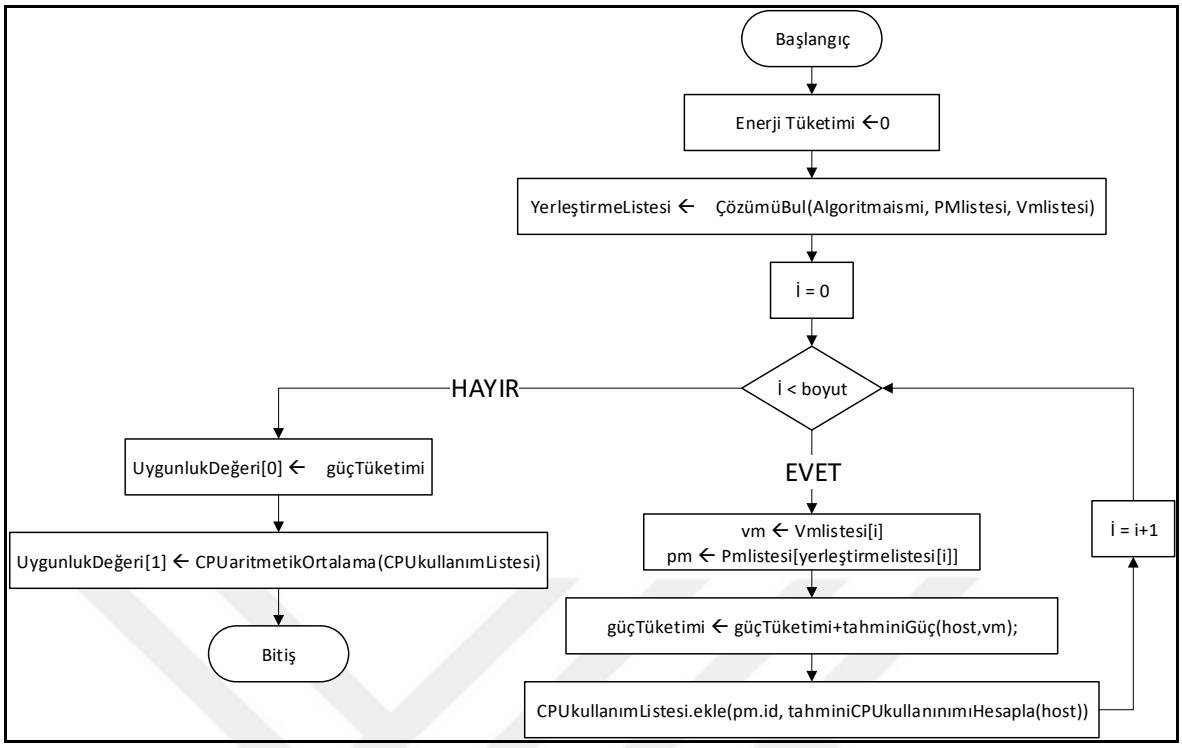
Çalışmanın yürütülebilmesi için bir veri merkezine ihtiyaç bulunmaktadır. Problemin yapısına uygun olarak belli sayıda fiziksel makine ve sanal makinenin tanımlı olması ve belirli bir senaryo dâhilinde işlemesi gerekmektedir. Bu gereklilik veri merkezi PlanetLab verisetinin benzetim ortamında kullanılmasıyla karşılanmıştır.

5.2. Problemin MOEA Problem Yapısına Dönüştürülmesi

MOEA Framework'ün kullanılabilmesi için öncelikle kullanıcının problemini tanımlaması gerekmektedir. Problemin tanımlanması amacıyla *org.cloudbus.cloudsim.power.moea* isimli pakete MOEAProblem.java sınıfı eklenmiştir. Oluşturulan MOEAProblem org.moeaframework.core paketindeki *Problem* sınıfı uygulanarak oluşturulmuştur. MOEAProblem sınıfı parametre olarak taşınacak sanal makine listesini ifade eden *vmstomigrate* ve aktif fiziksel makine listesini ifade eden *availablehosts* listesini alarak override yöntemiyle *evaluate isminde* uygunluk fonksiyonu oluşturulmuştur. Uygunluk fonksiyonunun ayrıntıları bölüm 5.3'de anlatılmaktadır.

5.3. Uygunluk Fonksiyonu

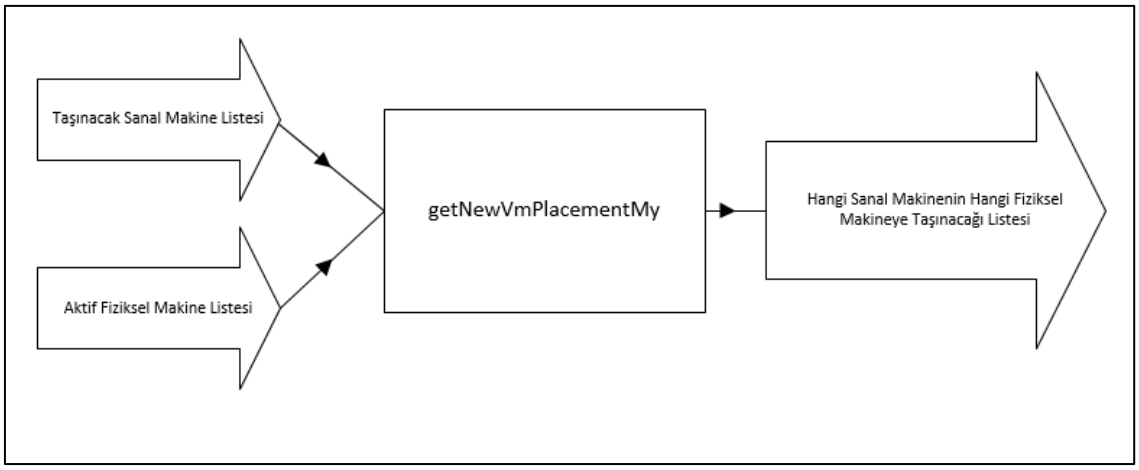
Uygunluk fonksiyonu çözümlerin geçerliliğini kontrol etmektedir ve çözümleri derecelendirmektedir. Bu fonksiyon oluşturulurken Şekil 5.2'de görülen algoritma kullanılmıştır. Bu fonksiyonda *d integer* dizisinde çözüm, *f double* dizisinde amaçlar, g *double* dizisinde kısıtlar tutulmaktadır. Bunun dışında *map* cinsinden oluşturulan *dizi* isimli dizide fiziksel makinelerin CPU kullanım oranları tutulmaktadır. *Double* cinsinden *fark* değişkeninde sanal makine fiziksel makineye atandıktan sonra oluşacak olan güç tüketim farkı tutulmaktadır. *Boolean* cinsinden *check* değişkeni çözüm geçerliyse *true* olarak atanmaktadır. For döngüsü içinde herbir çözümün herbir parçasının gerçekleştirilebilir olup olmadığı kontrol edilmektedir. RAM, CPU durumundan uygun olup olmadığı ve çözümün gerçeklenmesi sonucu harcanan enerji miktarındaki fark hesaplanmaktadır. Çözümlerin tamamı gerçekleştirilebilir ise *check* değeri *true* olarak kalmaktadır ve çözümün amaç ve kısıt değerleri atanmaktadır. Eğer çözüm geçersiz ise kısıt değeri 1 atanmaktadır, amaç değerine ise elenmeleri için en yüksek değer atanmaktadır.



Şekil 5.2. Uygunluk fonksiyonu algoritması iş akış şeması

5.3. Sanal Makine Yerleştirme Probleminin Çözülmesi

800 adet fiziksel makineye 1052 adet sanal makineyi kaynak kullanım amacına ulaşmak için CloudSim açık kaynak kodlu simülasyon yazılımı kullanılmıştır. Burada bulunan sanal makine yerleştirmesi yapan *org.cloudbus.cloudsim.power* paketindeki *PowerVmAllocationPolicyMigrationAbstract.java* sınıfı üzerindeki kodlar tezin amacı doğrultusunda yeniden yazılmıştır. Bu sınıfta bulunan *getNewVmPlacement* ve *getNewVmPlacementFromUnderutilizedHost* fonksiyonlarında değişiklik yapılarak yeniden yazılmıştır. Yeni yazılan sınıflara *getNewPlacementMy* ve *getNewVmPlacementFromunderUtilizedHostMy* isimleri verilmiştir.



Şekil 5.3. Sanal makine yerleştirme probleminin çözüm aşaması

Şekil 5.3'de görüldüğü gibi *getNewPlacementMy* sınıfı taşınacak sanal makinelerin listesi ve aktif fiziksel makinelerin listesini parametre olarak almaktadır, çıkışta ise hangi sanal makinenin hangi fiziksel makineye yerleştirileceği bilgisini içeren bir liste döndürülmektedir. Metodun birinci aşamasında fiziksel sunucu listesindeki aktif fiziksel sunucular tespit edilir. İkinci aşamada problem sınıfı olan MOEAproblem.java taşınacak sanal makine listesi ve aktif fiziksel makineler parametre olarak kullanılarak optimizasyon çalıştırılmaktadır. Bu aşamada karşılaştırma yapmak amacıyla 4 farklı optimizasyon algoritması (NSGA-II, SPEA2, PAES, ϵ-MOEA) çalıştırılmıştır. Sonuçta pareto eğrisi üzerinde bulunan n adet çözüm dönmektedir. Üçüncü aşamada, bir önceki aşamada bulunan n adet çözümden en düşük enerji tüketeni seçilmektedir. Dördüncü aşamada seçilen çözüm değeri integer cinsinden oluşturulan d dizisine atılmaktadır. d dizisi Eş. 3.1'de gösterilen yapıya göre oluşturulmuştur. Çözümü ifade eden d dizisi örneği Şekil 5.4'de gösterilmiştir.

Şekil 5.4. Sanal makinelerin bulunduğu fiziksel makine bilgisini tutan d dizisi

Şekil 5.4'deki gibi ifade edilen çözüm d dizisine atılmaktadır. Bu ifadenin CloudSim yazılımına uyumlu hale getirilmesi gerekmektedir. *String* ve *object* sınıflarını içeren *map* sınıfından nesne oluşturulur. Bunun dışında *migrationmap* isminde *map* cinsinden oluşturulan bağlı listenin oluşturulması gerekmektedir. Daha önce bulunan çözümlerin bulunduğu d dizisinin değerlerinden i, sanal makinenin numarası ve $d[i]$ ise fiziksel makinenin numarası ile *map* sınıfından migrate nesnesi oluşturulmaktadır. Oluşturulan migrate nesnesi *migrationmap* listesine atılmaktadır. Böylece dördüncü adımdaki işlemler ile *MOEA Framework'ten* elde edilen çözüm CloudSim yapısına uygun hale getirilmiş olmaktadır. Belirli bir eşik altındaki yoğunlukta çalışan fiziksel makine üzerindeki sanal makineleri diğer fiziksel sunuculara taşınması için *getnewvmplacementfromunderutilizedhostMy* metodu yazılmıştır. Bu metodda öncelikle aktif fiziksel sunucular tespit edilmektedir. Daha sonra optimizasyon algoritmasına taşınması gereken sanal makineler parametre olarak verilerek çözümler hesaplanmaktadır. Bulunan n adet çözümden en az enerji harcayan çözüm seçilerek d dizisine aktarılmaktadır. Bulunan çözüm CloudSim yazılımının yapısına uyumlu hale getirmek amacıyla listeye aktarılmaktadır.

5.3.1. Optimizasyon algoritmalarının çalıştırılması

Optimizasyon Algoritmaları Şekil 5.5'de görüldüğü gibi çağırılmıştır. MOEA Framework'te bulunan *Executer* fonksiyonu ile optimizasyon algoritması çağrılmaktadır. Parametre olarak oluşturulan problem sınıfı, taşınması gerekli olan sanal makineler ve aktif olarak çalışan fiziksel makineler girilmektedir. Ardından problemin hangi optimizasyon ile çözüleceği bilgisi parametre olarak verilir. Popülasyon sayısı ve döngü sayısı da verildikten sonra parametreler tamamlanmış olur. Bütün algoritmalarda nesil sayısı 500 ve popülasyon sayısı 100 olarak belirlenmiştir.

```
NondominatedPopulation result = new Executor()
                    .withProblemClass(MOEAProblem.class, vmsToMigrate, includeHosts)
                    .withAlgorithm("NSGAII")
                    .withProperty("populationSize", 100)
                    .withMaxEvaluations(500)
                    .run();
```

Şekil 5.5. Optimizasyon algoritmasının çağrılmasından bir kod kesiti

6. PEFORMANS DEĞERLENDİRMESİ

CloudSim ortamında oluşturulan veri merkezinin sanal makine yerleştirme problemini çözmek amacıyla NSGAII, SPEA2, ϵ-MOEA, PAES algoritmaları çalıştırılmıştır ve elde edilen sonuçlara göre performans değerlendirmesi yapılmıştır. Çok amaçlı optimizasyon algoritmalarında ortalama CPU kullanımını artırmak ve toplam enerji tüketimini azaltmak hedeflendiğinden bu metrikler incelenmiştir. CPU kullanımında yük dengeleme sunucunun performansı açısından çok önemlidir. Enerji tasarrufu açısından CPU kullanımı artırıldığında sunucuda hizmet kalitesinin düşme ihtimali artmaktadır. Algoritmanın sağladığı çözümün hizmet kalitesini tespit etmek için bununla ilgili olan VM taşıma sayısı, SLAPDM, SLATAH ve SLAV metrikleri de incelenmiştir. Simülasyon sonuçlarında elde edilen enerji tüketimi, VM taşıma sayısı, SLAPDM, SLATAH, SLAV ve ortalama CPU kullanımı verileri Çizelge 6.1'de ve EK-1 - EK-4'de görülmektedir.

Çizelge 6.1. Çalışmada elde edilen sonuçlar

Algoritma	Enerji Tüket. kWh	VM Taş. Say.	SLAPDM %	SLATAH %	SLAV %	Ortalama CPU Kull.
NSGAII	249,83	49995	0,28	9,52	0,02688	0,15267
SPEA2	251,32	50136	0,29	9,67	0,02798	0,15551
ϵ-MOEA	267,64	54461	0,35	9,92	0,03504	0,15585
PAES	268,60	55880	0,34	9,99	0,03364	0,16808

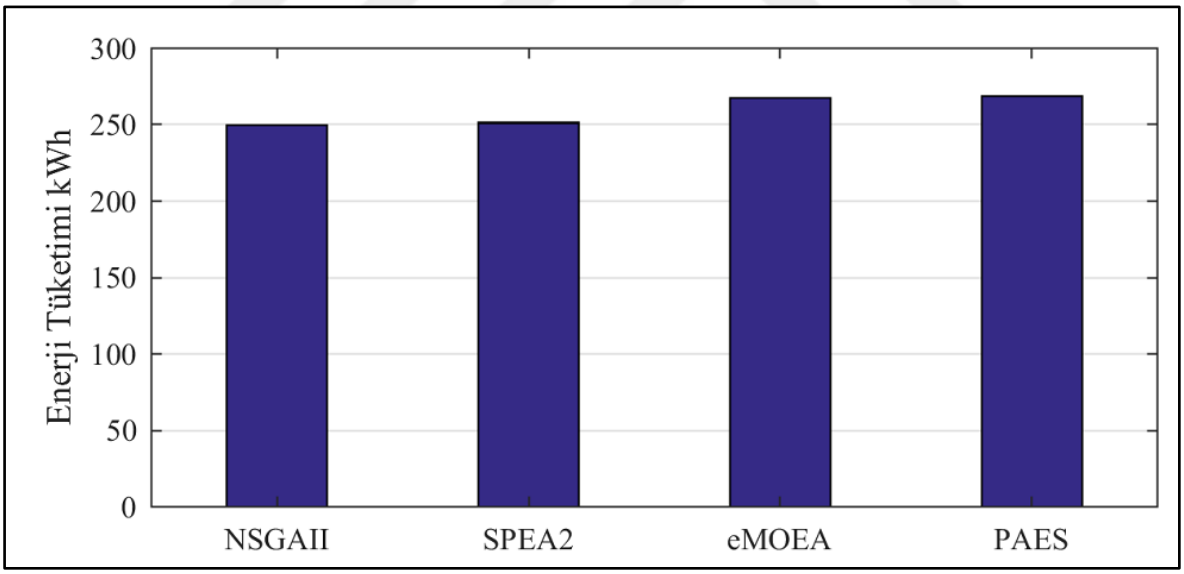
Çizelge 6.2'de en düşük ve en yüksek değeri veren algoritmalar görülmektedir. NSGAII algoritması enerji tüketimi, VM taşıma sayısı, SLAPDM, SLATAH ve SLAV değerleri en düşük olan yani avantajlı olan algoritmadır. Aktif fiziksel makine başına düşen ortalama CPU kullanımında en yüksek değeri verdiği için en başarılı algoritma PAES'dir.

Çizelge 6.2. Çalışma sonucunda elde edlen en düşük değerler (E.d.d) ve en yüksek değerler (E.y.d.)

Metrik	Algoritma	E.d.d.	Algoritma	E.y.d
Enerji tüketimi	SPEA2	251,32	PAES	268,60
VM taşınma sayısı	NSGAII	49995	PAES	55880
SLAPDM	NSGAII	0,28%	ϵ-MOEA	0,35%
SLATAH	NSGAII	9,52%	PAES	9,99%
SLAV	NSGAII	0,02688%	ϵ-MOEA	0,03504%
Ortalama CPU Kull.	NSGAII	0,15267	PAES	0,16808

6.1. Enerji Tüketimi'ne Göre Başarım Değerlendirmesi

Optimizasyondaki iki amaçtan birisi olan enerji metriği benzetim ortamında oluşturulan veri merkezindeki sunucuların harcadığı enerjiyi ifade etmektedir. Bölüm 4.5.5'de tüketilen enerjinin nasıl hesaplandığı anlatılmaktadır.



Şekil 6.1. Algoritmaların enerji tüketim değerleri

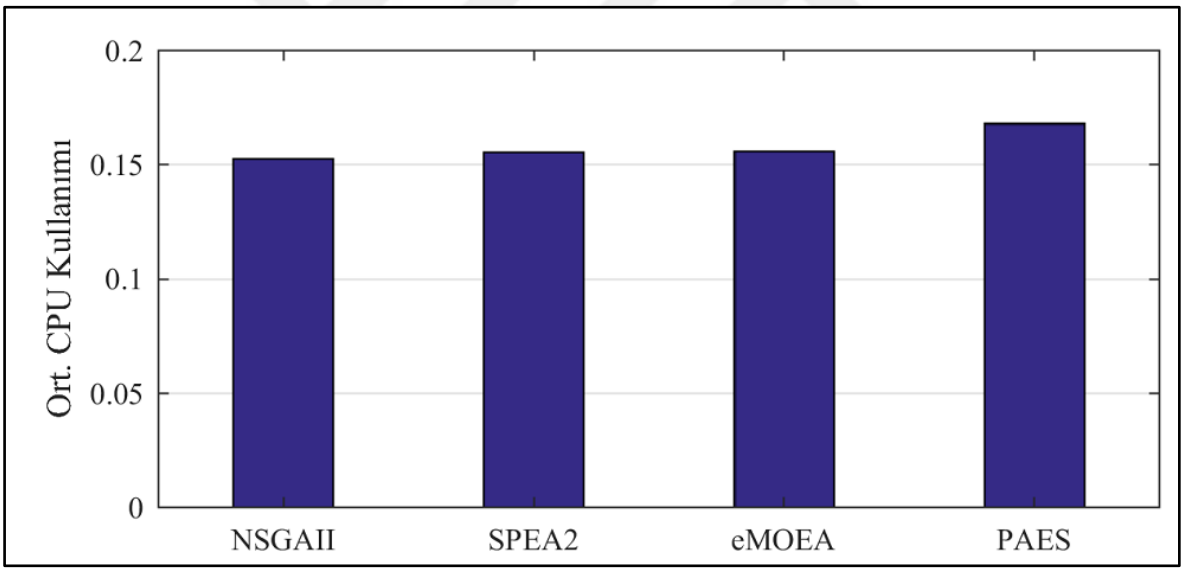
Çizelge 6.1 ve Şekil 6.1'de görüldüğü üzere enerji tüketimi en az olandan en çok olana göre sıralama yapıldığında NSGAII, SPEA2, ϵ-MOEA, PAES şeklinde sıralanır. Enerji konusunda en avantajlı algoritma Çizelge 6.2'de görüldüğü gibi NSGAII algoritması olmuştur. NSGAII, SPEA2 değerleri sırasıyla 249kwh ve 251 kwh olarak bulunmuştur.

Enerji tüketimi en yüksek olan algoritmalar ise PAES algoritmasıdır, değerleri sırasıyla 268 kwh'dır.

En az enerji kullanan algoritmalar değerleri birbirine çok yakın olan SPEA2 ve NSGAII algoritmalarıdır. Enerji kullanımında en iyi olan SPEA2 ve NSGAII algoritmları CPU kullanımı konusunda yine birbirlerine yakın değerlere sahiplerdir ve ortalama bir başarıya sahip olmuşlardır.

6.2. Fiziksel Makine başına düşen ortalama CPU kullanım miktarı'na Göre Başarım Değerlendirmesi

Bu metrik açık olan fiziksel makinelerin ortalama CPU kullanımını hesaplamaktadır. Bu metriğin değerinin yüksek olması kaynak israfının daha az olduğunu ifade etmektedir. Ayrıntılı bilgi Bölüm 4.5.6'da bulunmaktadır.



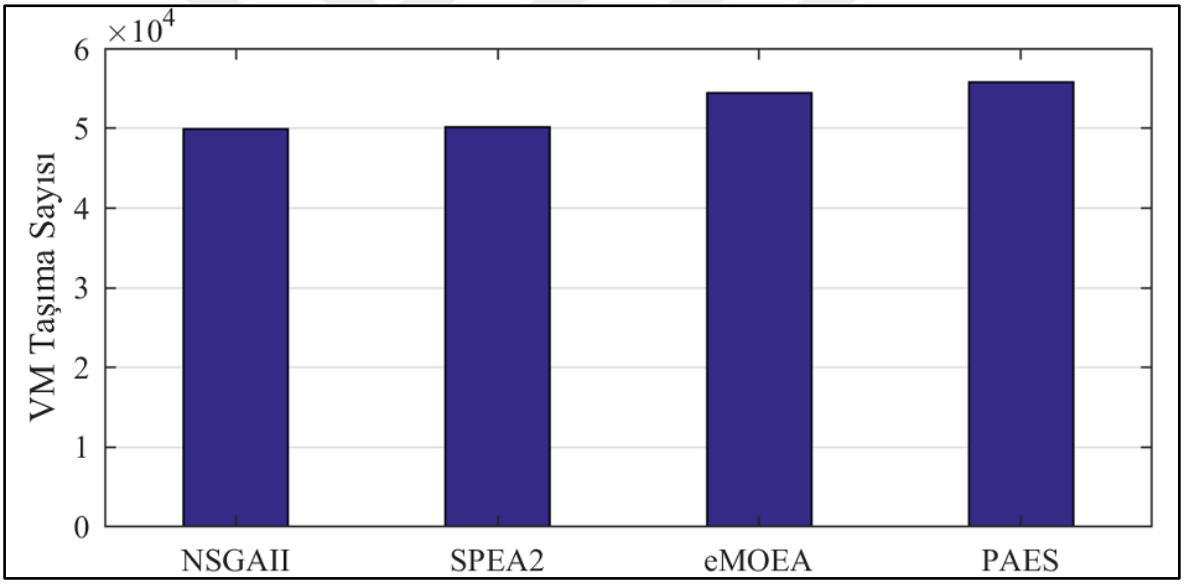
Şekil 6.2. Fiziksel makine başına düşen ortalama CPU kullanım oranları

Çizelge 6.1'de ve Şekil 6.2'de görüldüğü üzere aktif hostlar arasındaki CPU kullanım ortalaması en az olandan en çok olana göre sıralama NSGAII, SPEA2, ϵ-MOEA, PAES şeklinde gerçekleşmiştir. En avantajlı algoritma %16,808 değeri ile PAES'dir. PAES'in CPU kullanımında en avantajlı olup enerji tüketiminde en dezavantajlı algoritmalardan birisi olması başarı konusunda tutarsız olduğunu göstermektedir. NSGAII, ϵ-MOEA ve SPEA2 algoritmalarının ortalama CPU kullanım değerleri birbirlerine çok yakındır; sırasıyla %15,267, %15,585, %15,551'dir. [18]'de açık fiziksel makinelerin kaynak

kullanım miktarları da tek amaçlı optimizasyon algoritması olan Genetik Algoritma, çok amaçlı optimizasyon algoritmaları olan NSGA ve NSGAII arasında karşılaştırılmıştır, sonuçta NSGAII algoritması en avantajlı algoritma olmuştur.

6.3. Sanal Makine Taşınma Sayısı'na Göre Başarım Değerlendirmesi

Yoğun olan sanal makineleri rahatlatmak için daha az yoğun olan makinelere sanal makine taşınması yapılır. Bazen de az yoğun makineler üzerindeki sanal makineler başka makineler üzerine taşınarak boşaltılan fiziksel makineler kapatılır. Yapılan bu taşıma işlemi fazladan kaynak ve enerji tüketimine neden olmaktadır. Bu da performansı olumsuz yönde etkilemektedir. Sanallaştırma sistemlerinde sanal makine taşınması olsa da mümkün olduğunca az gerçekleşmesi beklenmektedir.

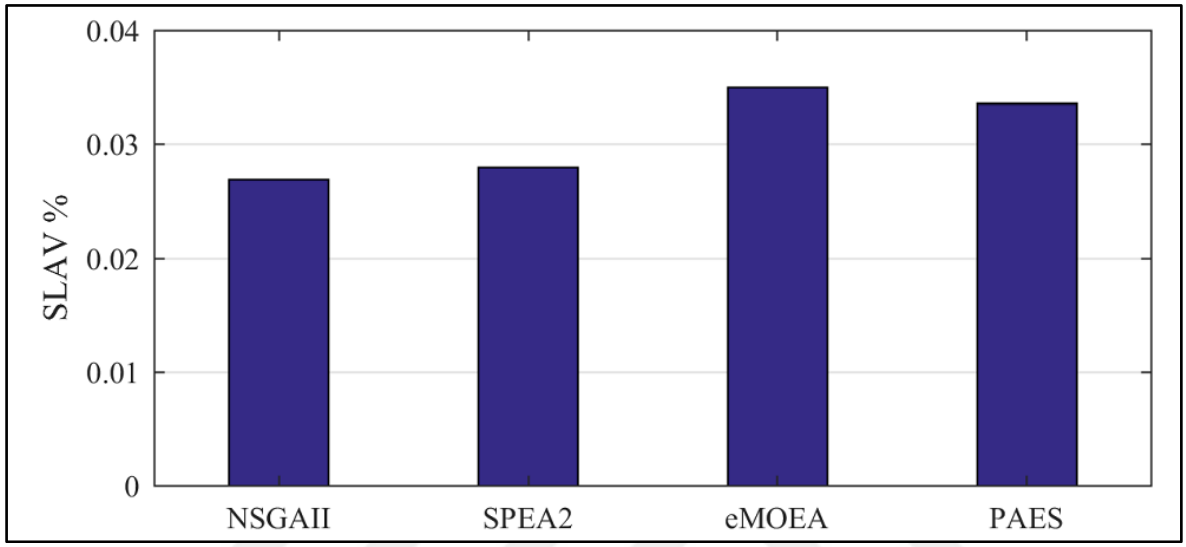


Şekil 6.3. Sanal makine taşınma sayıları

Çizelge 6.1'de ve Şekil 6.3'de görüldüğü üzere VM taşınma sayısı en az olandan en çok olana doğru NSGAII, SPEA2, ϵ-MOEA, PAES şeklinde sıralanmaktadır. VM taşıma işlemi performans düşürücü bir iş olduğundan en avantajlı algoritma Çizelge 6.2'de görüldüğü gibi 49995 değeri ile NSGAII algoritmasıdır. En çok taşıma yapan algoritma ise 55880 değeri ile PAES algoritmasıdır.

6.4. Hizmet Seviyesi Anlaşması İhlaline Göre Başarım Değerlendirme

SLAV, fiziksel makinenin CPU kapasitesinin üzerindeki sanal makinelere yetmediği durumlarda ortaya çıkmaktadır. SLAV ile ilgili ayrıntılı bilgi Bölüm 4.5.1'de bulunmaktadır.

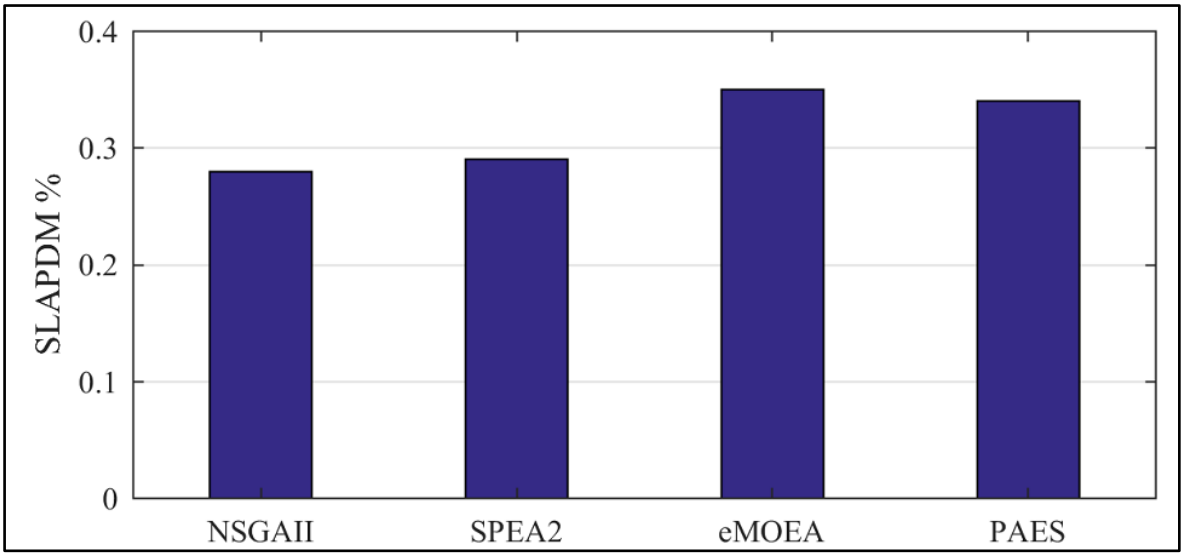


Şekil 6.4. Hizmet seviyesi anlaşması ihlali (SLAV)

Çizelge 6.1 de ve Şekil 6.4'de görüldüğü üzere SLAV değeri en az olandan en çok olana göre NSGAII, SPEA2, PAES, ϵ-MOEA şeklinde sıralanmaktadır. NSGAII ve SPEA2 algoritmalarının değerleri %0,02688 ve %0,02798 olarak birbirine çok yakın değerlere sahiptir. ϵ-MOEA ve PAES algoritmalarının değerleri de %0,03504, %0,03364 olmak üzere birbirlerine yakındır.

6.5. Sanal Makine Göçünden Kaynaklanan Performans Düşüşüne Göre Başarım Değerlendirmesi

SLAPDM, sanal makine taşımasından kaynaklanan performans düşüşünü ifade etmektedir. Bu metrikle ilgili ayrıntılı bilgi Bölüm 4.5.2'de verilmiştir.

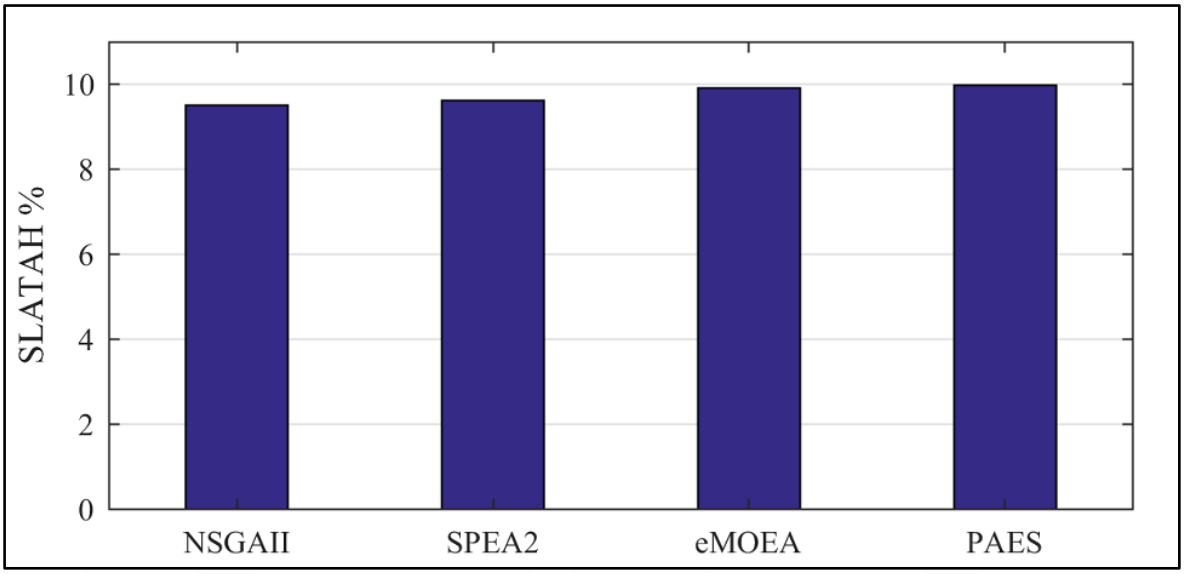


Şekil 6.5. Sanal makine taşınmasından kaynaklanan performans düşüşü

Çizelge 6.1 de ve Şekil 6.5'de görüldüğü üzere SLAPDM değeri en az olandan en çok olana göre NSGAII, SPEA2, PAES, ϵ-MOEA şekinde sıralanmaktadır. En düşük değerleri veren NSGAII ve SPEA2 algoritmalarının SLAPDM değerleri sırasıyla %0,28 ve %0,29 olmak üzere birbirlerine çok yakınlardır. PAES, ϵ-MOEA sonuç değerleri ise sırasıyla %0,34, %0,35 olmak üzere yakın değerlere sahiptir. SLAV metriğine göre yapılan sıralama ile SLAPDM metriğine göre yapılan sıralama aynıdır. Bu da SLAPDM değerinin SLAV değerini etkilediğini göstermektedir.

6.6. Aktif Fiziksel Makine Başına Düşen SLAV Zamanına Göre Performans Değerlendirmesi

SLATAH, aktif fiziksel makine başına düşen ortalama SLAV yaşanma zamanını ifade etmektedir. Ayrıntılı bilgi Bölüm 4.5.4'de bulunmaktadır.



Şekil 6.6. Aktif fiziksel makine sayısı başına düşen ortalama SLAV zamanı (SLATAH)

Çizelge 6.1'de ve Şekil 6.6'da görüldüğü üzere SLATAH değeri en az olandan en çok olana göre NSGAII, SPEA2, ϵ-MOEA, PAES şeklinde sıralama yapılmaktadır ve değerleri de sırasıyla %9,52, %9,67, %9,92, %9,99'dur. Burada diğer performans metriklerinden farklı olarak SLATAH değerleri birbirlerine çok yakın değerlerdir. Eş. 4.1'e göre değerlendirildiğinde, SLATAH değerleri neredeyse eşit olduğundan SLAV sıralamasındaki en büyük etken PDM'dir. PDM'nin sıralamasıyla SLAV sıralamaları aynı ve değerler de birbirleriyle orantılı olması da beklenen durumdur.

7. SONUÇ

Bu tezin konusu bulut bilişimde çok amaçlı optimizasyon algoritmaları ile dinamik yük dengelemedir. Bu kapsamda sanal makine yerleştirme problemi ele alınmıştır. Sanal makinelerin barındığı fiziksel makineler üzerindeki yük dengelenirken açık olan makinelerin ortalama CPU kullanımı ve toplam harcanan enerjinin azaltılması amaçlanmıştır. Çok amaçlı optimizasyon algoritmaları olan PAES, NSGAII, SPEA2, ϵ-MOEA algoritmaları problemin çözümünde kullanılmıştır. Sonuçta ortalama CPU kullanımı konusunda en avantajlı algoritmanın PAES olduğu tespit edilmiştir. En düşük enerji tüketimini sağlayan algoritmalar ise NSGAII ve SPEA2 olmuştur. SLAV metriği açısından NSGAII ve SPEA2 en avantajlı değerleri vermiştir. Yapılan çalışmalar, genel olarak NSGAII ve SPEA2'nin daha üstün olduğunu göstermiştir. Bu tez çalışması, veri merkezlerinin harcadığı enerjinin azaltılmasıyla doğanın korunması; kaynak israfının azaltılmasıyla maliyetin azaltılması konusunda yararlı bir çalışma olmuştur. Gelecekte farklı veri setleri ve farklı algoritmalar ile tez çalışmasının geliştirilmesi planlanmaktadır.

KAYNAKLAR

1. Sanderson, D. (2009). *Programming google app engine: build and run scalable web apps on google's infrastructure*. United States: O'Reilly Media, 1-13

2. Chappell, D. (2008). Introducing the Azure services platform. *White paper*, 1364(11).

3. Samani, R., Reavis, J. and Honan, B. (2014). *CSA guide to cloud computing: Implementing cloud privacy and security*. United States: Syngress, 2-8.

4. Zhang, Y. (2018). *Virtualization and Cloud Computing, in Network Function Virtualization: Concepts and Applicability in 5G Networks*. New Jersey: Wiley, 192.

5. Wang, S., Gu, H., and Wu, G. (2013). *A new approach to multi-objective virtual machine placement in virtualized data center*. Networking, Architecture and Storage (NAS), 2013 IEEE Eighth International Conference, Xi'an.

6. Salapura, V. (2012, September). *Cloud computing: Virtualization and resiliency for data center computing*. In Computer Design (ICCD), 2012 IEEE 30th International Conference on, New Jersey.

7. Xu, J. and Fortes, J.A. (2010, December). *Multi-objective virtual machine placement in virtualized data center environments.* In Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on and Int'l Conference on Cyber, Physical and Social Computing (CPSCom), New Jersey.

8. Xu, M., Tian, W. and Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience*, 29(12).

9. Fang, Y., Wang, F. and Ge, J. (2010, October). *A task scheduling algorithm based on load balancing in cloud computing.* In International Conference on Web Information Systems and Mining, Heidelberg.

10. Ortigoza, J., López-Pires, F. and Barán, B. (2016). *Workload generation for VMP in cloud computing environments.* 2016 XLII Latin American Computing Conference, Valparaíso.

11. Gao, Y., Guan, H., Qi, Z., Hou, Y. and Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, 79(8), 1230-1242.

12. Malekloo, M. and Kara, N. (2014, December). *Multi-objective ACO virtual machine placement in cloud computing environments.* In Globecom Workshops (GC Wkshps), New Jersey.

13. Kjamilji, A. (2014, May). *Multi-objective optimizations during parallel processing in a dynamic heterogeneous cloud environment.* In Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on, New Jersey.

14. Xu, J. and Fortes, J. A. (2010, December). *Multi-objective virtual machine placement in virtualized data center environments.* In Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on and Int'l Conference on Cyber, Physical and Social Computing, New Jersey.

15. Ma, F. and Zhang, L. (2015, September). *Multi-objective optimization for dynamic virtual machine management in cloud data center.* In Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on, New Jersey.

16. Xu, B., Peng, Z., Xiao, F., Gates, A.M. and Yu, J.P. (2015). Dynamic deployment of virtual machines in cloud computing using multi-objective optimization. *Soft computing*, 19(8), 2265-2273.

17. Liu, C., Shen, C., Li, S. and Wang, S. (2014, June). *A new evolutionary multi-objective algorithm to virtual machine placement in virtualized data center.* In Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on, New Jersey.

18. Adamuthe, A.C., Pandharpatte, R.M. and Thampi, G.T. (2013, November). *Multiobjective virtual machine placement in cloud environment.* In Cloud and Ubiquitous Computing and Emerging Technologies (CUBE), 2013 International Conference on, New Jersey.

19. Zheng, Q., Li, R., Li, X. and Wu, J. (2015, May). *A multi-objective biogeography-based optimization for virtual machine placement.* In Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on, New Jersey.

20. Xu, J. and Fortes, J. (2011, June). *A multi-objective approach to virtual machine management in datacenters*. In Proceedings of the 8th ACM international conference on Autonomic computing, New Jersey.

21. Masdari, M., Nabavi, S.S. and Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. *Journal of Network and Computer Applications*, 66, 106-127.

22. Jamali, S. and Malektaji, S. (2014, October). *Improving grouping genetic algorithm for virtual machine placement in cloud data centers.* In Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on, New Jersey.

23. Joseph, C.T., Chandrasekaran, K. and Cyriac, R. (2015). A novel family genetic approach for virtual machine allocation. *Procedia Computer Science*, 46, 558-565.

24. Buyya, R., Ranjan, R. and Calheiros, R.N. (2009, June). *Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities.* In High Performance Computing and Simulation, International Conference on, New Jersey.

25. Al Nuaimi, K., Mohamed, N., Al Nuaimi, M. and Al-Jaroodi, J. (2012, December). *A survey of load balancing in cloud computing: Challenges and algorithms*. In Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on, New Jersey.

26. Alla, Y.H. (2015). *Time dependent virtual machine consolidation with SLA (Service Level Agreement) consideration.* Master's thesis, Department of Informatics, University of Oslo, Oslo, 9-22.

27. Taddei, P.A. (2015). *Design and Development of a CloudSim Module to Model and Evaluate Multi-resource Dependencies.* Bachelor Thesis, Communication Systems Group (CSG), Department of Informatics (IFI), University of Zurich, Zurich, 1-18.

28. Hadka, D. (2016). *Beginner's guide to the MOEA framework.* Pennsylvania: CreateSpace Independent Publishing Platform, 1-46.

29. Beloglazov, A. (2013). *Energy-efficient management of virtual machines in data centers for cloud computing.* Doctoral dissertation, Department of Computing and Information Systems, The University of Melbourne, Melbourne, 1-203.

30. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2), 182-197.

31. Xu, J. and Fortes, J. A. (2010, December). *Multi-objective virtual machine placement in virtualized data center environments.* 2010 IEEE/ACM Int'l Conference on and Int'l Conference on Cyber, Physical and Social Computing, New Jersey.

32. Shabani, I., Kovaçi, A. and Dika, A. (2014, May). *Possibilities offered by Google App Engine for developing distributed applications using datastore.* In Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on, New Jersey.

33. Rostami, S. (2014). *Preference focussed many-objective evolutionary computation* Doctoral dissertation, Manchester Metropolitan University, Faculty of Science and Engineering, Manchester, 28-60.

34. Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. United States: John Wiley and Sons, 1-272.

35. Srinivas, N. and Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3), 221-248.

36. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.A.M.T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2), 182-197.

37. Kreutzer, W., Hopkins, J. and Van Mierlo, M. (1997, December). *SimJAVA—a framework for modeling queueing networks in Java.* In Proceedings of the 29th conference on Winter simulation IEEE Computer Society, 483-488.

38. Xu, M., Tian, W. and Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. *Concurrency and Computation: Practice and Experience*, 29(12).

39. Deb, K., Mohan, M. and Mishra, S. (2003). A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. *KanGAL report*, *2003002*, 1-18.

40. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1), 23-50.

41. Beume, N., Naujoks, B. and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3), 1653-1669.

42. Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6), 712-731.

43. Li, K., Kwong, S., Zhang, Q. and Deb, K. (2015). Interrelationship-based selection for decomposition multiobjective optimization. *IEEE Transactions on Cybernetics*, 45(10), 2076-2088.

44. Zhu, Q., Lin, Q., Chen, W., Wong, K. C., Coello, C. A. C., Li, J. and Zhang, J. (2017). An External Archive-Guided Multiobjective Particle Swarm Optimization Algorithm. *IEEE transactions on cybernetics*, 47(9), 2794-2808.

45. Zitzler, E., Laumanns, M. and Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report*, *103*.

46. Zitzler, E., and Künzli, S. (2004, September). *Indicator-based selection in multiobjective search*. International Conference on Parallel Problem Solving from Nature, Berlin, Heidelberg.

47. Beloglazov, A. and Buyya, R. (2012). Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud datacenters. *Concurrency and Computation: Practice and Experience*, 24(13), 1397-1420.

48. Tian, W., Xu, M., Chen, A., Li, G., Wang, X. and Chen, Y. (2015). Open-source simulators for cloud computing: Comparative study and challenging issues. *Simulation Modelling Practice and Theory*, 58, 239-254.

49. Jain, N. and Choudhary, S. (2016, March). *Overview of virtualization in cloud computing.* In Colossal Data Analysis and Networking (CDAN) Symposium on, New Jersey.

50. Deb, K. (2001). *Optimization for engineering design: algorithms and examples.* (2). New Delhi: Twelfth Printing, 85-142.

51. Dökeroğlu, T. (2014). *Multiobjective relational data warehouse design for the cloud.* Doktora Tezi, Orta Doğu Teknik Üniversitesi Fen Bilimleri Enstitüsü, Ankara, 1-15.

52. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A. and Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.

53. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J. and Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision support systems*, 51(1), 176-189.

54. Subashini, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1), 1-11.

55. Binitha, S. and Sathya, S. S. (2012). A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2), 137-151.

56. Marler, R.T. and Arora, J.S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6), 369-395.

57. Laumanns, M., Thiele, L., Deb, K. and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3), 263-282.

58. Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E.K., Moatti, Y. and Lorenz, D.H. (2011, June). *Guaranteeing high availability goals for virtual machine placement.* In Distributed Computing Systems (ICDCS), 2011 31st International Conference on, New Jersey.

59. Patel, P., Ranabahu, A.H. and Sheth, A.P. (2009, January). Service level agreement in cloud computing. The Ohio Center of Excellence in Knowledge–Enablede Computing (Kno.e.sis), 1(78).

60. İnternet: *University, Princeton. TITLE. PlanetLab.* URL:http://www.webcitation.org/query?url=https%3A%2F%2Fwww.planet-lab.org%2F&date=2018-01-09, Son Erişim Tarihi: 09.01.2018.

61. Fan, X., Weber, W.D. and Barroso, L.A. (2007, June). Power provisioning for a warehouse-sized computer. *In ACM SIGARCH Computer Architecture News*, 35(2), 13-23.

62. İnternet: *Population of San Francisco.* URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.census.gov%2Fquickfacts%2Ffact%2Ftable%2Fsanfranciscocitycalifornia%2CCA%2FPST040216%23viewtop&date=2018-03-23, Son Erişim Tarihi: 23.03.2018

63. Rosenblum, M. (1999, October). Vmwares virtual platform. *In Proceedings of hot chips* 1999(11), 185-196.

64. Velte, A. and Velte, T. (2009). *Microsoft virtualization with Hyper-V*. New York: McGraw-Hill.

65. Xu, Z., Yang, L. and Lei, J. (2015). *Conception and design of desktop virtualization cloud platform for primary education: Based on the citrix technology*, 2015

International Conference of Educational Innovation through Technology (EITT), Wuhan.

66. İnternet: Hadka, D. *MOEA Framework*. URL: http://www.webcitation.org/query?url=http%3A%2F%2Fmoeaframework.org%2F&date=2018-03-29, Son Erişim Tarihi: 29.03.2018.

67. İnternet: PlanetLab Datasets. *The Trustees of Princeton University*. URL: http://www.webcitation.org/query?url=https%3A%2F%2Fwww.planet-lab.org%2Fdatasets&date=2018-03-29, Son Erişim Tarihi: 29.03.2018.

EKLER

EK 1. NSGA-II Algoritması Benzetim Çalışması Sonuçları

Number of hosts	800
Number of VMs	1052
Total simulation time	86400,00 sec
Energy consumption	249,83 kWh
Number of VM migrations	49995
SLA	0,03%
SLA perf degradation due to migration	0,28%
SLA time per active host	9,52%
Overall SLA violation	2,26%
Average SLA violation	13,21%
Number of host shutdowns	12805
Mean time before a host shutdown	680,90 sec
StDev time before a host shutdown	1383,32 sec
Mean time before a VM migration	19,47 sec
StDev time before a VM migration	8,11 sec
Mean time before a VM migration	0,02525 sec
StDev time before a VM migration	0,01269 sec
Execution time - VM selection mean	0,00967 sec
Execution time - VM selection stDev	0,00729 sec
Execution time - host selection mean	0,05833 sec
Execution time - host selection stDev	0,02020 sec
Execution time - total mean	0,44622 sec
Execution time - total stDev	0,25038 sec
CPU Utilization mean	0,15267
BUILD SUCCESSFUL (total time: 29 minutes 46 seconds)	

EK 2. ϵ-MOEA Algoritması Benzetim Çalışması Sonuçları

Number of hosts	800
Number of VMs	1052
Total simulation time	86400,00 sec
Energy consumption	267,64 kWh
Number of VM migrations	54461
SLA	0,04%
SLA perf degradation due to migration	0,35%
SLA time per active host	9,92%
Overall SLA violation	3,62%
Average SLA violation	15,96%
Number of host shutdowns	14747
Mean time before a host shutdown	639,92 sec
StDev time before a host shutdown	1216,84 sec
Mean time before a VM migration	19,36 sec
StDev time before a VM migration	8,09 sec
Mean time before a VM migration	0,03450 sec
StDev time before a VM migration	0,02919 sec
Execution time - VM selection mean	0,01161 sec
Execution time - VM selection stDev	0,00695 sec
Execution time - host selection mean	0,09392 sec
Execution time - host selection stDev	0,04817 sec
Execution time - total mean	0,63371 sec
Execution time - total stDev	0,41667 sec
CPU Utilization mean	0,15585
BUILD SUCCESSFUL (total time: 42 minutes 22 seconds)	

EK 3. PAES Algoritması Benzetim Çalışması Sonuçları

Number of hosts	800
Number of VMs	1052
Total simulation time	86400,00 sec
Energy consumption	268,60 kWh
Number of VM migrations	55880
SLA	0,03%
SLA perf degradation due to migration	0,34%
SLA time per active host	9,99%
Overall SLA violation	3,30%
Average SLA violation	14,98%
Number of host shutdowns	14938
Mean time before a host shutdown	639,24 sec
StDev time before a host shutdown	1146,82 sec
Mean time before a VM migration	19,39 sec
StDev time before a VM migration	8,10 sec
Mean time before a VM migration	0,03876 sec
StDev time before a VM migration	0,02182 sec
Execution time - VM selection mean	0,01257 sec
Execution time - VM selection stDev	0,00567 sec
Execution time - host selection mean	0,09013 sec
Execution time - host selection stDev	0,04294 sec
Execution time - total mean	0,69351 sec
Execution time - total stDev	0,58376 sec
CPU Utilization mean	0,16808
BUILD SUCCESSFUL (total time: 53 minutes 54 seconds)	

EK 4. SPEA2 Algoritması Benzetim Çalışması Sonuçları

Number of hosts	800
Number of VMs	1052
Total simulation time	86400,00 sec
Energy consumption	251,32 kWh
Number of VM migrations	50136
SLA	0,03%
SLA perf degradation due to migration	0,29%
SLA time per active host	9,67%
Overall SLA violation	2,39%
Average SLA violation	13,43%
Number of host shutdowns	13089
Mean time before a host shutdown	670,76 sec
StDev time before a host shutdown	1338,18 sec
Mean time before a VM migration	19,52 sec
StDev time before a VM migration	8,09 sec
Mean time before a VM migration	0,03230 sec
StDev time before a VM migration	0,01698 sec
Execution time - VM selection mean	0,01111 sec
Execution time - VM selection stDev	0,00561 sec
Execution time - host selection mean	0,12607 sec
Execution time - host selection stDev	0,05237 sec
Execution time - total mean	1,38601 sec
Execution time - total stDev	0,67321 sec
CPU Utilization mean	0,15551
BUILD SUCCESSFUL (total time: 44 minutes 10 seconds)	

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : DÖRTERLER, Serap
Uyruğu : T.C.
Doğum tarihi ve yeri : 05.01.1989, Kayseri
Medeni hali : Bekar
Telefon : 03124103130
e-mail : sdorterler@gmail.com

Eğitim	Yer	Yıl
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	Devam ediyor
Lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2011
Lise	Ankara Atatürk Anadolu Lisesi	2006

İş Deneyimi

Yıl	Yer	Görev
2013-Halen	Devlet Malzeme Ofisi Genel Müdürlüğü	Sistem Yöneticisi
2012-2013	Gazi Üniversitesi	Sistem Yöneticisi

Yabancı Dil

İngilizce

Yayınlar

Dörterler, S. , Dörterler, M. and Ozdemir, S. (2017). *Multi-objective virtual machine placement optimization for cloud computing*, 2017 International Symposium on Networks, Computers and Communications (ISNCC), Marrakech, 1-6.

Hobiler

Yürüyüş yapmak, kitap okumak, tiyatro ve sinema izleme.

GAZİ GELECEKTİR..