BULUT TABANLI İÇERİK DAĞITIM AĞLARINDA REPLİKA SUNUCUSU YERLEŞİMİNİN TABU ARAMA ALGORİTMASI İLE OPTİMİZAYONU

Ömer Faruk YILDIRIM

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Dr. Öğr. Üyesi Deniz DAL
2019

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BULUT TABANLI İÇERİK DAĞITIM AĞLARINDA REPLİKA SUNUCUSU YERLEŞİMİNİN TABU ARAMA ALGORİTMASI İLE OPTİMİZASYONU

Ömer Faruk YILDIRIM

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ERZURUM
2019

T.C.
ATATÜRK ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü Müdürlüğü

TEZ ONAY FORMU

BULUT TABANLI İÇERİK DAĞITIM AĞLARINDA REPLİKA SUNUCUSU YERLEŞİMİNİN TABU ARAMA ALGORİTMASI İLE OPTİMİZASYONU

Dr. Öğr. Üyesi Deniz DAL danışmanlığında, Ömer Faruk YILDIRIM tarafından hazırlanan bu çalışma, 28/10/2019 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Bilim Dalı'nda Yüksek Lisans tezi olarak **oybirliği** / ~~**oy çokluğu**~~ (3./0) ile kabul edilmiştir.

Başkan: Dr. Öğr. Üyesi Deniz DAL İmza :

Üye : Prof. Dr. Abdulsamet HAŞILOĞLU İmza :

Üye : Dr. Öğr. Üyesi Saffet Gökçen ŞEN İmza :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu'nun 07./11./2019 tarih ve ..43.../..14..... nolu kararı ile onaylanmıştır.

Prof. Dr. Mehmet KARAKAN
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

BULUT TABANLI İÇERİK DAĞITIM AĞLARINDA REPLİKA SUNUCUSU YERLEŞİMİNİN TABU ARAMA ALGORİTMASI İLE OPTİMİZASYONU

Ömer Faruk YILDIRIM

Atatürk Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Deniz DAL

Son zamanlarda insanların, müzik dinleme, film izleme ve oyun oynama gibi bazı günlük aktivitelerinin işleyişi değişmeye başlamıştır. Önceden fiziksel olarak CD/DVD'ler ile satın alınan müzik ve filmler artık gelişen teknoloji ile beraber akış servis hizmetlerine evrilmiştir. Bu içeriklerin son kullanıcıya akıcı bir şekilde ulaştırılabilmesi için firmalar zaman içerisinde Proxy sunuculara, ardından CDN'lere ve son olarak da bulut tabanlı CDN'lere, yani CCDN'lere yönelmişlerdir. Bu yönelimdeki temel motivasyon, geliştirilmiş bir hizmet kalitesi (QoS) sunabilmektir. Bunu sağlamanın yolu ölçeklenebilir ve kaynak verimliliği yüksek içerik dağıtım ağları kurmaktır. Replika sunucusu yerleşimi, CCDN'lerde kilit bir tasarım konusudur ve replika sunucularının, işletme maliyetini en aza indiren ve son kullanıcıların QoS'sini karşılayan coğrafi olarak dağılmış bulut sitelerine yerleştirilmesine karar vermeyi içerir. Bu sorun NP-Hard türünde bir problemdir. Literatürde bu problemin çözümüne yönelik sezgisel algoritmalar mevcuttur. Bu tez çalışmasında replika sunucusu yerleşimi için ilk defa Tabu Arama algoritması tabanlı bir metasezgisel geliştirilmiştir. Maliyet, bulut hizmet sağlayıcılarının temel maliyet politikası ile belirlenirken, hizmet kalitesi (QoS) içeriğin CCDN içinde barındırıldığı yere göre belirlenmektedir. Test sonuçları önerilen metasezgisel yöntemin işletme maliyeti ve QoS açısından literatürde en son geliştirilen sezgisel algoritmadan daha başarılı olduğunu göstermiştir. Bu tez kapsamında ayrıca ilgili alanda mevcut önemli bir boşluğu dolduracağı düşünülen bir denektaşı seti oluşturulmuştur ve literatüre kazandırılmıştır.

2019, 89 Sayfa

Anahtar Kelimeler: Replika Yerleşimi, Bulut, Metasezgisel, Tabu Arama, İçerik Dağıtım Ağı

ABSTRACT

MS Thesis

OPTIMIZATION OF REPLICA SERVER PLACEMENT ON CLOUD-BASED CONTENT DELIVERY NETWORKS BY TABU SEARCH ALGORITHM

Ömer Faruk YILDIRIM

Ataturk University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor : Asst. Prof. Dr. Üyesi Deniz DAL

Recently, the daily activities of people, such as listening to music, watching movies and playing games, have started to change. Music and movies that were previously physically purchased on CD/DVDs have now evolved into streaming services by means of the advancing technology. In order to deliver these contents to the end user in an uninterrupted fashion, companies have employed to Proxy servers, then to CDNs and finally to cloud-based CDNs, ie CCDNs. The main motivation in this direction is to provide an improved service quality (QoS). The way to achieve this is to build scalable and resource-efficient content delivery networks. Replica server placement is a key design issue in CCDNs and involves the decision process of placement of replica servers on geographically dispersed cloud sites that minimizes the operating cost and meets the end users' QoS. This problem is categorized as an NP-Hard problem and there are heuristic algorithms in the literature for solving it. In this thesis, for the first time, a metaheuristic based on Tabu Search algorithm has been developed for the replica server placement. The cost here is determined by the basic cost policy of cloud service providers, while the quality of service (QoS) is obtained by the location where the content is hosted in the CCDN. The test results showed that the proposed metaheuristic method is more successful in terms of the operating cost and the QoS than the latest heuristic algorithm developed in the literature. Within the scope of this thesis, a benchmark set which is thought to fill an important gap in the related field has been also created and introduced to the literature.

2019, 89 Pages

Keywords: Replica Placement, Cloud, Metaheuristic, Tabu Search, CDN

TEŞEKKÜR

Tez çalışmam öncesinde ve çalışma sürecinde bilgilerini ve tecrübelerini esirgemeyen, kişisel olarak hayatımın en zor süreçlerini yaşadığım bu dönemde göstermiş olduğu sabırdan dolayı değerli danışman hocam Sayın Dr. Öğr. Üyesi Deniz DAL'a, çalışma yaptığım süre boyunca manevi desteğini esirgemeyen başta babam olmak üzere çok kıymetli aileme teşekkür ederim.

Ömer Faruk YILDIRIM
Kasım 2019

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar

API	: Uygulama Program Arayüzü (Application Programming Interface)
CDN	: İçerik Dağıtım Ağı (Content Delivery Network)
GU	: Açgözlü Kullanıcı (Greedy User)
GS	: Açgözlü Alan (Greedy Site)
LUG	: En Az Kullanım Açgözlü Algoritması (Least Usage Greedy)
SPBC	: Shortest Path Betweenness Centarlity
SLA	: Hizmet Seviyesi Anlaşması (Service Level Agreement)
QoS	: Ağ iletişimi Hizmet Kalitesi (Quality of Service)
ISP	: Internet Service Provider
ISS	: İnternet Servis Sağlayıcı
RRI	: İstek Yönlendirme Alt Yapısı (Request Routing Infrastructure)
RTT	: Gidiş Dönüş Süresi (Round Trip Time)
SA	: Benzetilmiş Tavlama (Simulated Annealing)
TA	: Tabu Arama (Tabu Search)
ODP	: Talep Yük Fiyatı (On Demand Price)
SP	: Depolama Fiyatı (Storage Price)
DNS	: Alan Adı Sunucusu (Domain Name Server)
ERP	: Kurumsal Kaynak Planlaması (Enterprise Resource Planning)
CRM	: Müşteri İlişkileri Yönetimi (Customer Relationship Management)
Benchmark	: Denektaşı
IaaS	: Servis Olarak Altyapı (Infrastructure as a Service)
PaaS	: Servis Olarak Platform (Platform as a Service)
SaaS	: Servis Olarak Yazılım (Software as a Service)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Her geçen gün hızla gelişen teknoloji günlük hayatımızın vazgeçilmez bir parçası olan teknolojik ürünlerin sayısının artmasına yol açmıştır. Söz konusu bu ürünler ilk başlarda sadece sabit bir amaç için kullanılırken veya fayda sağlarken, teknolojik gelişmeler sayesinde ve özellikle internetin hayatımıza girmesi neticesinde bir paradigma kayması yaşanmış ve bu durum beraberinde bu aletlerin kullanım senaryolarında değişikliğe neden olmuştur. Böylece farklı ihtiyaçlar için yeni çözümler üretilmeye başlanmıştır. Bu yeni çözümler yeni sorunlar doğurmuş ve bu döngü böylece devam edegelmiştir. Daha da önemlisi bu teknolojik gelişim kocaman bilgisayarların işlevselliklerini cebimizde taşıyabildiğimiz cihazlar tarafından karşılanabilecek hale getirmiştir. En nihayetinde bu özgürlük artık kendisinden vazgeçilemeyecek bir noktaya ulaşmış ve insan makine etkileşimi hiç olmadığı kadar bir üst seviyeye taşınmıştır. Bu sürekli etkileşim ve her an ulaşılabilir olma arzusu mobil ve internet teknolojilerinin makul fiyatlarla son kullanıcının erişimine sunulabilmesi lüksü ile birleşince artık her bir bireyin hayatının temel ihtiyaçlarından biri olmasının önünü açmıştır.

Yukarıda bahsedilen ve değişen kullanıcı alışkanlıklarına dikkat çeken bilgileri desteklemek amacıyla *We are in Social* raporuna göz atılabilir (Anonim 2018). Bu rapor bireysel teknoloji kullanım yoğunluğunu bir önceki senenin verilerini baz alarak her sene Ocak ayı sonunda bir rapor formunda paylaşıma açmaktadır. Bu rapora göre Türkiye'deki kullanıcıların internette geçirdiği ortalama süre 7 saattir. Bu sürenin yaklaşık üç saatinin sosyal medya kullanımı, üç saatinin video akışı servisleri ve 1 saatinin de müzik dinleme servisleri ile ilişkili olduğu rapor edilmektedir. Öte yandan toplam nüfusun %67'sinin internet kullandığı ve bu kullanıcıların %63'ünün bu erişimi mobil ve akıllı bir telefon üzerinden gerçekleştirdiği anlaşılmaktadır. Yine aynı raporda her gün en az bir kere internete giren kullanıcıların oranının %84 ve haftada en az bir kere internetten faydalanan kullanıcı oranının da %12 olduğu belirtilmektedir.

İnternetin yaygınlaşması ve web teknolojilerinin gelişmesi bireysel kullanımdaki artışa ek olarak birçok iş kolunu dijital dönüşüme zorlamıştır. Bu dönüşüm iş dünyasının bir çok açıdan işlerini kolaylaştırmış ve verimliliğini arttırmıştır. Ancak yaşanan ve katlanarak artan bu değişim ile yeni sorunlar ortaya çıkmaya başlamıştır. Web'in artan yükü geleneksel yöntemler ile bu dönüşüme cevap veremez duruma gelmiştir. Bu sorunlar neticesinde ağ ve depolama teknolojilerinde farklı atılımlar yapmak birer zorunluluk halini almıştır. Bu amaca yönelik olarak dünyanın farklı üniversitelerinde ve araştırma merkezlerinde yeni teknolojiler üzerinde çalışmalar başlatılmış ve araştırma bulguları literatüre kazandırılmıştır.

Ağ ve depolama teknolojilerindeki performans kaybını önlemek için önceleri *Proxy* sunucular kullanılmaya başlanmıştır. Bu sunucular temelde paylaşılan önbelleklere sahip sunuculardır. Bu paylaşılan ön bellek sayesinde istemcilere öncelikle buradan cevap verilmesi yoluna gidilmiştir. Daha sonra, özellikle 1998 yılında Web'in yukarıda bahsedilen dezavantajlarını gidermek için içerik dağıtım ağları (*Content Delivery Network)* kısaca CDN'ler ortaya çıkmıştır. Bu içerik dağıtım ağları farklı coğrafi bölgelere kurularak kullanıcıların veriye erişim süresini azaltmayı hedeflemiş ve mekanik sorunlar oluşması durumunda karşılaşılacak kayıplara karşı veri güvenliğini sağlamıştır (Hosanagar 2004).

CDN'lerin gelişimi sonrası teknoloji ilerlemeye devam etmiştir. Gelişen ağ altyapısı ve bağlantı hızları yeni bir kavram olan bulut bilişimin yaygınlaşmasına ön ayak olmuştur. Bulut bilişim aslında CDN'leri de kapsayan çok geniş bir kavramdır. Sunduğu altyapı ve mimari, donanım ve yazılımın ayrı ayrı ve birlikte farklı senaryolarda hizmetler sunmasına imkan sağlamıştır. Reel sektör yavaş yavaş buradaki fırsatları keşfetmeye başlamış ve CDN'ler sonrası yeniden bir dönüşüm süreci başlamıştır.

Hem CDN hem de bulut teknolojileri için olmazsa olmaz koşul Web'deki içeriğin ölçeklenebilir ve güvenilir bir şekilde sunulmasıdır. CDN'nin ilk yıllarında veriler tek bir internet servis sağlayıcı (ISS) üzerindeki dağıtık kümeler üzerinde tutulmuştur. Bu durum beklenen ölçeklemeyi sağlamıştır. Öte yandan tüm verilerin tek bir ISS alt yapısı

aracılığıyla sağlanması oluşabilecek herhangi bir ağ hatasında sunucuların tamamen ulaşılamaz olmasına neden olmuştur. Bu nedenle bu sunucuları farklı ISS ve fiziksel ağlara dağıtmak yoluyla bu sorunun çözülmesi yoluna gidilmiştir.

Verinin farklı fiziksel konumlardaki sunuculara dağıtılması için öncelikle bu coğrafi konumların belirlenmesi gerekmektedir. Bu *NP-Hard* bir problemdir (Salahuddin *et al.* 2017). Replika ya da çoğaltma sunucuları olarak adlandırılan bu sunucuların yerleşimi başlı başına bir optimizasyon problemidir. Bu nedenle literatürde birçok yaklaşım sezgisel olarak kurgulanmıştır. Özellikle CDN'ler için çalışılmış, denenmiş ve de uygulanmış birçok algoritma bulunmaktadır. Yakın zamanda daha da yaygınlaşmaya başlayan bulut tabanlı CDN'ler için ise farklı yaklaşımlar gerektiğinden bu alanda yeni çalışmalar yürütülmektedir.

İlk başlarda giderek artan sayıdaki web sitesi verimini arttırmak için ayna sunucular (*Mirror Server*) ortaya konulmuştur ve bir web sitesine gelen bir isteğin cevaplanacağı durumda hangi ayna sunucunun en iyi performansı sunacağını belirlemek çözülmesi gereken bir sorun olarak ortaya çıkmıştır (Myers *et al.* 1999). Dilley vd. (2002) yılında ilk ticari CDN sağlayıcılardan olan *Akamai* de ortaya çıkan aşırı yüklenme durumlarında bir sitenin çökebildiğini veya olağandışı şekilde yüksek tepki sürelerine neden olabildiğini, bu durumların her ikisinin de gelir kaybına veya bir ürüne veya markaya yönelik olumsuz bir müşteri tutumuna dönüşmesine neden olduğunu tespit etmiştir. Bunun üzerine web içeriğini tek bir konumdan sunmanın site ölçeklenebilirliği, güvenilirliği ve performansı için ciddi sorunlar doğurabileceği gözlemine dayanarak vekil sunucular ve bunların dağılımı üzerine çeşitli çalışmalar yapmışlardır.

"Hizmet tesislerinin sayısını ve yerini dağıtılmış ve ölçeklenebilir bir şekilde nasıl belirleyebiliriz?" sorusuna odaklanan Laoutaris (2007) hizmet tesislerinin yani replika sunucuların büyük ölçekli ağlara yerleştirilmesi sorunu için dağıtılmış bir yaklaşım tanımlamışlardır. Sunucuların hizmet verecekleri konumları ve tesis sayılarını birkaç adımdan oluşan ve iyileştirilmiş olan yerel optimizasyonlarla yeniden optimize ederek klasik merkezi yaklaşımların ölçeklenebilirlik sınırlamalarının üstesinden gelmişlerdir.

Literatürde daha önce *Greedy User* (GU), *Greedy Site* (GS) gibi kümeleme algoritmalarından yola çıkarak replika sunucuları kademeli olarak yerleştiren algoritmalar geliştirilmiştir (Chen *et al.*2012). Diğer bir çalışmada özellikle replika sunucusundan geçen en kısa yol sayısının toplam en kısa yol sayısına oranı olan SPBC (En Kısa Yol Arasındaki Kısa Yol) metriği temel alınarak öncelikli yerleşim sağlayan algoritmalar denenmiştir (Papagianni *et al.*2013).

Replika sunucu yerleşimi problemi için ilk defa deterministik sabit yaklaşım algoritmasını Rappaport and Raz (2013) sunmuş, kapsamlı simülasyonlar ve gerçekçi veri merkezi ve ağ topolojisi kullanarak algoritmalarının pratikte iyi yerleştirme kararları verdiğini göstermişlerdir. Diğer yandan R. Cohen *et al.* (2015) Ağ İşlev Sanallaştırması (NFV) adında yeni bir algoritma geliştirmişlerdir. Bu algoritma ağ işlevlerinin yürütüldüğü yeni bir ağ paradigması ve bu ağ üzerindeki kullanıcı sayısının oluşturduğu yüke bağlı performans değişimlerini öngören bir mantıkla çalışmaktadır.

Bulut teknolojisinin yaygınlaşması sonrasında, Papagianni et al. (2013) hiyerarşik bir çerçeve önermişlerdir. Bu kapsamda bulut içi iletişim ve bulut içi iletişim kaynaklarının geleneksel bulut bilişim kaynaklarıyla eşzamanlı olarak ele alındığı, ayrıca ölçeklenebilir bir içerik dağıtım çözümüne yönelik olarak çeşitli çalışmalar ortaya koymuşlardır.

CCDN'lerin sahip oldukları dinamik doğa nedeniyle ortaya çıkan yeni problemler çıkmıştır. Bu amaçla Hu et al. (2014) tarafından bulut tabanlı CDN'ler için dinamik talep modellerini ele alma vurgusuyla ortak kaynak sağlama ve önbellekleme (yani replika yerleştirme) sorununu çözmek için bir dizi yeni algoritma önerilmiştir.

Bu tez çalışmasında bu problemin çözümüne yönelik olarak daha önce geliştirilen sezgisel yaklaşımlardan farklı olarak metasezgisel bir yöntem olan Tabu Arama algoritmasından faydalanılmıştır. Bildiğimiz kadarıyla bir metasezgiselin bu problem üzerinde uygulandığı ilk çalışma da bu tezdir. Geliştirilen metasezgiselin verimlilik ve performans kriterlerini ne ölçüde sağlayabildiği öncelikle araştırılmış ve sonrasında

araştırma bulguları Sahoo ve Glitho (2016) ile verilen sezgisel algoritmanın sonuçları ile karşılaştırılmıştır. Literatürde bu konu üzerinde yapılan çalışmalarda kullanılan ve sonrasında paylaşılan standart bir denektaşı bulunmamaktadır. Bu ise yürütülen çalışmaları mukayese edebilme imkânını ortadan kaldırmaktadır. Bu nedenle bu tez kapsamında farklı büyüklüklerde 12 adet denektaşı içeren bir denektaşı seti oluşturulmuş ve deneysel sonuçlar bu set kullanılarak elde edilmiştir. Bu setin literatürdeki önemli bir boşluğu dolduracağı ve çalışmalara yeni bir yön kazandıracağı değerlendirilmektedir.

Şekil 1.1. Çalışmanın uygulama adımları

Şekil 1.1'de gösterildiği gibi geliştirilecek uygulamanın analizi ve testleri için öncelikli olarak kullanıcı ve bulutların bulunduğu konumlar ve bu konumlar arasındaki uzaklıkların yer aldığı test verileri, yani denektaşları hazırlanacaktır. Ardından uygulama içerisinde denektaşı verilerinin nasıl temsil edileceğine yanıt aranacak, başka bir deyişle ilgili veri yapıları oluşturulacaktır. Daha sonra Sahoo ve Glitho (2016) tarafından tasarlanan sezgisel yöntem gerçeklenecektir. Bu aşamanın son adımında önerdiğimiz Tabu Arama algoritması hayata geçirilicektir. ardından gerçeklenecek metasezgisel

algoritma yani Tabu Arama algoritması (TA) tamamlanıp karşılaştırma ve analizleri yapılıcaktır. En son adımda elde edilen sonuçlar karşılaştırılacaktır.

Bu tezin geri kalan kısmı şu şekilde organize edilmiştir. 2. Bölüm'de kuramsal temellerden bahsedilecektir. 3. Bölüm'de materyal ve yöntem anlatılacaktır. Araştırma bulguları 4. Bölüm'de tartışılacaktır. 5. Bölüm ise tezin sonuç bölümüdür.

2. KURAMSAL TEMELLER

Kullanıcıların web deneyiminde gecikme çok önemli bir kriterdir. Veri alışverişi kullanıcı ile veri merkezi arasında olmaktadır. Bu alışverişte gecikme çok önemlidir. Bu gecikme her kullanıcı için makul seviyelerde olmalıdır. Bu yüzden web siteleri birden fazla sunucu üzerinden kullanıcılarına hizmet vermektedirler. Kullanıcılar da bu hizmeti veri isteklerinin gidiş dönüş gecikmesi (RTT) önceden belirlenmiş kısıtlar dahilinde olduğu sürece farklı sunuculardan alabilmektedir. Replika sunucu yerleşimi probleminde asıl hedef sunucuların bant genişliği maliyetini minimize etmektir. Bu yönüyle bu problem aslında bir optimizasyon problemidir.

Optimizasyon kavramı aşağıdaki örneklerde olduğu gibi çeşitli şekillerde tanımlanmıştır:

- Optimizasyon, en genel anlamıyla, bir sistemde, belirli kısıtlar altında, bir amaç fonksiyonunun değerinin en iyilenmesi amacıyla karar değişkenlerinin alacağı değerleri belirleme işlemidir (Küçükkoç 2019).
- Optimizasyon, belirli bir hedef fonksiyonunda, eldeki değişkenleri ayarlayarak minimize yada maksimize etmek, bunları gerçekleştirirken bazı kısıtlamaları kullanmak ve optimum çözüm için çeşitli algoritmalar üretmektir (Cayıroğlu 2019).

Optimizasyon işleminde kullanılan teknikler analitik, sezgisel ve metasezgisel olarak üç kısımdan oluşmaktadır. Bunlardan analitik teknikler, matematiksel olarak modellenebilen ve boyutları çok büyük olmayan problemlerde sonuç üretebilmektedir. Sezgisel teknikler de probleme özgü çözüm yöntemleridir. Optimum çözümü garanti etmezler ancak optimuma daha yakın çözümleri analitik tekniklere göre daha kısa sürede üretebilirler. Metasezgisel teknikler ise çerçevesi ve yapısı belli olan algoritmaların, bir probleme uyarlanması ile problemi çözmeyi hedeflemektedir (Küçükkoç 2019).

2.1. CDN Nedir?

Ülkelerin internet kullanımına yönelik altyapı yatırımlarını arttırmasıyla internet hızlarında büyük bir iyileşme yaşanmıştır. Bunun sonucunda internet hızla yaygınlaşmaya başlamıştır. Bu yaygınlaşma ile birlikte bir çok iş kolu da dijitalleşme sürecine girmiştir. Yaşanan bu hızlı dönüşüm ise webdeki yükün artmasına neden olmuştur. Web performansını koruyabilmek için önceleri *Proxy* sunucular kullanılmaya başlanmıştır. Bu sunucular aslında paylaşılan bir önbellek yapısına sahiptir

birden çok istemci bu ortak havuz üzerinden hizmet almaktadır. Yani bu önbellek yapısında istenen bir verinin kopyası bulunuyorsa, istemci yani kullanıcı ihtiyaç duyduğu veriyi bu yapı üzerinden temin etmektedir. Böylelikle veri alışverişi arasındaki gecikme azaltılmış olmaktadır (Vakali ve Pallis, 2003).

Proxy sunucuların yaptığı ön bellekleme temelde 3 fayda sağlamaktadır:

- Sunucular coğrafi olarak kullanıcıya yaklaşmaktadır. Böylelikle:
  - Ağ iletişim süresi,
  - Bant genişliği kullanımı,
  - Ağ trafiği azalmaktadır.
- Proxy sunucularda bulunan veri, ana sunucudan daha önce transfer edilip önbelleklendiği için dış gecikme (*External latency)* giderilmektedir.
- Güvenilirlik arttırılmaktadır. Proxy sunucular önbelleklerinde veriyi sakladıkları için ana sunucuya erişim kaybolsa bile proxy sunucu üzerinden veri ulaşılabilir kalmaktadır.

1998 yılına kadar web kısa mesafelerde sağlıklı çalışıyor ve veriye zengin içerikler hariç makul bir gecikme ile ulaşılabiliyordu. Öte yandan bu dönemde web teknolojileri uzun mesafelerde büyük miktarda veriyi iletecek şekilde tasarlanmamıştı. Bu dezavantajı gidermek amacıyla CDN'lerin ortaya çıkışı da aynı döneme denk gelmektedir (Douglis ve Kaashoek, 2001).

2.1.1. Tek ISP yaklaşımı

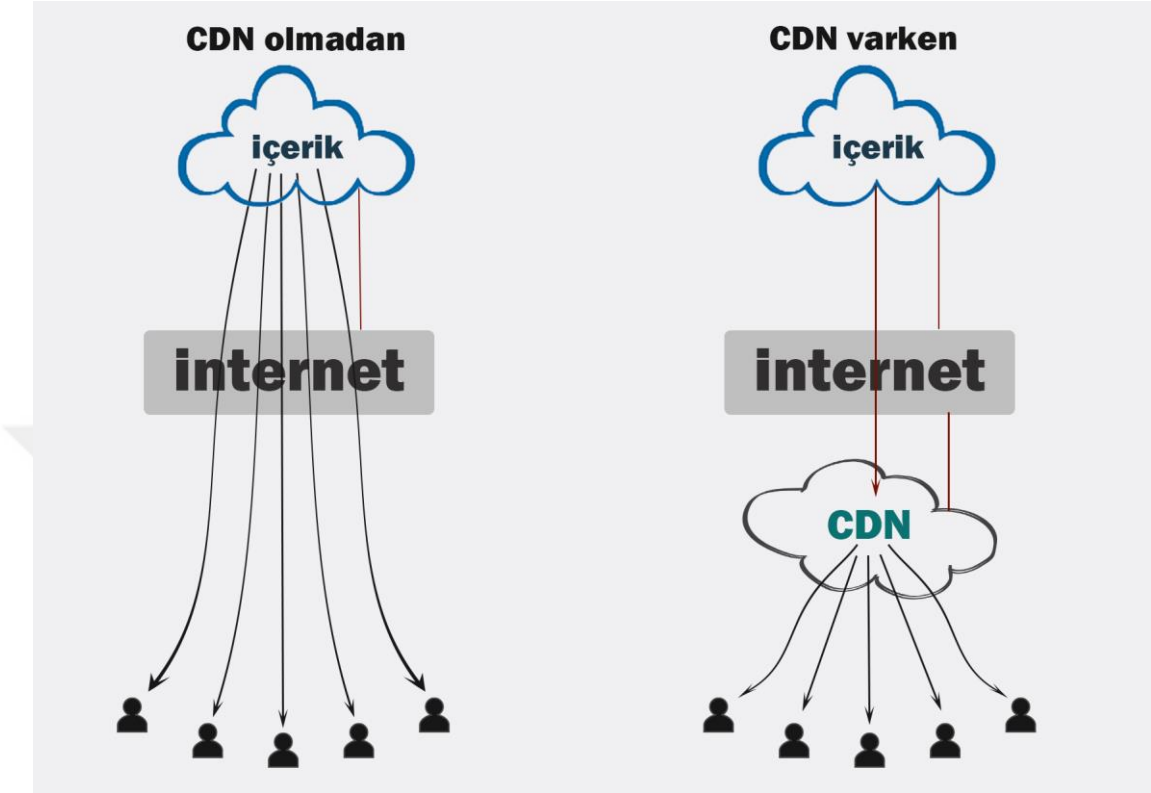
Douglis ve Kaashoek (2001), en az 40 tane replika sunucunun kendi network uçlarına dağıtıldığı bir yapı tasarlamışlardır. Böylelikle tek bir internet servis sağlayıcısı başka bir internet servis sağlayıcısına bağımlı olmadan yeterli bir küresel kapsama sağlayabilmekteydi. Bunun en yaygın kullanımı her büyük ülkede bir veya iki yerde vekil sunucuların dağıtımı şeklinde olmaktaydı. Bu yapıyı kullanan ISP'ler genellikle çok büyük alana sahip önbellek yapılarına sahiptirler. Böylelikle *hit ratio* denilen ve gelen isteğin anında karşılanması anlamındaki isabet oranı yüksek olmaktadır. Fakat bu yaklaşımın şöyle bir dezavantajı görülmüştür. Az sayıda kullanılan vekil sunucular coğrafi mesafe olarak içerik dağıtım ağlarına (CDN'lere) uzak kalabilmektedirler (Vakali and Pallis, 2003).

2.1.2. Çoklu ISP yaklaşımı

Çoklu ISP yaklaşımında birçok vekil sunucuya mümkün olduğunca çok küresel ISP noktasında yer bulunurak yerleştirilme yapılmaktadır (Vakali and Pallis, 2003). Buradaki amaç kullanıcılara en yakın sunucuları atamak ve içeriği hızlı ve güvenilir bir şekilde kullanıcıya iletmektir. Bu çoklu yapının en önemli avantajı kullanıcının bağlı olduğu internet servis sağlayıcısından içerik alabilmesidir.

Bu çoklu yapının şöyle bir dezavantajı oluşabilmektedir. Bazı büyük dağıtılmış sistemler 8000'den fazla sunucuya sahiptirler (Douglis and Kaashoek, 2001). Böyle yapılarda oluşturulan her vekil sunucu daha az hit alabilir ve bu CDN performansı açısından kötüdür (Vakali and Pallis, 2003). Aynı zamanda bu büyüklükteki çoklu ağlar daha kompleks yapıda olup bakım maliyetleri, tekli ISS yaklaşımına göre çok daha yüksektir.

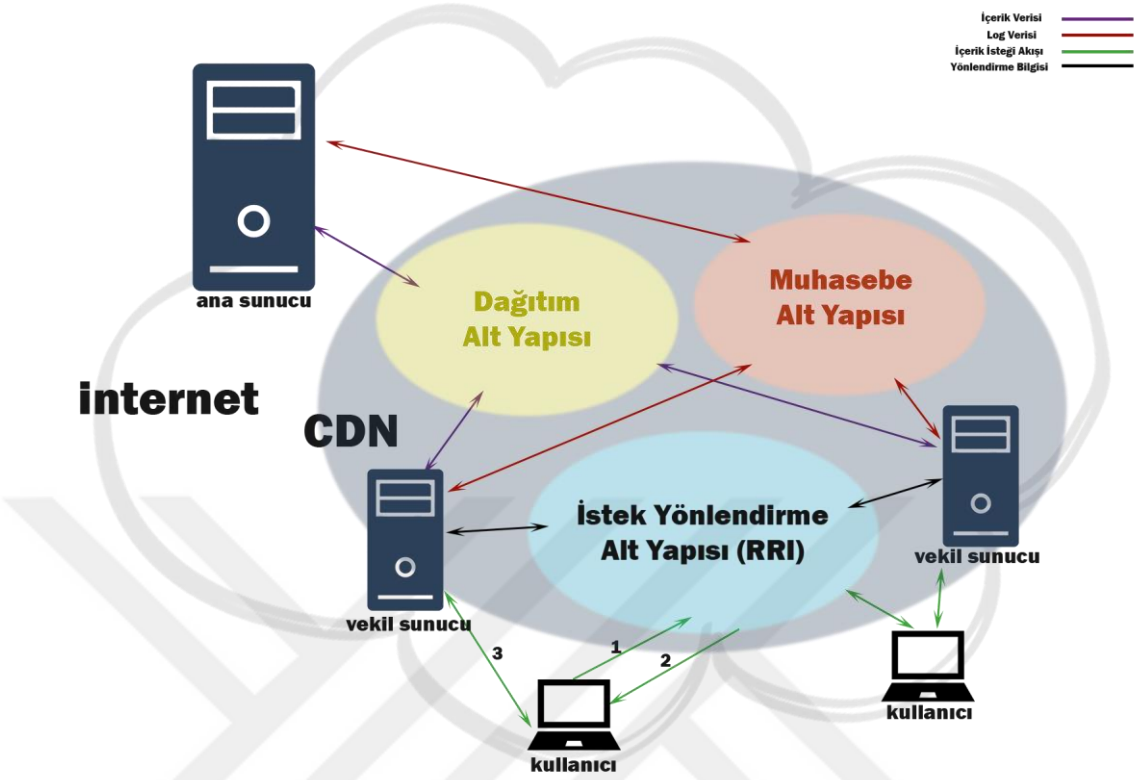
Douglis ve Kaashoek (2001), farklı trafik hacimleri altındaki performans değerlerinde Tek-ISS yaklaşımının düşük ve orta yoğunluklu trafik hacimleri olan siteler için daha verimli olduğunu, Çok-ISS yaklaşımının ise yüksek trafik hacmine sahip sitelerde daha iyi çalıştığını belirtmişlerdir.



Şekil 2.1. CDN temsili gösterim

2.1.3. CDN yönlendirme

Webdeki veriler belirli zaman aralıkları boyunca önbellekte tutulmaktadırlar. Bunun için bu verilere belirli yaşam süreleri atanır. Veriler bu süre boyunca önbellek sunucularında saklanmaktadır. Gelen isteğe göre uygun vekil sunucuya istek yönlendirilir. Şekil 2.1. de kabaca gösterilen CDN'lerde genel olarak CDN sunucuları vekil sunucu yönlendirmesini yapabilmek için DNS yani Alan Adı Sistemini kullanmaktadırlar (Vakali and Pallis, 2003).



Şekil 2.2. CDN yönlendirme temsili

DNS aracılığıyla CDN yönlendirme Şekil 2.2.'de görüldüğü üzere şöyle özetlenebilir.

1- Kullanıcı isteğini yerel DNS sunucusuna, bu yerel DNS sunucusu da ilgili isteği CDN'nin istek yönlendirme alt yapısına gönderir. (RRI: Request Routing Infrastructure)

2- RRI tüm vekil sunucuları kontrol eder ve gelen isteğin varsa cevap rotasını diğer vekil sunuculardan ister.

3- Tüm vekil sunucular kendi sonuçlarını hem yerel DNS sunucusuna hem de yerel sunucunun bulunduğu RRI'ya gönderir.

4- RRI gelen tüm sonuçları karşılaştırır ve yerel DNS sunucusuna bir DNS yanıtı olarak en uygun vekil sunucuyu seçerek gönderir.

5- Yerel DNS sunucusu da bu kullanıcıya bu yanıtı ulaştırır.

2.1.4. CDN fiyatlandırma

CDN'ler gelen istekleri yönlendirme, ulaştırma ve takip etme gibi durumları izleyen muhasebe yapıları kullanmaktadırlar (Mobasher *et al.* 2000). Bu yapılar gerçek zamanda yapılan istek yönlendirme ve sunucu cevaplarını günlük dosyalarda tutmaktadır. Bu dosyalar verilerin analiz edilerek özniteliklerinin çıkarılması ve geliştirilmesi için kullanılırlar. Tüm bu işlemlerde kullanılan veri ortak bir veri formatıdır. Müşteriler webdeki içerikleri ile ilgili kayıtları detaylı bir şekilde görmek isterler. Bunun için CDN sağlayıcıları müşteri bazlı bireysel sunucu günlükleri tutup bu bilgileri müşterileriyle paylaşırlar. Tüm bu logların çıkarılıp hazırlanması önemli bir veri akışını içermektedir ve bu işlem zaman alabilen bir işlemdir.

Müşterilere kesilecek faturanın hesaplanması için her ay toplanan milyonlarca sunucu loglarının kısa bir özete indirgenmesi gerekmektedir. Bunu yapacak olan alt yapının da ölçeklenebilir olması gerekmektedir. Çünkü sunucu ve müşteri sayısı artıkça işlem yapılması gereken hem günlük bilgi miktarı artmakta hem de bazı müşteriler bu verileri anlık görmek istemektedir. Faturalandırma bilgilerini sağlamak için işte bu veriler analiz edilmektedir. Bu raporlar, yani mevcut kullanım ve maliyet analizleri incelenerek elde edilen bilgiler ile CDN'nin iletişimi ve trafik maliyetleri azaltılabilmektedir.

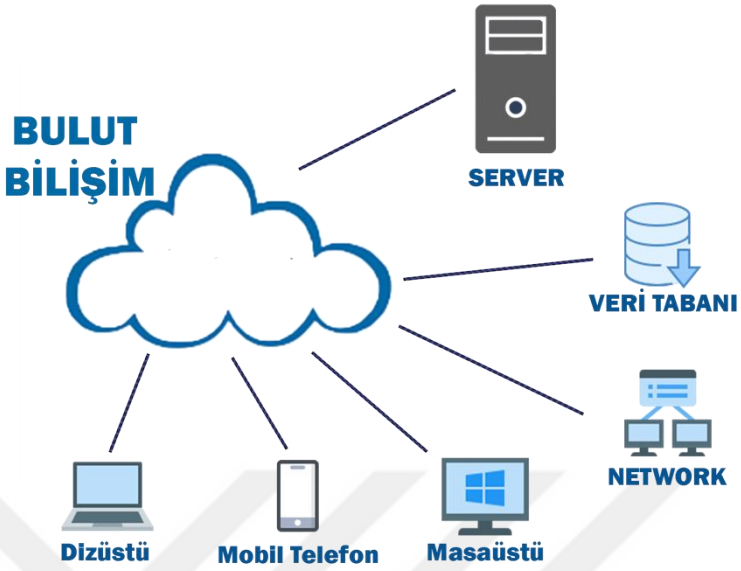
Müşteri açısından en önemli CDN kriterleri maliyet, performans ve ulaşılabilirliktir. Müşterinin trafik hacmi arttıkça maliyet daha ekonomik hale gelmektedir. Bu yüzden ilk başta CDN kullanmak, başlangıçta kullanmamaya göre maliyetli görülebilmektedir (Douglis and Kaashoek, 2001). CDN sağlayıcıları ücretleri genellikle gigabayt birim fiyat üzerinden hesaplamaktadır. CDN'ye gelen kullanıcı isteği bazen Şekil 2..2'de gösterilen RRI tarafından vekil sunucularında bulunamayabilir. Bu tür bir durumda eğer bu CDN sunucusunun başka CDN'ler ile anlaşması var ise o anlaşma çerçevesinde kendine eş CDN'lerden gelen isteği talep edebilmektedir. Bu işleme *Peering* adı verilmekte ve bu CDN sağlayıcı için ekstra maliyet anlamına gelmektedir.

2.2. Bulut Nedir ?

Geleneksel CDN'ler web trafiğini desteklemede başarılıdırlar, ancak mali yönden önemli yatırımlar ve dağıtım çalışmaları gerektirirler ve büyük ölçüde tek bir uygulamaya özeldirler. Bu bağlamda ilk kurulum maliyetleri ve yeni bölge çoğaltmaları ciddi yatırımlar gerektirmektedir. Bu maliyetleri oldukça hafifletecek yeni mimariler geliştirilmiştir ve yeni bir paradigma ortaya çıkmıştır.

Bulut bilişim kavramı aşağıdaki örneklerde olduğu gibi çeşitli şekillerde tanımlanmıştır.

- "Bulut bilişim; sunuculara, depolama alanlarına, veri tabanlarına ve çok sayıda uygulama hizmetine internet üzerinden erişmenizi sağlar. Bu uygulama hizmetleri için gereken ağ bağlantılı donanımın mülkiyeti ve bakımı bulut hizmetleri sağlayıcısı tarafından üstlenilirken, kullanıcılar bir web uygulaması aracılığıyla ihtiyacı olan kaynakları tedarik edip kullanabilir." (Anonim, 2019a).
- "Basitçe açıklamak gerekirse, bulut bilişim bilgi işlem hizmetlerinin (sunucu, depolama, veritabanı, ağ, yazılım, analiz ve makine zekâsı dahil) İnternet ("bulut") üzerinden sağlanarak daha hızlı inovasyon, esnek kaynaklar ve ekonomik ölçeklendirme sunulması anlamına gelir. Normalde yalnızca kullandığınız bulut hizmetleri için ödeme yaptığınızdan işletim maliyetlerinizi düşürebilir, altyapınızı daha verimli bir şekilde çalıştırabilir ve değişen iş gereksinimlerinize uygun şekilde ölçeklendirme yapabilirsiniz." (Anonim, 2019b).
- Başka bir ifade ile bulut bilişim; işlem gücünün, veri tabanının, depolama alanının, uygulamaların ve diğer kaynaklarının kullandıkça öde modeliyle internet üzerinden isteğe bağlı olarak sunulmasıdır.



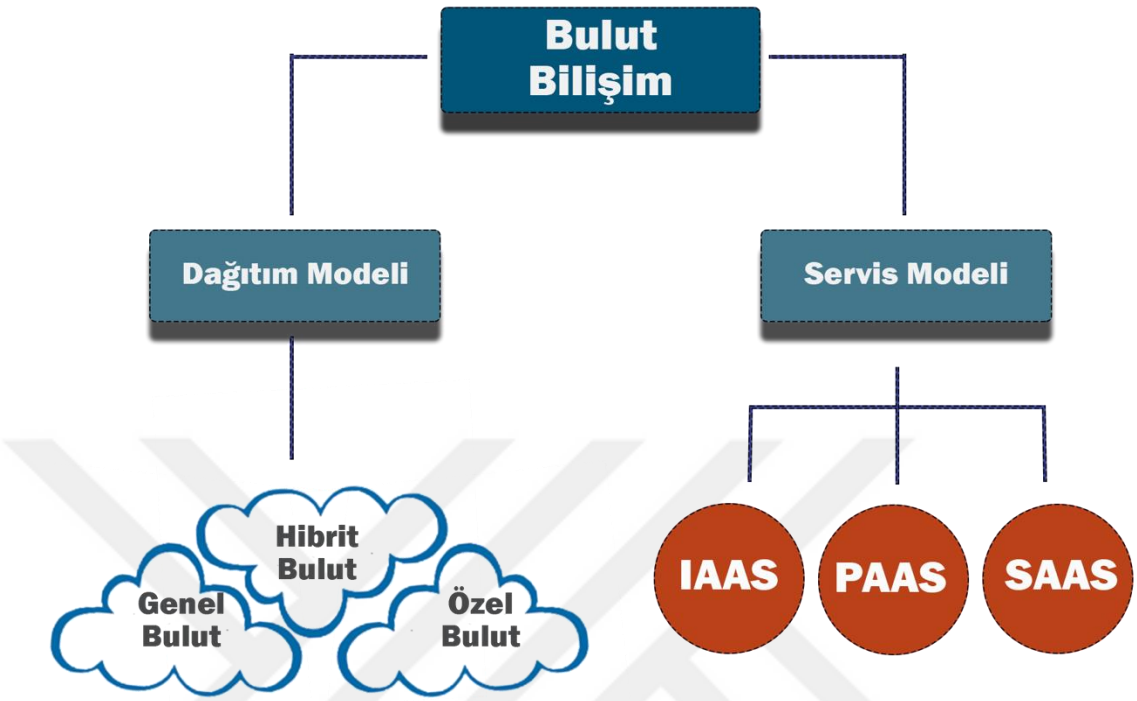
Şekil 2.3. Bulut bilişim kapsamı

Kurum ve kuruluşların bulut bilişim hizmetlerine ilgi göstermesini sağlayan altı temel neden aşağıda açıklanmaktadır: (Anonim, 2019b)

- Maliyet
    - Bulut bilişim ile donanım ve yazılım alma, veri merkezi kurma gibi yatırım giderleri ortadan kalkmıştır. Özellikle veri merkezlerindeki sunucu rafları, sunucuları soğutmak için harcanan enerji, tüm altyapının yönetilmesi için gereken uzman personel ciddi maliyetlerdir.
- Hız
    - Bulut bilgi işlem hizmetleri isteğe bağlı olarak sunulduğu için istenilen işlem kaynakları ve gücü birkaç tıklama ile dakikalar içinde elde edilebilir. Böylelikle kurumlar başlangıç için ve ileriye dönük kapasite hesaplamaları yapmak zorunda kalmamaktadırlar.
- Küresel Ölçek ve Esneklik
    - İhtiyaç duyulduğu anda istenilen miktarda bilgi teknolojisi kaynağına anında ulaşım sağlanabilir. Örneğin daha fazla işlem gücü, depolama veya bant genişliğinin istenildiği anda istenen coğrafi konumda sunulabilmesi gibi…

- Verimlilik
    - Kurumların kendi içinde kullandıkları veri merkezlerinin devamlılığı için yapılan donanım ayarlamaları ve yazılım güncellemeleri zaman alan işlemlerdir. Bilgi teknoloji (BT) ekipleri zamanlarının bir kısmını burada harcamak durumundadırlar. Bulut bilişim ile bu görevlerin çoğu ortadan kalkmış ve BT ekiplerinin zamanlarını daha önemli işlerle ve verimli olarak kullanması sağlanmış olmaktadır.
- Performans
    - Bulut bilgi işlem hizmetleri veren kurumlar dünya çapında güvenli veri merkezleri ağında çalışmaktadırlar. Burada her zaman en güncel donanımlar kullanılmaktadır. Bu da her zaman daha düşük ağ gecikmesi ve ölçeklendirme maliyeti anlamına gelmektedir.
- Güvenilirlik
    - Verilerin bulut bilgi işlem ağında yedekli olarak birden fazla konumda tutulmasından dolayı veri yedekleme ve olağanüstü durum kurtarma işlemleri daha güvenli ve ekonomiktir.
- Güvenlik
    - Bulut hizmet sağlayıcıları kullanıcı verilerini, uygulamalarını ve alt yapılarını çeşitli teknolojiler, denetimler ve yöntemler ile olası tehditlere karşı maksimum korumayı sağlamaktadır.

Bulut hizmetleri Şekil 2.4.'te de görüldüğü üzere kullanım alanlarına göre üç farklı tipte ve üç farklı servis modeli şeklinde olmaktadır.



Şekil 2.4. Bulut hizmetinde dağıtım ve servis modeli türleri

2.2.1. Genel bulut

Bir bulut, halka açık bir şekilde kullanıma sunulduğunda buna Genel Bulut denilmektedir. Sunucular ve depolama gibi bulut kaynakları, üçüncü taraf bulut hizmeti sağlayıcıları tarafından sağlanmıştır. Bu hizmet sağlayıcıları, sunucu ve depolama gibi bilgi işlem kaynaklarını İnternet üzerinden sunmaktadır. Bu bulutu kullanan kullanıcılar veya "kiracılar" aynı donanım, depolama ve ağ cihazlarını paylaşarak kullanmaktadır (Anonim, 2019b).

Microsoft Azure, genel buluta bir örnektir. Genel bulutta tüm donanım, yazılım ve diğer destekleyici altyapı bulut sağlayıcısına aittir ve bu sağlayıcı tarafından yönetilir. Bu hizmetlere erişmek ve hesabınızı yönetmek için bir web tarayıcısı kullanmanız gerekmektedir. Genel bulutlara örnek olarak Amazon Web Servisleri, Google AppEngine ve Microsoft Azure verilebilir.

Bu yapıda yazılım veya donanım satın almaya gerek olmadığından maliyet düşüktür. Yalnızca kullanılan hizmet için ödeme yapılmaktadır. Bakım işlemlerini hizmet sağlayıcı yaptığı için bakım gerekmemektedir. Kullanıcı işletme ihtiyaçlarını karşılamak için isteğe bağlı olarak kaynak tahsis edilmektedir. Böylelikle sınırsıza yakın ölçeklenebilirlik sağlanmaktadır. Büyük ölçekli sunucu ağı sayesinde kullanıcılar arızalardan en az oranda etkilenmekte ve böylece yüksek güvenilirlik sağlanmaktadır.

2.2.2. Özel bulut

Bir kurum, kuruluş veya işletmenin herkese açık olmayan, yalnızca kendi içerisinde kullanılan veri merkezlerini ifade etmek için kullanılır. Bu veri merkezleri fiziksel olarak kurum içinde olabileceği gibi kurum dışında üçüncü parti bir hizmet sağlayıcıda da bulunabilir. Buralarda bulunan donanım ve yazılımlar tamamen özel bir ağda yer alır ve bu kuruma ayrılmıştır. Bu ayrıcalıklı ve özel yapı kurum ihtiyaçlarına göre bilgi teknolojileri gereksinimlerini özelleştirebilmeyi kolaylaştırır (Anonim, 2019h).

Özel bulutlar çoğu zaman devlet kurumları, finans kuruluşları ve iş açısından kritik operasyonlara sahip diğer orta-büyük ölçekli kuruluşlar tarafından kullanılır.

Özel bulutların avantajları:

- **Daha fazla esneklik:** Kurumlar belirli iş ihtiyaçlarını karşılamak için bulut ortamını özelleştirebilir.
- **Yüksek güvenlik:** Kaynaklar başkalarıyla paylaşılmadığından daha yüksek denetim ve güvenlik düzeyleri sağlanabilir.
- **Yüksek ölçeklenebilirlik:** Özel bulutlar, genel bulutların ölçeklenebilirliğine ve verimliliğine sahiptir.

2.2.3. Hibrit bulut

Hem genel bulut yapısının hem de özel bulut yapısının birlikte kullanıldığı yapılardır. Buradaki en önemli fayda iki bulut türünün sağladığı avantajların bir arada elde edilmesidir. Örnek verecek olursak düşük güvenlik gereksinime sahip bir hizmet genel bulut üzerinden karşılanacaktır. İhtiyaç duyulduğunda bu veriler özel buluta da aktarılacaktır. Bu sayede fiyat ve performans maksimize edilmiş olacaktır. Diğer yandan kurumun finansal raporları gibi özel bilgilerin işlenmesi ve oluşturulması kurumun özel bulut alt yapısı üzerinden yapılacaktır. Böylelikle hibrit bulut sistemlerinde hem veri geçişleri güvenlik zafiyeti oluşturulmadan sağlanacak, hem de maliyet azaltılacaktır (Anonim, 2019b).

Hibrit bulutların avantajları:

- **Denetim:** Kurumlar hassas varlıkları için özel bir altyapı kullanabilmektedir.
- **Esneklik:** İhtiyaç olduğunda genel buluttaki ek kaynaklardan yararlanılmaktadır.
- **Maliyet etkinliği:** Gerketiğinde genel bulut gibi ölçeklendirme yapılarak yalnızca ihtiyaç anında ek bilgi işlem gücü kullanılmakta, böylelikle tasarruf sağlanmaktadır.

2.2.4. Hizmet olarak altyapı (IaaS, infrastructure as a service)

Bulut bilgi işlem hizmetlerinin en temel kategorisidir. *IaaS*, bir bulut sağlayıcısından kullandıkça öde esasına dayalı olarak BT altyapısı (sunucular ve sanal makineler, depolama, ağ, işletim sistemleri) kiralanmasına olanak tanımaktadır. Böylelikle yatırım giderlerini ortadan kaldırılmakta ve maliyetler azaltılmaktadır. *IaaS*, şirket içi veri merkezi kurmak ve yönetmek için gereken masrafların peşin olarak ödenmesi gerekliliğini ortadan kaldırarak, yeni iş kuranlara veya yeni fikirler denemek isteyen işletmelere yönelik avantajlı bir seçenek sunmaktadır (Anonim, 2019b).

Bu altyapılar olağanüstü durumlarda oluşabilecek veri kayıplarına karşı gelişmiş veri kurtarma olanağı sağlarlar. Normal sunucularda kullanılabilirliğin ve sürekliliğin sağlanması ciddi teknoloji ve uzman personel gerektirdiğinden maliyetli bir iş olmaktadır.

Ancak *IaaS* kullanımında yapılan doğru bir hizmet seviyesi anlaşması (SLA) ile bu maliyetler minimize edilebilmektedir. Böylelikle olağanüstü bir durumda uygulama ve verilere her zamanki gibi erişim sağlanabilmektedir.

IaaS sayesinde planlanan yeni fikirler haftalar, hatta aylar süren bilgi işlem altyapısı kurulumunu beklemek zorunda kalmadan saatler içinde uygulanabilmektedir. Böylelikle yeni iş planlamalarında veya var olan bir işin büyütülmesi aşamalarında ciddi bir motivasyon ve verimlilik sağlanmaktadır. Normalden fazla işlem gücü gerektiren dönemsel yoğunluklar esnasında bu ihtiyaç anlık olarak birkaç tıklamayla elde edilebilmekte ve bu sayede önemli bir tasarruf sağlanmaktadır. Yoğun dönemlerin ardından kullanılan kaynakların ölçeği normal kullanım seviyesine anında indirilebilmektedir. Bu sayede uzman ekip altyapı yerine işin özüne odaklanabilecektir.

IaaS kullanımı ile donanım bakımı, yenileme ve yükseltme gibi temel altyapı sorunları ortadan kaldırılmış olmaktadır. Bu durum, servis sağlayıcı ve kullanıcı arasındaki SLA'lar ile garanti altına alınmıştır. Aynı zamanda veri güvenliği açısından da kurum içinde kurulacak bir veri merkezine göre çok daha iyi düzeyde güvenlik sağlanabilmektedir.

2.2.5. Hizmet olarak platform (PaaS, platform as a service)

Hizmet olarak platform, yazılım uygulamaları geliştirmek, test etmek, teslim etmek ve yönetmek için isteğe bağlı bir ortam sağlayan bulut bilgi işlem hizmetleri olarak tanımlanır. PaaS, geliştiricilerin web uygulamalarını veya mobil uygulamaları, geliştirme için gereken sunucular, depolama alanı, ağ ve veri tabanlarından oluşan temel altyapıyı kurma ve yönetme endişesi taşımadan hızla oluşturmasını kolaylaştırmak üzere tasarlanmıştır (Anonim, 2019b).

PaaS, web uygulaması yaşam döngüsünün tamamını sürdürmek üzere tasarlanmıştır. Yani oluşturmak için gereken alt yapı, test aşamaları, dağıtımı, güncellemesi gibi tüm süreç için hazır bir platform sağlanmaktadır. Kullanıcı geliştirdiği uygulamaları ve

hizmetleri yönetirken geriye kalan altyapısal tüm işlemler ve yazılım lisansları, geliştirme araçları ve iş zekâsı gibi kaynakların yönetimini bulut hizmet sağlayıcısı yönetmektedir.

2.2.6. Hizmet olarak yazılım (SaaS, software as a service)

Hizmet olarak yazılım, yazılım uygulamalarının internet üzerinden isteğe bağlı olarak ve genellikle abonelik yöntemi aracılığıyla dağıtılmasıdır. *SaaS* sayesinde bulut sağlayıcıları, yazılım uygulamalarını ve temel altyapıyı barındırıp yönetmenin yanı sıra yazılım yükseltmeleri, güvenlik ve hata düzeltme gibi bakım işlerini de üstlenmiştir. Kullanıcılar uygulamaları genelde telefon, tablet veya bilgisayarlarında bulunan bir web tarayıcısı ile İnternet bağlantısı sayesinde kullanabilmektedir (Anonim, 2019b).

SaaS uygulamalar hem hizmet sağlayan hem de son kullanıcı açısından son derece avantajlıdır. Çünkü hizmet sağlayıcı bulut altyapısındaki tüm imkânları kullanarak uygulamasını üyelik karşılığı kullanılacak şekilde konumlandırmaktadır ve bu yazılım bir çok kullanıcıya buradan dağıtılmaktadır. Böylelikle bir yazılımın yönetimi, güncelleştirilmesi ve sorunlarının giderilmesi son derece verimli hale gelmiş olmaktadır. Son kullanıcı açısından bakıldığında da kullanıcı bilgisayarına veya telefonuna herhangi bir uygulama kurmak zorunda kalmadan bu uygulamaları kullanabilmektedir. Verilerini bulutta depolayan kullanıcılar internete bağlı herhangi bir cihaz üzerinden bunlara ulaşabilmektedir. Böylelikle bilgisayarının bozulması veya kaybolması gibi durumlarda bir kullanıcı verilerini ve uygulamasını kaybetmemiş olacaktır. Verilerin bulutta tutulması ile *SaaS* hizmet sağlayıcıları iş modellerini mobilize etmiş olmaktadır. Bu da her yerden ulaşılabilir olma avantajını sağlamaktadır.

ERP ve *CRM* gibi karmaşık, mali yükü yüksek kurumsal uygulamalar daha düşük maliyetlerle daha ulaşılabilir olmaktadır. Böylelikle kendi başına bu yazılımları satın alacak ve yönetecek kaynaklara sahip olmayan kuruluşlar için çok büyük bir avantaj söz konusu olmaktadır.

2.2.7. Bulut tabanlı cdn çalışmaları

Gelişen teknoloji ve artan kullanıcı talepleri doğrultusunda firmaların bulut yatırımları artmış ve CDN sağlayıcılar da bu akıma ayak uydurmaya başlamıştır. Çünkü bulut teknolojilerinin en önemli avantajı altyapı maliyetlerini ortadan kaldırmasıdır. Böylelikle firmalar yatırım ve bakım maliyetinden kurtulmuş olmaktadır. Bulut hizmet sağlayıcıları farklı coğrafi konumlarda bulut altyapılarına sahiptirler. CDN sağlayıcıları da kendi replika sunucularını yerleştirmek için bulut altyapısını kullanmaktadır. Bu hizmeti alırken de bulut sağlayıcısının kullandıkça-öde (pay as you go) fiyatlandırma modelini kullanarak aslında bulut sağlayıcısının kaynaklarını kiralamaktadırlar (Sahoo *et al.* 2016).

İşte tam bu noktada CDN sağlayıcılar hem operasyonel maliyeti düşük bir düzeyde, hem de kullanıcı memnuniyetini belirli bir seviyede tutacak optimum bir sistem kurmak istemektedirler. Bu esnada karşılarına iki temel sorun çıkmaktadır: "Hangi bulut hizmet sağlayıcısını tercih etmeliyim?" ve "Coğrafi olarak sunucularım nerede olmalı?" Bu sorular bir NP-Hard problemin varlığına işaret etmektedir. Her ülkeye birden çok replika sunucu kurmak teknik olarak mümkün değildir. Bu yüzden belirli bölgeler tercih edilmeli ama bu tercih edilen bölgeler kullanıcı deneyimini olumsuz etkilemeyecek yeterlilikte olmalıdır. Kullanıcı ile sunucu arasındaki coğrafi uzaklık, RTT denilen -isteğin sunucuya ulaşması ve cevabın dönmesi arasındaki süre- gecikme süresinin artmasına yol açmaktadır. Bu istenmeyen bir durumdur. Genel kullanıcı portföyüne göre kabul edilebilir bir gecikme zamanını yakalayabilecek ve maliyeti de minimumda tutacak şekilde ihtiyaç duyulan ideal sayıda replika sunucusu doğru bulut hizmet sağlayıcısından kiralanmalıdır.

Bu sorunu çözmek için ilk başlarda kullanıcı memnuniyetini ikinci planda bırakan çözümler üretilmiştir. Bu çözümler, kurulum ve operasyon maliyetlerini minimum yapacak şekilde CDN sunucularını belirledikleri coğrafi konumlara yerleştirmişler ve ölçeklenebilirliği ve ulaşılabilirliği sağlamışlardır. Ancak bu CDN sağlayıcılarından hizmet alan kullanıcılar istenmeyen gecikme süreleri ile bu deneyimi yaşamak zorunda kalmışlardır. Bu çözümler QoS parametresini öncelik haline getirmedikleri için kullanıcı

taleplerinin artış gösterdiği video ve müzik akış servislerini sunmakta zorluk çekmişlerdir. Mokhtarian ve Jacobsen (2015) video sunucularının en uygun şekilde konuşlandırılması için katmanlı bir mimari oluşturmuşlardır. Bu mimari replika sunucu sayısını, kullanıcılar için gerekli bant genişliğini ve önbellek boyutunu içermektedir. Böylelikle depolama maliyeti ve bant genişliği maliyetini en aza indirmek hedeflenmektedir. Fakat QoS parametresi bu modelde kullanılmadığı için CCDN, yani bulut tabanlı içerik dağıtım ağları için uygun bir çözüm olamamıştır.

Bulut hizmet sağlayıcıları bir veri merkezi seçerken hizmet sunan firma ile bir anlaşma yapmaktadır. Bu anlaşma servis kullanıcısı için önemlidir. Çünkü hizmetin kalite standartları, teslim süresi, teknik destek ve yedek alma gibi sınırları bu anlaşma çerçevesinde çizilir. İşte bu anlaşmaya, SLA yani hizmet seviyesi anlaşması denir. Bu anlaşma tek tip olmayıp müşteri ihtiyaçlarına göre değişkenlik göstermektedir. Bu özelleştirme fiyatlarda da farklılık olmasına sebep olmaktadır (Anonim, 2019c).

Zhang (2013), replika sunucuları yerleştirme sorununda bölgesel elektrik dağıtım maliyetlerinde ortaya çıkabilecek fiyat değişikliklerine karşı dinamik fiyatlandırmanın kullanılacağı bir çözüm önermiştir. Böylece bulut hizmet sağlayıcıları kar marjlarında istikrar sağlamış olacaktır. QoS'si sağlamak için yeterli sayıda replika sunucusu kullanılacaktır. Diğer yandan farklı fiyatlandırma önerisi ile beraber bu problem, uygun sayıda replika sunucusu bulma ve kullanıcı SLA'larını karşılayacak şekilde uygun replika sunucularına atama şekline dönüşmüştür.

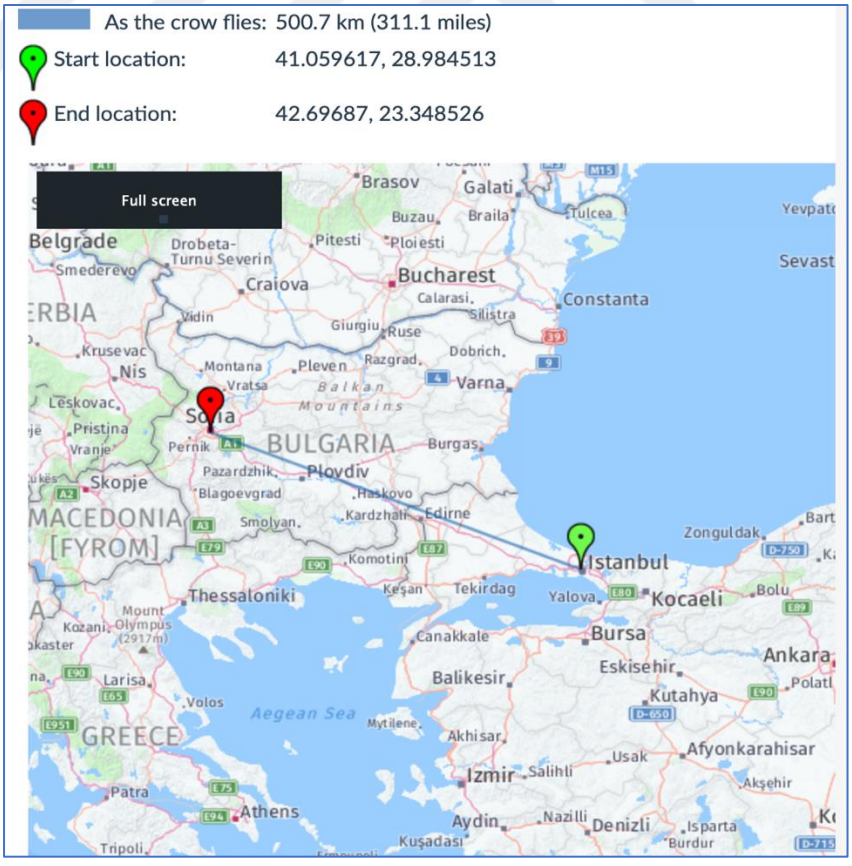
Bu alanda literatürde rastlanan tüm çözümler sezgiseldir. Sezgisel algoritmaların amacı ilgili problemi daha basit hale getirmek veya tatmin edici bir sonuç bulmaktır. Çünkü NP-Hard problemler brute-force yani kaba kuvvet yaklaşımı ile çözülmek istendiğinde çözüm günler, aylar hatta yıllar sürebilmektedir.
Metasezgisel algoritmalar ise belirli bir probleme özel olmayan, küçük değişiklikler ile farklı optimizasyon problemlerine uyarlanabilen genel algoritma çerçeveleridir. Bu algoritmalarda belirli bir kalite ölçüsünde aday çözümler iyileştirilmeye çalışılarak en iyi

sonuca ulaşılmaya çalışılmaktadır. Bu nedenle bu tür algoritmalara çözüm uzayının umut verici noktalarına yönelik yapılan sezgisel aramalar da denilebilir (Yaghini, 2009).

3. MATERYAL ve YÖNTEM

3.1. Materyal

İnternetin yaygınlaşması ve kullanım alanlarının gelişmesi ile birlikte Web'in artan iş yükünü azaltmak amacıyla yenilikçi çözümler ortaya çıkmaya başlamıştır. İlk başta *Proxy* sunucular, sonra *CDN'ler* ve en son olarak da bulut tabanlı *CDN'ler* çözüm olarak kullanılmaya başlanmıştır. Bu çalışmada gerçeklenen TA algoritmasını daha önce geliştirilen sezgisel algoritmalar ile karşılaştırabilmek için o algoritmalarda bir girdi olarak kullanılan gecikme parametresi aynen kullanılmıştır. Bu amaçla kullanıcının ve aday sunucunun coğrafi konumu arasında kuş uçuşu olarak hesaplanan fiziksel uzaklık bilgisinden faydalanılmıştır. Şekil 3.1 örnek olarak verilmiştir.



Şekil 3.1. İki konum arasındaki kuş uçuşu uzaklık hesabı

Fiziksel uzaklığın gecikme bilgisine dönüştürülmesi işlemi için ise Sahoo ve Glitho (2016) tarafından geliştirilen Formül 1'den faydalanılmıştır.

Gidiş Dönüş Süresi (RTT) hesabında kullanılan coğrafi uzaklık bilgisi kuş uçuşu uzaklık şeklinde (Anonim Doogal, 2019) sitesinden alınmıştır. Buradan elde edilen veri formülde yerine konularak gecikme süresi elde edilmiştir. Mesafeye dayalı gecikme hesabı için literatürde bu modelden sıklıkla faydalanılmaktadır.

$$RTT(i, j) = 0.02 * Dist(i, j) + 5 \quad \textbf{(1)}$$

Şekil 3.1. ile verilen örnek için RTT değeri aşağıdaki gibi hesaplanır:

$$\begin{aligned} RTT(\text{İstanbul, Sofia}) &= 0.02 * Dist(\text{İstanbul, Sofia}) + 5 \\ &= 0.02 * 500.7 + 5 \\ &= 15.014 \text{ ms} \end{aligned}$$

Yani İstanbul'daki bir kullanıcının Sofya'da bulunan bir sunucuya ortalama 15 ms'lik bir gecikme ile bağlanacağı kabülü yapılmaktadır.

3.1.2. Veri Seti Oluşturma

Algoritmaların sonuçlarının karşılaştırılmasında kullanılacak veri setleri hazırlanırken Avrupa öncelikli olarak üç büyük Bulut Hizmet sağlayıcısının sundukları bulut tabanlı içerik dağıtım servisleri, yani CCDN'ler baz alınmıştır. Bu şirketler Google, Amazon ve Microsoft şirketleridir. Bu şirketlerin Avrupa, Kuzey Amerika ve Asya'daki veri merkezlerinin konum bilgilerinden faydalanılmıştır. Kullanıcı konum bilgileri olarak ise Avrupa, Kuzey Amerika ve Asya'da bulunan ve o kıtayı coğrafi olarak temsil edebilecek şehirler seçilmiştir. Bu şehirler seçilirken Avrupa, Kuzey Amerika ve Asya coğrafyasını kapsayacak çeşitlilikteki konumlar tercih edilmiştir.

3.1.2.a. Google bulut servisleri

Google, dünya üzerindeki 90'dan fazla sunucusu ile kullanıcılarına her zaman kullanabilecekleri bir altyapı hizmeti sağlamaktadır. Aynı zamanda birçok *CDN* sağlayıcıdan farklı olarak konumlandırılacak site için tek bir *IP* adresi sağlamakta, böylelikle bölgesel *DNS* sunucularına ihtiyaç duymadan küresel bir performans elde edilmesini mümkün kılmaktadır. Google'ın sunduğu altyapı, son kullanıcı tarafında popüler olarak kullanılan Google arama ve Youtube gibi uygulamaların da dahili olarak kullandığı ortak altyapıdır. Bu uygulamalar önemli birer referans olmaktadırlar (Anonim, 2019e).

Google bulut sistemi fiyatlandırma politikasını altyapı, platform ve servis bazlı ayırmıştır. Ayrılan her birim de kendi içinde farklı parametreler kullanarak fiyat politikası geliştirmiştir.

Avrupa veri seti oluşturulurken Google'un Avrupa'da yerleşik ve aşağıdaki konumlarda bulunan sunucularından faydalanılmıştır.

- Amsterdam, Netherlands
- Budapest, Hungary
- Dublin, Ireland
- Frankfurt, Germany (3)
- Groningen, Netherlands
- Hamburg, Germany (2)
- Hamina, Finland
- London, England (3)
- Madrid, Spain (2)
- Marseille, France
- Milan, Italy
- Munich, Germany
- Paris, France (2)
- Prague, Czech Republic
- Sofia, Bulgaria
- St. Ghislain, Belgium
- Stockholm, Sweden (2)
- Warsaw, Poland
- Zurich, Switzerland (2)

Benzer şekilde Kuzey Amerika veri seti oluşturulurken aşağıdaki konumlarda bulunan sunucular kullanılmıştır.

- Ashburn, Virginia (2)
- Atlanta, Georgia (3)
- Charleston, South Carolina
- Chicago, Illinois (2)
- Council Bluffs, Iowa
- Dallas/Fort Worth, Texas (2)
- Denver, Colorado (2)
- Lenoir, North Carolina
- Los Angeles, California (3)
- Miami, Florida (2)
- Montréal, Québec (2)
- New York, New York
- Querétaro, Mexico (2)
- San Francisco, California (2)
- Seattle, Washington (2)
- The Dalles, Oregon
- Toronto, Ontario
- Tulsa, Oklahoma

Son olarak Asya veri seti oluşturulurken aşağıdaki konumlarda bulunan sunucular kullanılmıştır.

- Changhua County, Taiwan
- Chennai, India (2)
- Hong Kong (2)
- Kuala Lumpur, Malaysia (2)
- Mumbai, India (2)
- New Delhi, India
- Osaka, Japan (2)
- Singapore (3)
- Taipei, Taiwan (2)
- Tokyo, Japan (3)

3.1.2.b. Amazon bulut servisleri

Amazon, dünyanın farklı konumlarındaki sunucuları ile kullanıcılarına düşük gecikme ve yüksek aktarım hızları ile güvenli bir şekilde veri, video ve uygulamalar ulaştırdığı içerik dağıtım ağına *CloudFront* ismini vermiştir. Amazon omurga ağı içerisinde yer alan sayıları 100'den fazla uç sunucuyla kullanıcılara performanslı ve ulaşılabilir bir hizmet sunulmaktadır (Anonim, 2019f).

Amazon CloudFront, hem ağ hem de uygulama düzeyinde koruma sağlayan son derece güvenli bir CDN'dir. Web trafiği ve uygulamalar ek maliyet olmadan AWS Shield Standard gibi yerleşik koruma hizmetlerinden faydalanır. Ayrıca ek bir maliyet olmadan özel SSL sertifikaları oluşturmak ve yönetmek için AWS Certificate Manager (ACM) gibi pratik yapılar kullanılabilmektedir (Anonim 2019a).

Avrupa veri seti oluşturulurken aşağıdaki konumlarda bulunan Amazon sunucuları kullanılmıştır.

- Amsterdam, The Netherlands (2)
- Berlin, Germany (2)
- Copenhagen, Denmark
- Dublin, Ireland
- Frankfurt, Germany (8)
- Helsinki, Finland
- London, England (9)
- Madrid, Spain (2)
- Manchester, England
- Marseille, France
- Milan, Italy
- Munich, Germany (2)
- Oslo, Norway
- Palermo, Italy
- Paris, France (5)
- Prague, Czech Republic
- Stockholm, Sweden (3)
- Vienna, Austria
- Warsaw, Poland
- Zurich, Switzerland

Kuzey Amerika veri seti oluşturulurken aşağıdaki konumlarda yer alan Amazon sunucularından faydalanılmıştır.

- Ashburn, VA (6)
- Atlanta, GA (5)
- Boston, MA (2)
- Chicago, IL (7)
- Dallas/Fort Worth, TX (6)
- Denver, CO (2)
- Hayward, CA
- Hillsboro, OR
- Houston, TX (2)
- Jacksonville, FL
- Los Angeles, CA (5)
- Miami, FL (3)
- Minneapolis, MN
- Montreal, QC
- New York, NY (3)
- Newark, NJ (5)

- Palo Alto, CA
- Philadelphia, PA
- Phoenix, AZ
- San Jose, CA (2)
- Seattle, WA (3)
- South Bend, IN
- Toronto, ON

Son olarak Asya veri seti oluşturulurken aşağıdaki konumlarda yer alan Amazon sunucuları kullanılmıştır.

- Bangalore, India
- Chennai, India (2)
- Hong Kong, China (3)
- Hyderabad, India (2)
- Kuala Lumpur, Malaysia
- Mumbai, India (2)
- Manila, Philippines
- New Delhi, India (3)
- Osaka, Japan
- Seoul, South Korea (4)
- Singapore (3)
- Taipei, Taiwan(3)
- Tokyo, Japan (11)

3.1.2.c. Microsoft bulut servisleri

Microsoft, 54 farklı bölgede 100'den fazla uç sunucu ile hizmet vermekte ve 140 ülkede bu servisleri ile kullanılabilmektedir. Microsoft bu hizmetleri Azure başlığı altında sunmaktadır. Azure İçerik Dağıtım Ağı (CDN), web sitesi veya mobil uygulama geliştirirken veya yönetirken ya da medya akışı, oyun yazılımı, üretici yazılımı güncelleştirmesi veya IoT uç noktası kodlarken veya dağıtırken yükleme sürelerinin kısaltılmasına, bant genişliğinden tasarruf edilmesine ve daha hızlı yanıt verilmesine olanak tanımaktadır (Anonim, 2019g).

Avrupa veri seti oluşturulurken Microsoft'un Avrupa'da yerleşik ve aşağıdaki konumlarda bulunan sunucularından faydalanılmıştır.

- Vienna, Austria
- Brussels, Belgium
- Prague, Czech Republic
- Copenhagen, Denmark
- Helsinki, Finland
- Marseille, France
- Paris, France
- Berlin, Germany
- Frankfurt, Germany
- Athens, Greece
- Budapest, Hungary
- Dublin, Ireland
- Milan, Italy
- Amsterdam, Netherlands
- Warsaw, Poland
- Barcelona, Spain
- Madrid, Spain
- Stockholm, Sweden
- Zurich, Switzerland
- London, UK
- Manchester, UK

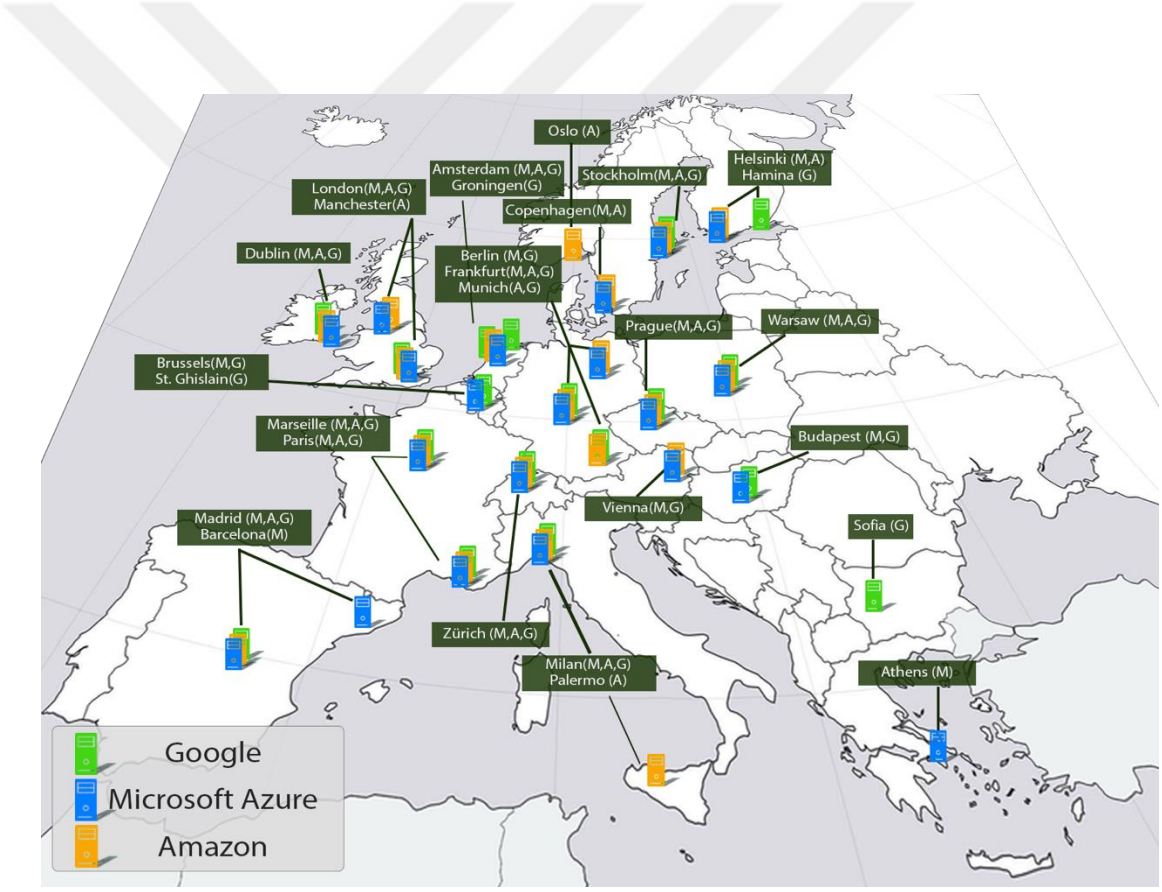
Kuzey Amerika veri seti oluşturulurken aşağıdaki konumlarda bulunan Microsoft sunucularından faydalanılmıştır.

- Toronto, Canada
- Querétaro, Mexico
- San Juan, Puerto Rico
- Ashburn, VA, USA
- Atlanta, GA, USA
- Boston, MA, USA
- Cheyenne, WY, USA
- Chicago, IL, USA
- Dallas, TX, USA
- Denver, CO, USA

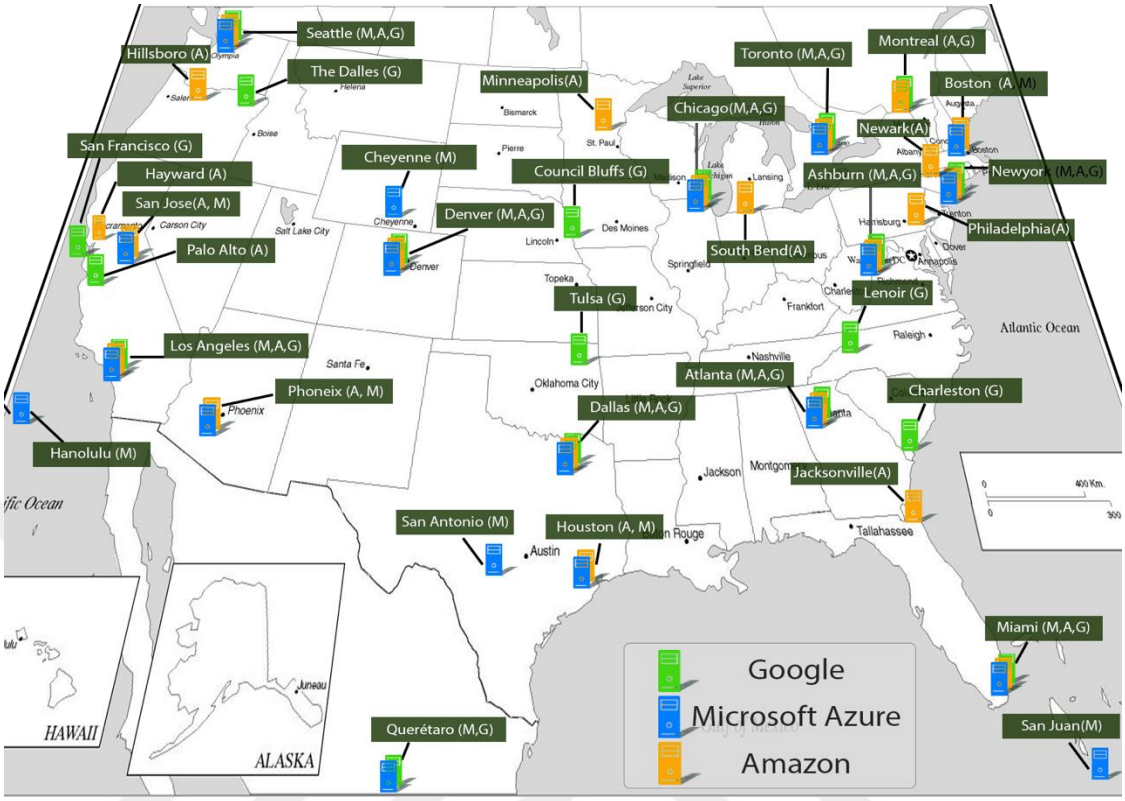
Honolulu, HI, USA
- Houston, TX, USA
- Las Vegas, NV, USA
- Los Angeles, CA, USA
- Miami, FL, USA
- New York, NY, USA
- Phoenix, AZ, USA
- San Antonio, TX, USA
- San Jose, CA, USA
- Seattle, WA, USA

Son olarak Asya veri seti oluşturulurken aşağıdaki konumlarda yer alan Azure sunucuları kullanılmıştır.

- Hong Kong
- Osaka, Japan
- Tokyo, Japan
- Kuala Lumpur, Malaysia
- Manila, Philippines
- Singapore
- Busan, South Korea
- Seoul, South Korea
- Taipei, Taiwan
- Bangkok, Thailand



Şekil 3.2. Microsoft, Amazon ve Google'ın Avrupa'daki bulut ervis sağlayıcıları



Şekil 3.3. Microsoft, Amazon ve Google'ın Amerika'daki bulut servis sağlayıcı konumları

3.1.2.d. Kullanıcı konumları

Kullanıcı konumları belirlenirken Türkiye ‘den 6 tane il seçilmiştir. Bu seçim yapılırken ülkeyi konum olarak kapsayacak bölgelerden seçim yapılmıştır. Avrupa’dan 15 şehir seçilmiş ve bu şehirlerin konumları referans olarak kullanılmıştır. Kuzey Amerika’da tüm kıtayı temsil edecek şekilde en çok bilinen şehirler arasından 25 tanesi seçilmiş ve konumları referans olarak kullanılmıştır. Asya’da ise 40 farklı şehir belirlenmiştir. Seçilen bu şehirler aslında kullanıcıları temsil etmektedirler. Bu şehirlerin konum bilgileri ile bulut servis sağlayıcıların konum bilgilerinden yola çıkılarak gidiş dönüş süresi, yani gecikme hesaplanmaktadır.

Türkiye’den, Avrupa’dan, Kuzey Amerika’dan ve son olarak Asya’dan seçilen şehirler aşağıda listelenmiştir:

TÜRKİYE

-İstanbul
-Ankara
-İzmir
-Mardin
-Erzurum

AVRUPA

-Donetsk
-Lviv
-Budapest
-Szeged
-Sofia
-Stockholm
-Zürich
-Basel
-Rome
-Barcelona
-Madrid
-Palermo
-Glasgow
-Moscow
-Oslo

KUZEY AMERİKA

-Newyork
-San Francisco
-Chicago
-Boston
-Los Angles
-Washington DC
-Seattle
-Austin
-New Orleans
-Las Vegas
-San Diego
-Nashville
-Portland
-Miami
-Denver
-Philadelphia
-Minneapolis
-Detroit
-Dallas
-Atlanta
-Houston
-Orlando
-Memphis
-St. Louis
-Sacramento

ASYA

-dhaka
-shanghai
-beijing
-guangzhou foshan
-chongqing
-tianjin

-shenzhen
-chengdu
-hangzhou
-dongguan
-wuhan
-shenyang
-nanjing
-zhengzhou
-quanzhou
-hong kong
-delhi
-mumbai
-kolkata
-bangalore
-chennai
-hyderabad
-ahmedabad
-jakarta
-tehran
-baghdad
-tokyo
-osaka
-nagoya
-kuala lumpur
-karachi
-lahore
-manila
-riyadh
-singapore
-seoul
-taipei
-bangkok
-ho chi minh city
-hanoi

Hem Sahoo and Glitho (2016) ile verilen sezgisel algoritmada hem de bizim uyarladığımız metasezgisel algoritmada kullandığımız depolama maliyet bilgisi ve kullanılan bant genişliği yük bilgisi Google, Amazon ve Microsoft'un resmi sitelerinden alınmıştır ve aynen kullanılmıştır. (Bu fiyatların kurumlar tarafından çok sık değiştirildiği unutulmamalıdır.)

Çizelge 3.1. Bulut sağlayıcılarının on demand fiyat tablosu

	Standart CPU	**High Memory**	**High CPU**	**SSD**	**Ücret**
Google Cloud	*var*	*yok*	*yok*	*var*	**0,136**
	yok	*var*	*yok*	*var*	**0,159**
	yok	*yok*	*var*	*var*	**0,112**
	var	*yok*	*yok*	*yok*	**0,095**
	yok	*var*	*yok*	*yok*	**0,118**
	yok	*yok*	*var*	*yok*	**0,071**
Amazon Cloud Front	*var*	*yok*	*yok*	*var*	**0,133**
	yok	*var*	*yok*	*var*	**0,166**
	yok	*yok*	*var*	*var*	**0,105**
	var	*yok*	*yok*	*yok*	**0,100**
	yok	*var*	*yok*	*yok*	**0,133**
	yok	*yok*	*var*	*yok*	**0,085**
Microsoft Azure	*var*	*yok*	*yok*	*var*	**0,100**
	yok	*var*	*yok*	*var*	**0,133**
	yok	*yok*	*var*	*var*	**0,085**
	var	*yok*	*yok*	*yok*	**0,100**
	yok	*var*	*yok*	*yok*	**0,133**
	yok	*yok*	*var*	*yok*	**0,085**

Çizelge 3.2. Bulut sağlayıcılarının ortalama depolama fiyatları

	Kuzey Amerika	**Avrupa**	**Asya**	**Avustralya**	**Hindistan**
Google	0,08	0,08	0,09	0,11	0,09
Amzaon	0,085	0,085	0,140	0,114	0,170
Microsoft	0,081	0,081	0,129	0,130	0,181

Çizelge 3.1 ve 3.2' de belirtilen fiyatlar bulut servis sağlayıcılarının resmi web sitelerinden alınmıştır. Bahsedilen fiyatların güncel detaylı halleri Ekler kısmında *Bulut Servis Sağlayıcılarının Konum Bazlı Fiyatları* başlıklı tabloda verilmiştir.

3.2. Yöntem

Bu bölümde tez çalışması kapsamında geliştirilen metasezgisel yaklaşım ve karşılaştırma amacıyla kullanılacak ve LUG olarak isimlendirilen sezgisel algoritma (Sahoo *et al.*, 2016) detaylandırılacaktır. Ayrıca bu yaklaşımlarda kullanılan araçlar ve veri yapıları da açıklanacaktır.

Şekil 3.4. Kullanılan algoritmalar Kotlin programlama dili ile yazılmıştır.

3.2.1. Replika sunucusu

Kullanıcıların web deneyimlerinde gecikme çok önemli bir kriterdir. Bu gecikme her kullanıcı için makul bir seviyede olmalıdır. Veri alışverişi kullanıcı ile veri merkezi arasında gerçekleşmektedir. Bu iki nokta arasında fiziksel bir uzaklık mevcuttur ve bu fiziksel uzaklık başlı başına bir gecikme nedenidir. Bu yüzden kullanıcıların içeriklere makul gecikme seviyelerinde ulaşabilmesini amaçlayan gecikme kısıtları belirlenmiştir. Bu gecikme kısıtlarını karşılayabilmek (QoS) için web siteleri birden fazla sunucu

üzerinden kullanıcılarına hizmet verirler. Kullanıcılar da bu hizmeti rtt gecikmesi önceden belirlenmiş kısıtlar dahilinde olduğu sürece farklı sunuculardan alabilirler.

İşte bu noktada replika sunucu yerleşimi, belirli parametreler doğrultusunda kısıtlanmış veya kısıtlanmamış bir optimizasyon problemi olarak karşımıza çıkmaktadır. Bu problemde temel amaç maliyeti düşürmek ve kullanıcı memnuniyetini arttırmaktır. (Literatürde kullanıcı memnuniyetini dikkate almayan çalışmalar da mevcuttur.)

Konuyla ilgili literatürde rastlanan çalışmalarda

- Gecikme,
- Replika sunucusu kapasitesi,
- Maksimum replika sunucusu sayısı,
- Sunucu yerleşimi

gibi kıstılar kullanılmıştır. Bu kısıtlar son kullanıcı açısından QoS'i iyi veya kötü etkilemiştir (Sahoo *et al*, 2016).

Bu problem için giriş parametreleri literatürde iki ana kategoride belirlenmiştir:

1- Maliyet bağlı parametreler (cost related)
2- Ağ ile ilgili parametreler

Maliyete bağlı parametreler,

1- Dağıtım maliyeti (deployment),
2- Teslim maliyeti (delivery),
3- Yenileme veya güncelleme maliyeti (update) ve
4- Depolama maliyetidir.

Dağıtım maliyeti, satın alma veya kiralama yoluyla temin edilen replika sunucularının yerleştirilmesi sonucu ortaya çıkan maliyeti ifade etmektedir.

Teslim maliyeti, verinin replika sunucusundan son kullanıcıya ulaştırılması esnasındaki ağ maliyetini, yani veri iletimi maliyetini ifade etmektedir. Yenileme maliyeti ise ana sunucudan replika sunucusuna ya da replika sunucusundan başka bir replika sunucusuna veri gönderimi sırasındaki ağ maliyeti için kullanılmaktadır. Depolama maliyeti de bulut sağlayıcı üzerinde sabit olarak saklanan verilerin oluşturduğu maliyettir (Sahoo *et al*, 2016).

Ağ ile ilgili parametreler,

1- Gecikme (Latency)
2- Atlama Sayısı (Hop Count) ve
3- Kullanılabilir Bant Genişliği'dir.

Gecikme, istenen içeriğin replika sunucusundan son kullanıcıya ulaştırılması için gereken süreyi içermektedir. Bu Gidiş-Dönüş Süresi (RTT) düğümler arasındaki coğrafi mesafenin bir fonksiyonudur. (Sahoo *et al*, 2016). Bazı replika sunucusu yerleştirme algoritmaları, gecikmeyi ölçeklenebilir ve anlık hesaplanabilen Global Network Positioning (GNP) gibi popüler tekniklere dayandırmıştır (Eugene and Zhang, 2002).

Atlama sayısı, son kullanıcı ile replika sunucusu arasında belirlenecek ağ üzerinde geçilen ara noktaların sayısıdır. Belirlenen atlama sayısının az olmasının gecikmeyi azaltacağı bilinmektedir.

Kullanılabilir bant genişliği, replika sunucularının yerleştirileceği bölgeler temelini ele alır ve iki sunucu arasındaki minimum bant genişliğini ifade eder. Bu aynı zamanda QoS parametresi olarak da kullanılmaktadır. Bu bilgiye gerçekte internet servis sağlayıcılarından ulaşılabilmektedir.

Geleneksel CDN'lerde replika sunucusu yerleştirmek amacıyla kullanılan algoritmalar,

- QoS hassasiyeti gözetenler,
- Tutarlılık hassasiyeti gözetenler ve
- Enerji hassasiyeti gözetenler

olarak gruplandırılmışlardır. Bulut tabanlı CCDN'lerde de benzer yaklaşımlar gözetilmektedir.

Bulut tabanlı içerik dağıtım ağlarında bulut hizmet sağlayıcıları, farklı coğrafi konumlarda yer alan bulut altyapılarına sahiptirler. CDN sağlayıcı da kendi replika sunucularını dağıtmak için bulut sağlayıcısının altyapısını kullanmaktadır. Bunu da bulut sağlayıcının kullandıkça öde fiyatlandırma modelini (pay as you go) kullanarak ve bulut sağlayıcının kaynaklarını aslında kiralayarak yapmaktadırlar.

İlerleyen aşamalarda problemin çözümünde kullanılan maliyet fonksiyonu hesaplanırken yukarıda değinilen içerik dağıtım ağlarının maliyete bağlı parametrelerinden teslim maliyeti ve depolama maliyeti dikkate alınacaktır.

3.2.2. Denektaşlarının yapısı

Bu tez kapsamında geliştirilen denektaşı setine ,
https://github.com/Captainwoof/Benchmark-Files adresinden ulaşılabilir.
Denektaşı setinde yer alan üç farklı tür düzyazı dosya söz konusudur. Bunlar aşağıda sıralanmıştır:

1. Kullanıcıların konumlarının ve bulut hizmet sağlayıcılarına olan uzaklıklarının bilgisinin saklandığı bir txt dosyası (distance.zip),
2. Bulutların bulunduğu konumların ve o konumda bulunan servis sağlayıcıların mevcudiyetlerinin bilgisinin, ODP ve SP ücret bilgilerinin saklandığı txt dosyası (price.zip),
3. Bu tez kapsamında geliştirilen farklı denektaşı verilerini içeren txt dosyaları (bench ön ekli tüm dosyalar).

1 Numaralı Dosya:

Avrupa, Asya ve Kuzey Amerika bölgesinden seçilen kullanıcı konumlarının o bölgelerdeki Amazon, Azure, Google Bulut Hizmet noktalarına olan uzaklık bilgilerini içermektedir. Yapısı şu şekildedir.

Çizelge 3.3. Kullanıcı - bulut – mesafe (.txt) dosyası

KULLANICI KONUMU	BULUT KONUMU	MESAFE
İSTANBUL	**Frankfurt**	**1869 km**

Kullanıcı – Bulut Konum dosyaları olarak aşağıdakiler belirlenmiştir;

Asya – Avrupa	Avrupa – Amerika	Amerika -Avrupa
Asya – Amerika	Avrupa – Asya	Amerika Asya

Sol taraf kullanıcı konumunu sağ taraf bulut konumunu göstermektedir.

2 Numaralı Dosya:

Avrupa, Asya ve Kuzey Amerika bölgelerinde bulunan Amazon, Azure, Google Bulut Hizmet noktalarına ait fiyat bilgisini içermektedir. Yapısı şu şekildedir:

Çizelge 3.4. Bulut sağlayıcı– fiyat (.txt) dosyası

Bulut Alanının Bulunduğu şehir	Hizmet Sağlayıcı	Hizmet Durumu	ODP	SP
Viyana	Azure	1	0,133	0,0196
	Amazon	1	0,166	0,0238
	Google	0	0,0	0,0

3 Numaralı Dosya:

Denektaşı veri dosyaları da txt formatında olup kendi içinde 3 bölümden oluşmaktadır. İlk satırda kaç kullanıcı ve kaç bulut servis sağlayıcısı olduğuna dair bir bilgi bulunmaktadır. Örneğin ilk satırda 10 – 5 gibi bir ifade varsa ilgili denektaşının 10 farklı kullanıcıyı 5 farklı şehirde yer alan bulut hizmet sağlayıcısına atamak amacına yönelik hazırlandığına işaret etmektedir.

Çizelge 3.5. Örnek denektaşı dosyası (id-konum-aylık yük-depolanan yük)

	7	**5**	
U1	szeged	70	35
U2	moscow	80	55
U3	palermo	90	25
U4	glasgow	80	80
U5	zurich	100	44
U6	donetsk	90	45
U7	barcelona	60	70
STOCKHOLM			
WARSAW			
DUBLIN			
VIENNA			
HAMINA			

Örnek bir denektaşı dosyasının içeriği Çizelge 3.5 ile verilmiştir. Bu denektaşı 7 farklı kullanıcıyı 5 farklı konumda (Stockholm, Warsaw, Dublin, Vienna ve Hamina) bulunan bulut hizmet sağlayıcısına atama amacıyla hazırlanmıştır. U ile başlayan her bir satır 7 kullanıcıdan birisine aittir ve 4 sütun içermektedir: kullanıcı id'si sütunu, kullanıcının konumu sütunu, GB cinsinden aylık veri kullanım miktarı sütunu ve son olarak GB cinsinden depolanan veri miktarı sütunu.

Her iki algoritma çalıştırılmadan önce yukarıdaki paragraflarda tanıtılan bu veri dosyaları okunmakta ve KULLANICI ve BULUT sınıfı nesnelerine aktarılmaktadır.

3.2.3. Algoritmaların tanıtılması ve problemin adım adım çözülmesi

Problem temelde internet kullanıcılarının web deneyimini iyileştirmek için söz konusu kullanıcıları en doğru konumdaki sunucu ile eşleştirme işleminin nasıl yapılması gerektiğine cevap aramaktadır. Bu nedenle bulut hizmet sağlayıcıları replika sunucularını coğrafi olarak en uygun alanlara koymak ve bunu yaparken de maliyeti minimumda tutmak istemektedirler.

Problemin özneleri:

1. Bulut Hizmet Sağlayıcıları
2. Web Kullanıcıları

Çözüm Aranan:

1. Kullanıcı deneyimi için en uygun konum (QoS)
2. Bulut Hizmet Sağlayıcı için en ucuz maliyetli yerleşim

Çizelge 3.5 ile verilen denektaşı için örnek bir rastgele çözüm aşağıdaki gibi oluşturulabilir.

Çizelge 3.6. Örnek Çözümün Kullanıcı-Bulut Konum gösterimi

Koyu Yazılan Şehirler Kullanıcı Konumlarını
Altlarındaki Şehirler Atandıkları Konumları Belirtir

Kullanıcı Konumları	**U1** **ZURICH**	**U2** **MOSCOV**	**U3** **PALERMO**	**U4** **GLASGOW**	**U5** **SZEGED**	**U6** **DONETSK**	**U7** **BARCELONA**
Bulut Konumu	DUBLIN	WARSAW	DUBLIN	HAMINA	DUBLIN	VIENNA	STOCKHOLM
Bulut Tipi	GOOGLE	AZURE	AZURE	GOOGLE	AMAZON	AMAZON	GOOGLE

Çizelge 3.6'da ilk satır kullanıcıları temsil etmektedir. Her kullanıcının altında bu kullanıcının hangi konumda bulunan hangi bulut servis sağlayıcısına atandığı bilgisi mevcuttur. Yani U1 kullanıcısının konumu Zurich'tir ve Dublin'deki Google sunucuna atanmıştır.

Çizelge 3.7. Bulut sağlayıcıları konum bazlı fiyatları

Koyu Yazılan Şehirler Bulut Konumlarını
Altlarındakiler de Bulut Servis Sağlayıcıları Belirtir
En Alt Satır Servis Sağlayıcıların ODP ve SP Aylık GB Birim Fiyatını Gösterir

C1		C2		C3		C4		C5		C6		C7	
DUBLIN		**WARSAW**		**DUBLIN**		**HAMINA**		**DUBLIN**		**VIENNA**		**STOCKHOLM**	
GOOGLE		AZURE		AZURE		GOOGLE		AMAZON		AMAZON		GOOGLE	
0,159	0,023	0,133	0, 022	0,133	0,0192	0,159	0,02	0,166	0,0230	0,166	0,0238	0,159	0,0226

Çizelge 3.8. Denektaşında bulunan maliyet hesabında kullanılacak kullanım verileri

Koyu Yazılan Şehirler Kullanıcı Konumlarını
Altlarındakiler de Aylık Kullanım Olarak Toplam Yük (GB) ve Depolanan Veriyi (GB) Belirtir

ZURHICH		MOSCOW		PALERMO		GLASGOW		SZEGED		DONETSK		BARCELONA	
100	44	80	55	90	25	80	80	70	35	90	45	60	70

Bu ratsgele çözümün maliyeti aşağıda verilen 3'nolu formül ile hesaplanmaktadır. Bu formülün detayları LUG algoritmasının açıklandığı bölümde detaylandırılacaktır.

$$C = \sum_{i \in F} W \alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} \qquad (4)$$

1. Adım

   U1: ZURICH | Aylık Kullanım Miktarı : 100 - 44

   C1: DUBLIN – GOOGLE | Ücret: ODP: 0,159 - SP: 0,023

   Maliyet(U1) = 100*0,159 + 44*0,023

   M1: Maliyet(U1) = 16,912

2. Adım

   U2: MOSCOW | Aylık Kullanım Miktarı : 80 - 55

   C2: WARSAW – AZURE | Ücret: ODP: 0,133 - SP: 0,022

   Maliyet(U2) = 80 *0,133 + 55*0,022

   M2: Maliyet(U2) = 11,85

3. Adım

   U3: PALERMO | Aylık Kullanım Miktarı : 90 - 25

   C3: DUBLIN – AZURE | Ücret: ODP: 0, 133 - SP: 0, 0192

   Maliyet(U3) = 90 *0, 133 + 25*0, 0192

   M3: Maliyet(U3) = 12,45

4. Adım

   U4: GLASGOW | Aylık Kullanım Miktarı : 80 - 80

   C4: HAMINA – GOOGLE | Ücret: ODP: 0,159 -SP: 0, 02

   Maliyet(U4) = 80 *0,159 + 80 *0, 02

   M4: Maliyet(U4) = 14,32

5. Adım

   U5: SZEGED | Aylık Kullanım Miktarı : 70 - 35

   C5: DUBLIN – AMAZON | Ücret: ODP: 0, 166-SP: 0,023

   Maliyet(U5) = 70 *0, 166+ 35*0, 023

   M5: Maliyet(U5) = 12,425

6. Adım

   U6: DONETSK | Aylık Kullanım Miktarı : 90 - 45

   C5: VIENNA – AMAZON | Ücret: ODP: 0, 166-SP: 0,0238

   Maliyet(U6) = 90 *0,159 + 45*0, 0238

   M6: Maliyet(U6) = 15,38

7. Adım

   U7: BARCELONA | Aylık Kullanım Miktarı : 60 - 70

   C7: STOCKHOLM – GOOGLE | Ücret: ODP: 0,159 -SP: 0, 0226

   Maliyet(U7) = 60 *0,159 + 70*0, 0226

   M7 : Maliyet(U7) = 11,122

8. Adım

   Maliyet(Toplam) = M1 + M2 + M3 + M4 + M5 + M6 + M7

   Maliyet(Toplam) = 16,912 + 11,85 + 12,45 + 14,32 + 12,425+ 15,381+ 11,122

   Maliyet(Toplam) = 80,14

Yukarıdaki bilgiler ışığında rastgele oluşturulan örnek çözümün maliyeti 80,14 olarak bulunmuştur.

Çizelge 3.9. Rastgele oluşturulmuş farklı bir çözüm

Koyu Yazılan Şehirler Bulut Konumlarını
Altlarındakiler de Bulut Servis Sağlayıcıları Belirtir
En Alt Satır Servis Sağlayıcıların ODP ve SP Aylık GB Birim Fiyatını Gösterir

C1		**C2**		**C3**		**C4**		**C5**		**C6**		**C7**	
STOCKHOLM		**WARSAW**		**VIENNA**		**STOCKHOLM**		**STOCKHOLM**		**WARSAW**		**WARSAW**	
AZURE		AMAZON		AZURE		AZURE		AZURE		AMAZON		AMAZON	
0,133	0,022	0,166	0, 0238	0,133	0,0196	0,133	0,022	0,133	0,022	0,166	0,0238	0,166	0,0238

Çizelge 3.9'da rastgele oluşturulmuş başka bir çözüm verilmiştir. Yukarıdaki adımlar takip edilerek elde edilen bu çözümün maliyeti ise 100,15 olarak bulunmaktadır. Maliyetler karşılaştırıldığında birinci çözümün ikinci çözümden daha iyi bir çözüm olduğu kolaylıkla anlaşılabilmektedir.

3.2.3.a. Çözüm uzayının büyüklüğü

N adet kullanıcının, ***M*** adet şehirde yerleşik ***O*** adet bulut sağlayıcısından herhangi birine atanması durumunda çözüm uzayının büyüklüğü aşağıdaki formül ile hesaplanabilir.

$$(\boldsymbol{M} \cdot \boldsymbol{O})^{\boldsymbol{N}} \qquad (5)$$

N adet kullanıcının her biri ***M*** farklı şehirde yerleşik bir bulut sağlayıcıya atanabilir. Bu ise

$$\boldsymbol{M}^{\boldsymbol{N}}$$

farklı çözüm olduğu anlamına gelmektedir. Benzer şekilde her bir kullanıcı atandığı şehirdeki bulut sağlayıcılardan herhangi birisinden hizmet alabilir. Bu ise

$$\boldsymbol{O}^{\boldsymbol{N}}$$

farklı çözüm olduğu anlamına gelir.

Çizelge 3.10. Kullanılan denektaşlarının çözüm uzaylarının büyüklükleri

	KULLANICI KONUM SAYISI	BULUT KONUM SAYISI	SERVİS SAĞLAYICI SAYISI	BÜYÜKLÜK
BENCHMARK_ASIA	40	16	3	$1{,}7768 \times 10^{67}$
BENCHMARK_ASIA_100	100	16	3	$1{,}3348 \times 10^{168}$
BENCHMARK_ASIA_200	200	16	3	$1{,}7711 \times 10^{336}$
BENCHMARK_EU	19	32	3	$4{,}4041 \times 10^{37}$
BENCHMARK_EU_100	100	32	3	$1{,}6870 \times 10^{198}$
BENCHMARK_EU_200	200	32	3	$2{,}8460 \times 10^{396}$
BENCHMARK_USA	25	34	3	$1{,}6406 \times 10^{50}$
BENCHMARK_USA_100	100	34	3	$7{,}2446 \times 10^{200}$
BENCHMARK_USA_200	200	34	3	$5{,}2484 \times 10^{401}$
BENCHMARK_ALL_100	100	82	3	$1{,}2402 \times 10^{239}$
BENCHMARK_ALL_200	200	82	3	$1{,}5382 \times 10^{478}$

3.2.3.b. Sezgisel ve metasezgisel algoritmalara olan ihtiyaç

Çizelge 3.10 eldeki problemin çözüm uzayındaki bütün kombinasyonları tek tek deneyen kaba kuvvete dayalı bir algoritma ile çözülemeyeceğini göstermektedir. Dolayısıyla, bu problemi makul sürelerde çözebilmek için sezgisel ve metasezgisel algoritmalara ihtiyaç duyulmaktadır. Bu nedenle bu tez kapsamında bu problemin çözümüne yönelik literatürde ilk defa bir metasezgisel yaklaşım (Tabu Arama Algoritması Tabanlı) geliştirilmiştir ve elde edilen sonuçlar bir sonraki bölümde anlatılan sezgisel ile karşılaştırılmıştır

3.2.4. LUG algoritması

Sahoo ve Glitho (2016), 2016 yılında yaptıkları yayınlarında replika sunucularının yerleşimi için kullanılan bazı açgözlü sezgisel algoritmaları ele almışlar, bu algoritmaların avantajlarını ve dezavantajlarını değerlendirmişler ve yine bu algoritmaların dezavantajlarını giderecek bir çözüm üretmeye çalışmışlardır. Bu yaklaşıma da Least Usage Greedy (LUG) adını vermişlerdir.

Kendi çözümleri için temel olarak ele aldıkları sezgisel algoritmalar *Greedy Site (GS)* ve *Greedy User (GU)* algoritmalarıdır (Chen *et al*, 2012). Bu ekip bu algoritmaların başarılı bir şekilde replika sunucularını dağıtabildiğini, ancak bazı dezavantajlara sahip olduklarını tespit etmişlerdir. Bu algoritmalardan *Greedy Site* her döngüde en faydalı bulut sunucusunu bulmaya çalıştığından çok fazla işlem yapmaktadır. Dolayısıyla işlem maliyeti olarak verimsizdir. *Greedy User* algoritması da benzer şekilde her kullanıcı için minimum maliyetli bir bulut sunucusunu bulmaya çalışmaktadır. Bu algoritma da her döngüde çok fazla sayıda işlem gerektirmektedir. Bu algoritmalar uygun sunucuları ararken daha fazla yerleştirme işlemi yapamadan sonlanabilmektedirler. Her iki algoritma da yerleştirdikleri kullanıcıların QoS eşiklerini sağlamaktadırlar. Temel alınan bu algoritmalar çevrimdışı olarak çalışan algoritmalardır. Yani daha önce alınan kullanıcı ve bulut verisi üzerinde işlem yapmaktadırlar. Literatürde çevrim içi çalışan benzer başka algoritmalar da bulunmaktadır. Bu algoritmaların temel sorunu talep anında uygun bulut

sunucu aranırken QoS eşiğinin aşılmasıdır. Bu da son kullanıcı tarafında istenmeyen bir durum olmaktadır.

Sahoo ve Glitho (2016) geliştirilen LUG algoritması bulut ve kullanıcı konumlarına bakarak kullanıcıların QoS eşiklerinden daha düşük olduğu konumları potansiyel bulut konumları olarak tanımlamaktadır. Bu eşik değerleri gecikme doğrultusunda fiziksel uzaklıkla bağlantı olarak hesaplandığı için konum bilgileri anlam kazanmaktadır.

LUG algoritması iki adımdan oluşmaktadır:

1. Yerleştirme (Placement)
2. İyileştirme (Refinement)

Yerleştirme adımı bulut alanlarının işletme maliyetlerine göre artan bir şekilde seçilmesini ve henüz herhangi bir bulut alanına atanmamış tüm potansiyel son kullanıcıların bir buluta atanmasını içermektedir. Bu aşamada kullanılan maliyet fonksiyonu aşağıdaki formüller yardımıyla hesaplanmaktadır:

$$X_{ij} = \begin{cases} 1 \\ 0 \end{cases} \qquad (2)$$

$$y_i = \begin{cases} 1 \\ 0 \end{cases} \qquad (3)$$

$$C = \sum_{i \in F} W \alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} \qquad (4)$$

Burada α_i^S birim depolama kiralama maliyetini, yani kullanıcıların depoladıkları veri miktarının birim maliyetini, α_i^B ise bant genişliği birim teslim maliyetini, yani aylık trafik sonucu oluşan yükün birim maliyetini göstermektedir. $\boldsymbol{F}$ potansiyel replika sunucularının, $\boldsymbol{H}$ ise kullanıcıların kümesidir. $\boldsymbol{W}$ replika boyutunu, $\boldsymbol{\omega_j}$ ise üretilen istek sayısını, yani toplam yükü belirtmektedir. $\boldsymbol{X_{ij}}$, son kullanıcı *j*, *i* potansiyel sunucusuna atanmışsa 1, değilse 0 değerini almaktadır. $\boldsymbol{y_i}$ replika sunucusu atamak için seçilmişse 1, değilse 0 değerini almaktadır.

Potansiyel bulut alanları bulunurken kullanıcının konumu ile bulut sağlayıcı konumu arsındaki uzaklığa bağlı gecikme fonksiyonundan faydalanılmaktadır. Hesaplanan gecikmeler her kullanıcı için gecikme eşik parametresi ile karşılaştırılmaktadır. Bu karşılaştırma sonucunda eşik değerin altında değere sahip bulut sağlayıcı o kullanıcı için potansiyel bir bulut alanı olmaktadır.

Bulut alanları önce depolama maliyetlerinin, daha sonra birim teslim maliyetlerinin artan sırasına göre sıralanırlar. Kullanıcılar sıralanan bu bulut alanlarının en düşük maliyetli olanlarından başlanarak ilgili alanlar ile eşleştirilirler. Bulut alanlarının seçimi ve son kullanıcıların potansiyel bulut sitelerinden birine atanması, tüm son kullanıcılar atanana kadar devam eder (Sahoo ve Glitho, 2016).

İyileştirme adımında ise potansiyel bulut alanlarından yola çıkılarak oluşturulan bulut alanı – kullanıcı listesi üzerinde sırasıyla tüm bulut alanları ziyaret edilmektedir. En düşük birim teslim maliyetli bulut alanı ile başlanarak, ilk çözümdeki her bulut alanı için iyileştirme işlemi, yani bulut alanını kaldırma ve son kullanıcılarını başka bulut alanlarına yeniden atama olasılığı denetlenir. Eğer maliyeti düşürecek bir iyileştirmeye rastlanırsa gerekli güncelleme yapılmaktadır. Bu adımlar takip eden paragrafta detaylandırılmıştır.

İyileştirme esnasında öncelikle tüm son kullanıcıların potansiyel bulut alanları kontrol edilir ve başlangıçta atanan alanın dışında başka bir alan olup olmadığına bakılır. Zaten depolama ve birim teslim maliyeti açısından sıralanmış olan başlangıç çözümü üzerinde en düşük maliyetli bulut alanından başlanarak bir sonraki en düşük bulut alanına doğru ilerleyen bir akış mevcuttur. İyileştirme işleminde taramaya bu bulut alan listesinin en başından başlanır ve ilgili bulut alanına atanmış olan potansiyel kullanıcılar bir sonraki bulut alanına kaydırılır. Bu esnada bu kullanıcıların ilgili bulut alanı açısından potansiyel kullanıcı olup olmadıkları öncelikli olarak kontrol edilir ve yeni bir maliyet hesaplanır. Maliyetin arttığı anlaşılırsa o bulut alanı listeden çıkarılır. Kullanıcılar bir sonraki en iyi bulut alanına tekrar atanır. Maliyet artmazsa bir sonraki bulut alanı durumu daha da iyileştirebilmek için kontrol edilir ve korunur.

Sahoo ve Glitho (2016), algoritmalarını denerken maliyet hesabında ticari bulut firmalarının gerçek kiralama fiyatlarını kullanmışlar, kullanıcı konumlarını ve bulut alanlarını ise rastgele oluşturmuşlardır. Son kullanıcı sayısını 100 ile 700 arasında değiştirerek, bulut konum sayısını da 70'e kadar arttırarak çeşitli denemeler yapmışlardır. Son kullanıcı sayısı, bulut alan sayısı ve QoS eşiği dışındaki parametreleri sabit tutmuşlardır. Yani her kullanıcı için varsayılan yük aylık 1.5 GB, replika boyutu da 5 GB olarak belirlenmiştir. Her bir farklı test için varsayılan olarak 45ms-60ms aralığında bir QoS eşiği belirlemişlerdir. Her testi 20 kere çalıştırıp ortalamasını kullanmışlardır.

Bu tez kapsamında LUG algoritması da yukarıda detaylandırıldığı şekliyle Kotlin programlama dili kullanılarak gerçeklenmiş ve sonuçları önerilen Tabu Arama algoritması ile karşılaştırılmıştır.

3.2.4.a. LUG algoritmasının sözde kodu ve bir örnek ile açıklanması

LUG algoritmasının yerleştirme adımına ait sözde kod Şekil 3.2 ile verilmiştir.

LUG Algoritması Yerleştirme Adımı

Input F: Bulut Alanları, U: Kullanıcı Listesi

Output X:Atanmış Kullanıcıları İçeren Bulut Alanları Dizisi

```
1.  Bulut Alanlarını (F) artan şekilde depolama maliyetine göre sırala
2.  Bulut Alanlarını (F) artan şekilde birim teslim maliyetine göre sırala
3.  for ∀ bulut i ∈ F
4.      for ∀ kullanıcı  j ∈ Ui
5.          if  kullanıcı j henüz atanmamışsa
6.              Ai ←Ai U{ j }  // rtt' ye kullanıcıyı ata
7.          end if
8.      end for
9.      if  Ai  ≠ ∅  ise
10,         S ← S U{i} // bulut alanı seçilir
11.     end if
12. end for
13. Seçilen S Bulut alanlarından ve atanan Ai ilerden bir X çözüm nesnesi oluştur
14. return X
```

Şekil 3.5. LUG algoritması yerleştirme adımı

Yukarıdaki sözde kodda ***Ai*** , ***S***'e atanacak olan son kullanıcı kümesini ifade eder. ***S*** ise çoğaltma sunucularını yerleştirmek için seçilen geçerli bulut alanını belirtir. İyileştirme adımı temelde her bir kullanıcıyı bir bulut alanına yerleştirmektedir. Bu adımın en can alıcı noktası bu eşleştirme yapılırken potansiyel sunucu isimli bir kavramdan faydalanılmasıdır. Bir kullanıcı açısından potansiyel sunucu ilgili sunucudan temin edilecek bir verinin gecikmesinin eşik değerin altında olmasıdır.

Çizelge 3.5 ile verilen denektaşının (10 kullanıcı ve 5 farklı bulut servis sağlayıcısı içeren) LUG algoritmasının iyileştirme aşamasıyla çözümüne ait adımlar aşağıda sıralanmıştır:

1) Oluşturulan bulut alanları önce depolama (SP) daha sonra birim teslim maliyetlerinin (ODP) artan sırasına iki kere sıralanmaktadır. Sonrasında potansiyel bulut alanları bulunan kullanıcılar, maliyetlerine göre sıralanmış bu bulut alanlarına bu sıralama gözetilerek yerleştirilmektedirler. Sıralanan bu potansiyel bulut alanlarının en düşük maliyetli olanlarından başlanarak kullanıcılar bu alanlar ile eşleştirilmektedir. Her bulut alanı için henüz atanmamış tüm potansiyel kullanıcılar ilgili buluta atanmaktadır. Bulut alanlarının seçimi ve son kullanıcıların potansiyel bulut sitelerinden birine atanması, tüm son kullanıcılar atanana kadar devam etmektedir (Sahoo and Glitho, 2016).

2) Başlangıçta kullanıcıların eşleşebileceği potansiyel replika bulut sunucu konumları tespit edilmektedir. Burada temel nokta gecikme süresidir. Potansiyel sunucular belirlenirken 45 - 60 ms gibi bir gecikme eşiği temel alınıp sunucular her bir kullanıcı için atanmaktadır.

Çizelge 3.11. Bulut servis sağlayıcıya atanabilecek kullanıcıların uzaklıkları

Kuş Uçuşu Mesafe (KM)	**szeged**	**moscow**	**palermo**	**glasgow**	**zurich**	**donetsk**	**barcelona**
STOCKHOLM	1456	1229	2381	1379	1467	1790	2277
WARSAW	670	1150	1679	1691	1038	1277	1865
DUBLIN	2036	2796	2260	308	1238	3106	1471
VIENNA	357	1669	1145	1642	590	1586	1348
HAMINA	1657	816	2678	1894	1895	1556	2726

Bulut alanlarının kullanıcı konumlarına göre gecikme süreleri Çizelge 3.11'deki mesafe bilgileri ve 1 numaralı formül kullanılarak hesaplanacaktır. Aşağıda sadece ilk bulut alanı için hesaplama yapılırsa:

1 - *RTT(Stokholm, Szeged) = 0,02 * Dist(Stokholm, Szeged) + 5*

*= 0,02 * 1456 + 5*

= ***34,12***

2 - *RTT(Stokholm, Moscow) = 0,02 * Dist(Stokholm,* Moscow*) + 5*

*= 0,02 *** 1229 *+ 5*

= ***29,58***

3 - *RTT(Stokholm, Palermo) = 0,02 * Dist(Stokholm, Palermo)+ 5*

*= 0,02 ** 2381*+ 5*

= ***52,62***

4 - *RTT(Stokholm,* Glasgow*)= 0,02 * Dist(Stokholm,* Glasgow*)+ 5*

*= 0,02 ** 1379*+ 5*

= ***32,58***

5 - *RTT(Stokholm, Zurich) = 0,02 * Dist(Stokholm, Zurich)+ 5*

*= 0,02 ** 1467*+ 5*

= ***34,34***

6 - *RTT(Stokholm,* Donetsk*)= 0,02 * Dist(Stokholm,* Donetsk*)+ 5*

*= 0,02 ** 1790*+ 5*

= ***40,8***

7 - *RTT(Stokholm, Barcelona) = 0,02 * Dist(Stokholm, Barcelona)+ 5*

*= 0,02 ** 2277*+ 5*

= ***50,54***

Çizelge 3.12. Bulut sağlayıcı ve kullanıcılar arasındaki kuş uçuşu mesafe (km) ve gecikme süresi (ms)

Kuş Uçuşu Mesafe (KM)	**szeged**	**moscow**	**palermo**	**glasgow**	**zurich**	**donetsk**	**barcelona**
STOCKHOLM	1456	1229	2381	1379	1467	1790	2277
Gecikme Süreleri	34,12	***29,58***	52,62	32,58	34,34	40,8	50,54

Çizelge 3.12 incelendiğinde uzaklığa bağlı gecikme değeri <= 30 ms (bu değerin girdi olarak algoritmaya verildiği kabulü yapılmaktadır.) olan tek şehir Moscow'dur. Yani Stockholm şehrindeki bulut servis sağlayıcı, Moscow'daki kullanıcılar için potansiyel bulut alanlarından birisidir, daha da önemlisi bu örnek için tek potansiyel bulut alanıdır.

Bu sonuçlara göre Stockholm şehrindeki bulut sağlayıcının eşleşebileceği kullanıcı konumları:

1. Moscow şehridir.

Kullanıcılar açısından bakıldığında bu gecikme eşiğinin altında bulunan tüm bulut sağlayıcılar kullanıcı için potansiyel bulut sağlayıcılarıdır. Diğer tüm bulut sağlayıcıları içinde bu adımlar tekrarlanarak bu hesaplamalar yapılır. Hesaplamalardan sonra uygun bulunan bulut–kullanıcı konumları eşleştirilir ve algoritmanın yerleştirme adımı tamamlanır.

İyileştirme Adımı:

İyileştirme algoritmasına ait sözde kod aşağıdadır.

LUG Algoritmasının İyileştirme Adımına Ait Sözde Kod

Giriş: **X:** Kullanıcıları Atanmış Bulut alanları, Bulut Servis Sağlayıcı Ücretleri, Kullanıcı Aylık Kullanım Verileri

Çıkış: Güncellenmiş Bulut Alanı-Kullanıcı Atama Listesi

```
1   C cur ← X çözümünün maliyeti
2   for her kullanıcı  j ∈ H
3       Kullanıcının atanmış Bulutu hariç potansiyel bulut listesindeki en uygun bulutu
        ata
4       S_j^min ← N_j içinde ki en iyi  bulut alanı
5   end for
6   for her bulut alanı i ∈ S
7       Cnt ← NULL
8       for her kullanıcı  j ∈ Ai
9           if  S_j^min ≠ NULL
10              Cnt ← Cnt + 1
11          end if
12      end for
13      if Cnt = Ai
14          C^new ← i bulutundan j bulutuna S_j^min atanması sonucu yeni maliyet
15          if  C^new ≤ C^cur
16              for  her kullanıcı j ∈ Ai
17                  A_{S_j^min} = A_{S_j^min} U { j } // j kullanıcısını yeniden
                                                      //i bulut alanına ata
18                  Update  S_j^min
19              end for
20              S = S −{i}//  i bulut alanını sil
21              C^new ≤ C^cur
22          end if
23      end if
24  end for
25  Seçilen S Bulut alanlarından ve atanan Ai ilerden bir X* çözüm nesnesi oluştur.
26  Return X
```

Şekil 3.6. LUG algoritması iyileştirme adımı

Her bulut alanı ***i*** için, iyileştirme işlemi aşağıdaki şekilde tarif edilir.

Öncelikle, tüm son kullanıcılar için bir sonraki en iyi bulut sitesinin var olup olmadığı kontrol edilir. Sözde koddaki ***Cnt*** bu kontrol değişkenini belirtmektedir. Bir son kullanıcının bir sonraki en iyi bulut alanı yani S_j^{min}**,** bir sonraki en düşük maliyetli olan bulut alanıdır. Son kullanıcı için, halihazırda atanmış olduğu bulut alanının, kalan bulut alanları arasında en düşük birim maliyetli alan olduğu unutulmamalıdır.

i bulut alanındaki tüm son kullanıcılar için bir sonraki en iyi bulut alanı mevcutsa, o zaman tüm son kullanıcılar bir sonraki en iyi bulut alanlarına yeniden atanırlar. Bu atama işlemi sonunda ortaya çıkacak maliyet hesaplanır. Maliyet iyileşirse, bulut alanı ***i*** kaldırılır ve son kullanıcıları bir sonraki en iyi sitelerine tekrar atanır. Son kullanıcılar yeniden atandığında, bir sonraki en iyi bulut alanları yani S_j^{min}'leri güncellenir. Maliyet artmazsa, bir sonraki bulut sitesi çözümü daha da iyileştirmek için kontrol edilir.

1. Adım
   Yerleştirme adımı sonucu çözüm Çizelge 3.14'deki gibi olsun.
   Bu çözümün ilk maliyeti hesaplanır.

Çizelge 3.13. Kullanıcılar ve yerleştikleri bulut alanları

Bulut Sunucu	Kullanıcı Sayısı
Stockholm	1
Warsaw	5
Dublin	-
Vienna	1
Hamina	-
TOPLAM	7

Maliyet hesabının nasıl yapıldığı daha önce Çizelge 3.7 ve 3.8'deki örneklerin çözümünde gösterilmiştir. Aynı şekilde 4 numaralı formül kullanılarak Çizelge 3.14.'ün maliyeti hesaplanır.

$$C1 = \sum_{i \in F} W\alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} = 69{,}27$$

Sonra sırası ile Stockholm, Warsaw, Dublin, Vienna, Hamina konumlarındaki bulut sağlayıcılar tek tek ele alınıp eşleştikleri kullanıcılar kontrol edilir.

2. Adım

Stockholm'deki kullanıcının Stockholm'den başka eşleşebileceği bir potansiyel bulut alanı mevcut ise o kullanıcı ilgili bulut alanına taşınır. Ardından maliyet tekrar hesaplanır.

$$C2 = \sum_{i \in F} W\alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} = 69{,}79$$

Maliyetler karşılaştırıldığında C2 > C1 olduğu görülür. Yani daha iyi bir sonuç elde edilememiştir.

Yeni kullanıcı – bulut eşleşmeleri Çizgelge 3.15. deki gibi olmuştur.

Çizelge 3.14. 2. Adım sonunda kullanıcılar ve yerleştikleri bulut alanları

Bulut Sunucu	Kullanıcı Sayısı
Stockholm	1
Warsaw	5
Dublin	-
Vienna	1
Hamina	-
TOPLAM	7

3. Adım

Aynı şekilde Warsaw'daki kullanıcıların Warsaw'dan başka eşleşebileceği başka bir potansiyel bulut alanı mevcut ise o bulut alanlar arasında en düşük maliyetli üçüncü bulut alanına taşınır. Ardından maliyet tekrar hesaplanır.

$$C3 = \sum_{i \in F} W\alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} = 68{,}78$$

Maliyetler karşılaştırıldığında C3 < C1 < C2 olduğu görülür. Yani daha iyi bir sonuç elde edilmiştir. Yeni eşleşme aşağıdaki gibidir.

Çizelge 3.15. 3. Adım sonunda kullanıcılar ve yerleştikleri bulut alanları

Bulut Sunucu	*Kullanıcı Sayısı*
Stockholm	1
Warsaw	0
Dublin	-
Vienna	6
Hamina	-
TOPLAM	7

4. Adım

Dublin hiçbir kullanıcı ile eşleşmediği için çözümde kullanılmaz ve Vienna'ya geçilir ve Vienna'dan başka eşleşebileceği başka bir potansiyel bulut alanı mevcut ise o bulut alanlar arasında en düşük maliyetli ikinci bulut alanına taşınır. Ardından maliyet tekrar hesaplanır.

$$C4 = \sum_{i \in F} W\alpha_i^S y_i + \sum_{i \in F} \sum_{j \in H} \omega_j \alpha_i^B X_{ij} = 68{,}78$$

Hesaplama sonunda bulunan maliyet daha önce hesaplanan maliyetlerden daha iyi olmadığı için bir değişiklik gözlenmez.

Daha önceki maliyetler:

C1: 69,29 C2: 69,79 C3: 68,78 C4: 68,78

Algoritmanın bu aşamasına kadar olan maliyetler karşılaştırıldığında C2 maliyetli çözüm aralarındaki en iyi çözüm olduğu için kullanıcı – bulut konumları hala aynı şekildedir.

Çizelge 3.16. 4. Adım sonunda kullanıcılar ve yerleştikleri bulut alanları

Bulut Sunucu	*Kullanıcı Sayısı*
Stockholm	1
Warsaw	0
Dublin	-
Vienna	6
Hamina	-
TOPLAM	7

5. Adım

Hamina hiçbir kullanıcı ile eşleşmediği için çözümde kullanılmaz.

Son durum Çizelge 3.18'deki gibidir.

Çizelge 3.17. Tüm adımlar sonucu kullanıcıların son durumu

Bulut Sunucu	*Kullanıcı Sayısı*
Stockholm	1
Warsaw	0
Dublin	-
Vienna	6
Hamina	-
TOPLAM	7

Tüm bulut alanları tarandığı için algoritma sonlanmıştır ve ideal çözüm elde edilmiştir. Bu örnek veri seti üzerinden değerlendirildiğinde 7 kullanıcı için en uygun çözümün

Stockholm için bir kullanıcı ve Vienna için 6 kullanıcıdan oluştuğu görülmüştür. LUG algoritması replika sunucu yerleşimini yukarıda anlatılan şekilde tamamlamaktadır.

3.2.5. Kısaca tabu arama algoritması (Tabu search)

Bu bölümde literatürde sezgisel olarak çözümler getirilen ve bu çalışmada da bu sezgisel algoritmalardan birinin (LUG algoritması) referans alındığı replika sunucusu problemi için metasezgisel bir yaklaşım tasarlanmıştır. Bu bağlamda metasezgisel olarak Tabu Arama algoritması ile problem modellenecek ve performans kriterleri değerlendirilecektir.

Tabu arama algoritması Glover (1995) tarafından ortaya çıkarılan iteratif bir arama algoritmasıdır. Metasezgisel algoritmalar çözüm uzayında arama yaparken dairesel hareketler yapabilmekte ve bazı durumlarda yerel minimum ve maksimum noktalara takılabilmektedir. Bu takılmayı ortadan kaldırmak için Tabu Arama algoritmasında bazı hareketler cezalandırılmaktadır. Böylece aynı çözümler etrafındaki tekrarlar azaltılmakta ve arama uzayında yeni keşifler yapabilmek mümkün olmaktadır.

Kötü sonuç veren bölgelerde daha fazla işlem yapılmaması (bu elemanların komşu sayılarının azaltılması), istenen çözüme daha az hesaplamayla, dolayısıyla daha hızlı ulaşılmasıyla sağlanmaktadır. İyi sonuç veren parametrelerin bir sonraki iterasyonda komşu sayıları artmakta, böylece algoritmanın verimliliği de yükselmektedir (Çayıroğlu, 2019).

Tabu arama algoritmasının işleyiş şekli kabaca Şekil 3.7'de gösterilmiştir. Bu algoritmaya ait temel parametreler ise aşağıda verilmiştir:

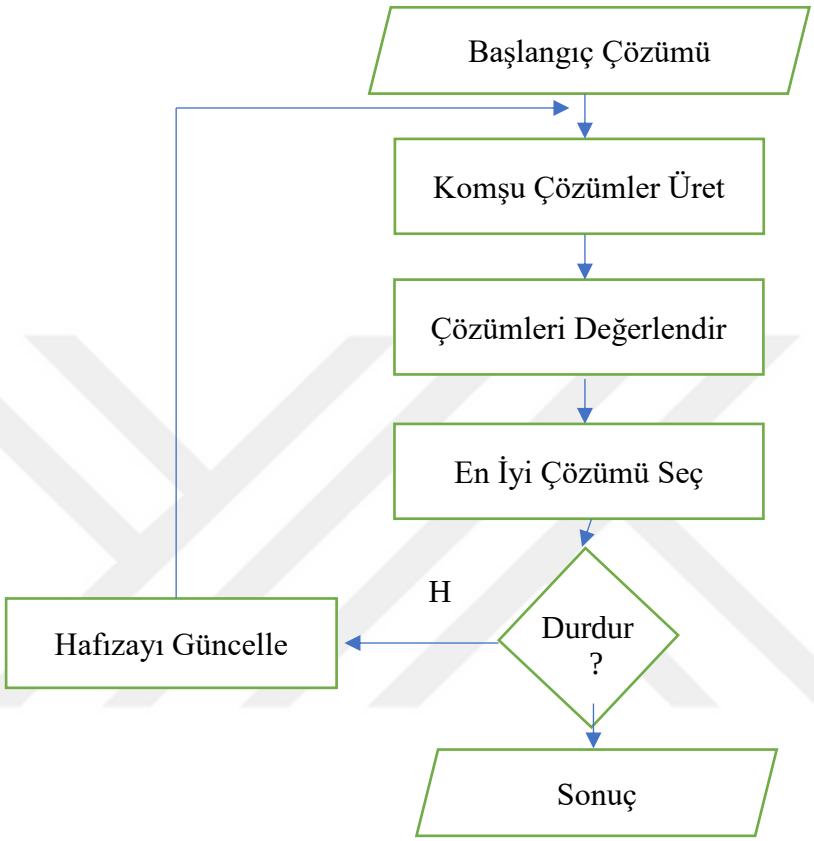
- Başlangıç çözümünün oluşturulması,
- Yeni çözüm, yani hareket mekanizması ile komşuluk yapısı,
- Hafıza,
- Tabu yıkma,
- Sonlandırma.

Başlangıç çözümü genellikle rastgele oluşturulmaktadır. Ancak burada iyileştirilmiş bir çözümle başlanmak istenirse sezgisel bir yaklaşımla başlangıç çözümü oluşturulabilmektedir. Başlangıç çözümü üzerinde yapılan bir değişiklik ile yeni aday çözümler elde edilmektedir. Bu tamamen problemin yapısına bağlıdır. Tabu Aramada en önemli adım aslında komşuluktur. Çünkü çözümü iyileştirmek için yeni çözümler bulunmalı ve en iyi çözümler korunmalıdır. Komşuluk bulma yapısı mevcut çözümün komşu çözümlerini üreterek algoritmanın çözüm uzayında gezinmesini sağlamaktadır.

Tabu arama algoritmasında tabu listesiyle korunan çözümlerin iyileştirilebilmesi için hafıza yapısına ihtiyaç vardır. Tabu listesi aslında kısa dönemli bir hafıza yapısıdır. Tabu listesi oluşturulurken her döngüdeki en iyi çözüm listeye alınmakta, listenin dolduğu durumda başlangıçtaki çözümler listeden atılmakta, son döngülerde elde edilen çözümler listeye alınmaktadır. Bu işlemin amacı her döngüdeki en iyi çözüme ulaşılmasını sağlayan hamleyi veya perturbasyonu saklayarak o hamlenin baskınlığını azaltmaktır. Uzun dönem hafıza yapısında ise bulunan en iyi, yani elit sonuçlar saklanmaktadır. Ayrıca sıklığa dayalı bellek yapısı da kullanılabilmektedir.

Tabu yıkma kriteri, tabu olarak kayda alınmış bir hareketin bu durumunu ortadan kaldıran kriteri ifade etmektedir. Yani en iyi çözümden daha iyi bir sonuç verecek bir hareket tabu listesinde olsa bile bu kriter sayesinde kabul edilebilmektedir. Durdurma koşulu ise tabu arama algoritmasının belirli bir iterasyona ulaşması sonucu veya komşu çözümlerin komşularının üretilememesi ya da belirli bir yerde tıkanması sonucu durdurma koşulu olarak kullanılabilir.

Tabu Arama Algoritmasının temsili çözümü aşağıda gösterilmiştir.



Şekil 3.7. Tabu arama algoritması (Çayıroğlu 2019)

3.2.5.a. Replika sunucu problemi için tabu arama (Tabu Search)

Tabu arama (TA) algoritmasını replika sunucu yerleştirme problemine uygularken öncelikle problemin temsil edilmesi gerekmektedir. Bu amaçla kullanıcı ve bulut nesneleri tanımlanmıştır. Bu nesneler kullanıcıların ve bulutların bilgilerini tutmaktadır.

Kullanıcı Nesnesi:
KULLANICI (USER_ID, CITY, LOAD, STORAGE) ,
şeklinde oluşturulmuştur. Denektaşından okunan kullanıcının bulunduğu konum ve aylık data kullanım verileri bu nesne üzerinde tutulmaktadır.

Bulut Nesnesi:
BULUT (CITY, PROVIDER, ISTHERE, ODP, SP) ,
şeklinde oluşturulmuştur. Uygulama başlangıcında bulut ücretlerinin olduğu metin dosyasından okunan bulut bilgileri bu nesne üzerinde tutulmaktadır.

Çözüm Temsili:
Çözüm temsili iki parçadan oluşmaktadır. Bu temsil için bir tamsayı dizisi kullanılmıştır. Dizinin uzunluğu kullanıcı sayısının iki katı olarak belirlenmiştir. Yani bu dizi, uzunluğunun ilk yarısı ve diğer yarısı olmak üzere iki parça şeklinde okunmalıdır. Dizinin indisleri kullanıcıları temsil etmektedir. Dizinin ilk yarısı içerisinde bulut indisleri, ilk yarıdaki her indisin öteleme simetrisindeki değerler olarak da hizmet sağlayıcı (provider) bilgilerinin indisleri tutulmaktadır.

Çizelge 3.16. Çözüm Dizisini Temsilen Gösteren Şekil

BULUT İNDİSLERİ			**HİZMET SAĞLAYICI**		
U1	U2	U3			
C1	C2	C3	P1	P2	P3

Çizelge 3.16. ile verilen çözüm temsili Çizelge 3.17'de gerçek değerlerle bir örneğe dönüştürülmüştür.

Çizelge 3.17. Çözüm Dizisini Örnek ile Gösteren Şekil

KULLANICI KONUMU - BULUT SERVİS SAĞLAYICI KONUMU			**KULLANICI KONUMU - BULUT SERVİS SAĞLAYICI KONUMUNDAKİ SERVİS SAĞLAYICI**		
0	1	2	3	4	5
barcelona	istanbul	lviv			
LONDRA	ATİNA	BERLİN	AMAZON	AZURE	GOOGLE

Çizelge 3.16 ve 3.17'ye bakılırsa 3 kullanıcı, 3 bulut sağlayıcı konumu ve 3 bulut sağlayıcısının olduğu gözükmektedir. Bu örnekte kullanıcı sayısı 3 ve buna bağlı olarak çözüm dizisinin uzunluğu da 6'dır. Çizelge 3.17'de bulunan Barcelona, İstanbul ve Lviv kullanıcı konumlarını, LONDRA, ATİNA ve BERLİN bulut sağlayıcı konumlarını ve AMAZON, AZURE ve GOOGLE ise bu bulut sağlayıcı konumlarında bulunan bulut servis sağlayıcı firmalarını temsil etmektedir. Yani dizinin ikinci yarısı, başka bir deyişle sağ tarafı, ilk kısımda bulunan bulut sağlayıcı konumlarına ait detay bilgiyi barındırmaktadır.

1. Başlangıç Çözümü

Başlangıç çözümünü oluşturmak için iki farklı rastgele yöntemden faydalanılmıştır: ilk yöntem dizinin sol yarısındaki her bir elemanına muhtemel bulut alanlarından birisinin id'sini rastgele atamaktadır. Bu durumda dizinin sağ yarısındaki her bir elemanına da bulut sağlayıcı firmalardan birisinin id'si rastgele atanmaktadır. İkinci yöntem dizinin sol ve sağ yarısında yer alan tüm elemanlarına rastgele seçilen bir bulut alanı id'si ile yine rastgele seçilen bir firma id'sini atamaktadır.

Hizmet Sağlayıcı İndisi							Hizmet Sağlayıcı Tipi						
3	2	4	2	3	1	0	2	1	2	2	0	1	0

Şekil 3.8. Örnek bir ilk çözüm dizisi

Şekil 3.8.'de oluşturulan dizi ile kullanıcı ve bulut nesneleri kullanılarak ilk temsil çözümün maliyeti maliyet fonksiyonu ile hesaplanmaktadır. Maliyet formülü LUG algoritmasından farklı olarak yerleştirilemeyen her kullanıcı için penaltı skoru eklemektedir.

$$\mathrm{C(ilk\ maliyet)} = \sum_{i \in F} W\alpha_i^S y_i + \sum_{i \in F}\sum_{j \in H} \omega_j \alpha_i^B X_{ij} + penaltı \quad (5)$$

Söz konusu penaltı skoru iki adettir. İlki (PM) çözümde yer alan bir bulut servis sağlayıcısının mevcudiyeti ile ilgilidir. Örneğin bir kullanıcı bir A şehrindeki bulut alanında mevcut olmayan bir sağlayıcıya atandığında bu penaltı skoru devreye girmektedir. İkincisi (PG) ise herhangi bir kullanıcının atandığı bulut sağlayıcı konumu ile ilgili kullanıcının konumu arasındaki uzaklığa bağlı olan gecikmenin ihlal edilmesi durumunda kullanılmaktadır.

Çizelge 3.7'de verilen örnek için maliyet hesaplaması aşağıda gösterilmiştir.
Penaltı skorları 1000 olarak tanımlanmıştır.

1. Adım

   U1: ZURICH | Aylık Kullanım Miktarı : 100 - 44

   C1: DUBLIN – GOOGLE | Ücret: ODP: 0,159 - SP: 0,023

   Maliyet(U1) = 100*0,159 + 44*0,023 + PM +PG

   Maliyet(U1) = 100*0,159 + 44*0,023 + 0 + 0

   M1: Maliyet(U1) = **16,912**

2. Adım

   U2: MOSCOW | Aylık Kullanım Miktarı : 80 - 55

   C2: WARSAW – AZURE | Ücret: ODP: 0,133 - SP: 0,022

   Maliyet(U2) = 80 *0,133 + 55*0,022 + PM + PG

   Maliyet(U2) = 80 *0,133 + 55*0,022 + 0 + 0

   M2: Maliyet(U2) = **11,85**

3. Adım

   U3: PALERMO | Aylık Kullanım Miktarı : 90 - 25

   C3: DUBLIN – AZURE | Ücret: ODP: 0, 133 - SP: 0, 0192

   Maliyet(U3) = 90 *0, 133 + 25*0, 0192 + PM + PG

   Maliyet(U3) = 90 *0, 133 + 25*0, 0192 + 0 + 1000

   M3: Maliyet(U3) = **1012,45**

4. Adım

   U4: GLASGOW | Aylık Kullanım Miktarı : 80 - 80

   C4: HAMINA – GOOGLE | Ücret: ODP: 0,159 -SP: 0, 02

   Maliyet(U4) = 80 *0,159 + 80 *0, 02 + PM + PG

   Maliyet(U4) = 80 *0,159 + 80 *0, 02 + 0 + 1000

   M4: Maliyet(U4) = **1014,32**

5. Adım

   U5: SZEGED | Aylık Kullanım Miktarı : 70 - 35

   C5: DUBLIN – AMAZON | Ücret: ODP: 0, 166-SP: 0,023

   Maliyet(U5) = 70 *0, 166+ 35*0, 023 + 0 + 1000

   M5: Maliyet(U5) = **1012,425**

6. Adım

   U6: DONETSK | Aylık Kullanım Miktarı : 90 - 45

   C5: VIENNA – AMAZON | Ücret: ODP: 0, 166-SP: 0,0238

   Maliyet(U6) = 90 *0,159 + 45*0, 0238 + PM + PG

   Maliyet(U6) = 90 *0,159 + 45*0, 0238 + 0 + 1000

M6: Maliyet(U6) = **1015,38**

7. Adım

   U7: BARCELONA | Aylık Kullanım Miktarı : 60 - 70

   C7: STOCKHOLM – GOOGLE | Ücret: ODP: 0,159 -SP: 0, 0226

   Maliyet(U7) = 60 *0,159 + 70*0, 0226 + PM + PG

   Maliyet(U7) = 60 *0,159 + 70*0, 0226 + 0 + 1000

   M7 : Maliyet(U7) = **1011,122**

8. Adım

   Maliyet(Toplam) = M1 + M2 + M3 + M4 + M5 + M6 + M7

   Maliyet(Toplam) = 16,912 + 11,85 + 1012,45 + 1014,32 + 1012,425+ 1015,381+ 1011,122

   Maliyet(Toplam) = **5080,14**

Çözüm dizisi aşağıdaki gibi bir *Solution* nesnesinde saklanacaktır.

SOLUTION (ADAYCOZUM:IntArray(), MOVE:Pair(KI, BI))

Bu nesne, komşuluk arama işleminde kullanılan komşuları saklarken aynı zamanda hangi hareket sonucu o komşuluğun elde edildiği bilgisini de tutar. Bu bilgi bir *Pair* nesnesinde *kullanıcı indisi (KI)* ve *bulut konumu indisi (BI)* şeklinde tutulur. Tabu listesi bu bilgiden yararlanarak oluşturulacaktır. İlk çözümün oluşturulmasının ardından komşuluk aramasına geçilmektedir.

Komşuluklar taranırken başlangıç çözümü komşu bul fonksiyonuna gönderilir. Bu fonksiyon gelen çözüm dizisi üzerinde ilk elemandan başlayarak bulut indislerini sırasıyla birer arttırarak komşu çözümler üretir. Bu arttırma işlemi genel olarak dizinin sol kısmı için yani bulut sağlayıcı konumları üzerinde yapılmaktadır. Arada bir sağ kısım yani bulut servis sağlayıcıları üzerinde de benzer bir işlemle komşu çözümler üretilmektedir. Bu durum bir sayaç yardımıyla belirlenmektedir. Bulunan tüm

komşuluklar *Solution* nesnesine atılarak bir *ArrayList* içerisinde saklanmaktadır. Aynı zamanda her komşuluğun *Pair*'leri de *Solution* nesnesine atanmaktadır.

Üretilen bu komşuların maliyetleri 4 numaralı formül ile hesaplanır. Daha önce anlatılan maliyet formülü ile komşu çözümlerin maliyetleri hesaplanırken o çözümdeki kullanıcıların atandıkları bulut konumlarına olan uzaklıklarına bakılır ve gecikme süresinin istenen değer aralığında olup olmadığı kontrol edilir. Bu gecikme süreleri daha önceki bölümlerde anlatıldığı şekilde 4 numaralı formül kullanılarak hesaplanır. Gecikme süresi istenen aralıkta ise maliyet normal hesaplanır. Değilse o çözüm cezalandırılır. Aynı şekilde kullanıcıların atandıkları bulut konumundaki servis sağlayıcının, o bulut konumundaki mevcudiyetine bakılır. Eğer bulut servis sağlayıcısı mevcut değilse çözüm yeniden cezalandırılmaktadır.

Algoritmanın başlangıcında, başlangıç çözümünün oluşturulması sonucu hesaplanan ilk maliyet ve bu maliyete sahip çözüm aynı zamanda için en iyi çözüm olarak kabul edilmiştir. Bu ilk çözüme ait komşu çözümlerin bulunmasının ardından maliyetleri hesaplanan komşu çözümler, maliyetlerine göre sıralanmaktadır. Sıralanan bu çözümlerden maliyeti en düşük olan çözüm o anki en iyi çözüm olarak kabul edilmekte ve ilk çözüm maliyeti ile karşılaştırılıp daha iyi bir çözüm olup olmadığı kontrol edilmektedir. Eğer daha iyiyse yeni en iyi çözümün bu komşu çözüm olması sağlanmaktadır. Bu komşuluğu sağlamak için yapılan hareket iki boyutlu olarak tanımlanan Tabu Listesinde işaretlenmektedir.

TABULİSTESİ[Kullanıcı Sayısı][Bulut Konum Sayısı]

TABULİSTESİ[Kullanıcı Sayısı][Servis Sağlayıcı Sayısı]

Tabu listesi iki adettir. İlkinde tabu listesinin ilk boyutunu kullanıcı sayısı, ikinci boyutunu bulut konum sayısı oluştururken, ikincisinde ilk boyutunu kullanıcı sayısı, ikinci boyutunu servis sağlayıcı sayısı oluşturmaktadır. Komşuluk fonksiyonunda

Solution nesnesi oluşturulurken kaydedilen *Pair(KI,BI)* bilgisi bu listedeki hangi indislerde işlem yapılacağı bilgisini vermektedir.

İlk çözüme ait komşu çözümler içerisindeki en düşük maliyetli çözüm eğer tabu listesinde değilse yeniden komşuluk fonksiyonuna iletilir. Yani bu çözüm oluşturulurken kaydedilen *Pair(KI,BI)* kullanıcı indis ve bulut konum indis bilgisi kullanılarak iki boyutlu tabu listesindeki değere bakılır. Bu değer sıfırdan farklı ise bu hareket tabu olmaktadır. Yani Örneğin Pair(2,3) gibi bir değere sahip olursa Tabu işlemi şöyle kontrol edilir.
Tabu listesi iki boyutlu bir dizi olduğu için Pair değerleri bu diziye ait boyutlardaki indisleri oluşturmaktadır. Yani:

TABULİSTESİ[2][3] = 0

ise bu indisler ile bulunan çözüm geçerlidir ve komşu aramaya bu çözüm ile devam edilebilir ve tabu listesinin bu elemanına aşağıdaki kullanıcı sayısı atanır, başka bir deyişle tabu tenure değeri olarak kullanıcı sayısı kullanılır.

TABULİSTESİ[2][3] = Kullanıcı Sayısı

Daha önce bulunan komşu en iyi çözüm tabu ise kendinden sonra gelen ve tabu olmayan en iyi çözüm ile yola devam edilmektedir. Bu işlem belli bir iterasyon boyunca tekrar etmektedir. Arama sonlanma kriteri olarak iterasyon sayısı kullanılmıştır. Bu belirlenen iterasyon boyunca en iyi çözüm aranır ve döngü sonunda eldeki en iyi çözüm, nihai çözüm olarak kabul edilmektedir.

Aşağıda önerilen Tabu arama algoritmasına ait sözde kod gösterilmiştir.

Algoritma Tabu Arama: İlk Çözüm Ve Sonuçlanma ait Pseudocode

Giriş: Denektaşı Dosyaları, Penaltı Skoru, Komşu Çözümleri Sol Yarıda Üret Limiti, Komşu Çözümleri Sağ Yarıda Üret Limiti, Gecikme Eşiği

Çıkış: Atama Dizisi

```
1.  Kullanıcı Sayısı, Bulut Konumu Sayısı ← ReadTheBenchmarkFiles()
1.  candidateNeighborsList ← arrayListOf<Solution>()
2.  bestNeighborsList ← arrayListOf<Solution>()
3.  solution       ← IntArray(Kullanıcı Sayısı x 2)
4.  bestSolution ← IntArray(Kullanıcı Sayısı x 2)
5.  tabuList1      ← IntArray[Kullanıcı Sayısı][Bulut Konumu Sayısı]
    tabuList2      ← IntArray[Kullanıcı Sayısı][3]
    sayac ← 0
6.  bestCost       ← Double
7.  ilkCozumIcinSecilenBulutKonumu ← Random.next(Bulut Konumu Sayısı.Size)
9.  for  i in 0 until KullanıcıSayısı
10.     |        Solution[i] ← ilkCozumIcinSecilenBulutKonumu
11. end for
12. ilkCozumIcinSecilenBulutSaglayici  ← Random.next(1,3)
13. for  i in Kullanıcı Sayısı until (Kullanıcı Sayısı x 2)
14.     |        Solution[i] ← ilkCozumIcinSecilenBulutSaglayici
14. end for
15. firstCost ← cost(Solution) //uygun servis sağlayıcı yoksa yada QoS sağlanmıyorsa penaltı ekle
16. bestCost ← firstCost
17. for  i in 0 until (Bulut Konum Sayısı x 100)
        |       sayac++
18.     |       getNeighbors(Solution)  // komşu çözümleri üretir
19.     |       tempBestSolution ← getBestNeighbors(candidateNeighborsList, tabuList )
20.     |       if (tempBestSolution.cost < bestCost  )
21.     |           |       bestNeighborsList  ← tempBestSolution.solution
22.     |           |       bestSolution   ←   tempBestSolution.solution
23.     |           |       bestCost       ← tempBestSolution.cost
24.     |           |       addTabuMove(tempBestSolution.move.first, tempBestSolution.move.second)
25.     |       end if
26.     |       solution  ←  tempBestSolution.solution
27. end while
28. return bestSolution
```

Şekil 3.9. Tabu arama algoritması uygulaması

getNeighbors() Alt Fonksiyonuna Ait ait Sözde Kod

```
     Giriş: Çözüm
     Çıkış: Çözüme Ait Komşular
1.   for i in 0 until Kullanıcı Sayısı - KS
2.       if sayac <= solLimit
3.           solution[i] ← solution[i]+1 % Bulut Konumu Sayısı
4.           candidateNeighborsList ← (solution[i] , Pair(i, solution[i]) )
5.       else if sayac <= (solLimit + sagLimit)
6.           solution[i+KS] = solution[i+KS]+1 % Servis Sağlayıcı Sayısı
7.           candidateNeighborsList ← (solution[i] , Pair(i, solution[i+KS]) )
8.       else
9.           sayac ← 0
10.      end if
11.  end for
```

Şekil 3.10. Tabu arama algoritmasının komşuluk yapısı

Tabu arama algoritmasının komşuluk bulma algoritması, en iyi çözümü elde etmek için önemli bir görev ifa eder. Komşulukları elde ederken oluşan aday çözümler çoğu zaman idealden uzak olmaktadır. Bu yüzden komşu arama algoritması verimliliği en çok etkileyen kısımdır.

Maliyet Fonksiyonuna Ait Sözde Kod

```
     Giriş: Çözüm
     Çıkış: Çözümün Maliyeti
1.   for i in 0 until KullanıcıSayısı(KS)
2.       Maliyet + ← BulutListesi[i].ODP  * Kullanıcı_Listesi[i].LOAD
3.       Maliyet + ← BulutListesi[i].SP   * Kullanıcı_Listesi[i].STORAGE
4.           if Not BulutListesi[i].ISTHERE
5.               Maliyet + ← PENALTI SKOR
6.           end if
7.           if Not checkQoS()
8.               Maliyet + ← PENALTI SKOR
9.           end if
10.  end for
11.  return Maliyet
```

Şekil 3.11. Tabu arama algoritmasının maliyet hesabı

Sonuç Değerlendirme Fonksiyonuna Ait Sözde Kod

```
      Giriş: Çözüm
      Çıkış: Çözümün Değerlendirilmesi Sonucu Rapor
1.    for i in 0 until KullanıcıSayısı(KS)
          if servisSaglayiciUygunMu(solution[i])
2.            saglayiciSayisi ← saglayiciSayisi + 1
3.        end if
4.        if gecikmeEsigiAsilmisMi(solution[i])
5.            gecikmeIhmalSayisi ← gecikmeIhmalSayisi + 1
6.        end if
7.    end for
8.    if saglayiciSayisi > 0
9.        println("Var olmayan servis sağlayıcı eşleştirmesi yapılmış. Tekrar deneyiniz")
10.   end if
11.   if gecikmeIhmalSayisi > 0
12.       println("Bazı kullanıcılar gecikme eşiği uygun olmayan bulut alanlarına atamıştır. QoS
          eşiğini değiştirip yeniden çalıştırınız")
13.   end if
```

Şekil 3.12. Sonuç değerlendirme fonksiyonuna ait sözde kod

Şekil 3.0.8'de gösterildiği şekilde üretilen her bir komşuluğun maliyeti maliyet fonksiyonu aracılığıyla hesaplanmaktadır. Maliyet fonksiyonunda sezgisel yaklaşımdan farklı olarak mevcut çözümde atanan herhangi bir kullanıcı için tercih edilen şehirde bulut sağlayıcısının olmaması durumunda bir ceza skoru eklenmektedir. Ayrıca QoS eşiği olarak da mevcut çözümdeki kullanıcılar QoS eşiğinin içerisindeki bir bulut sağlayıcıya atanmıyorsa çözüm yeniden cezalandırılmaktadır. Böylelikle ideal çözüme yakın çözümler öne çıkarılmaktadır.

Çizelge 3.18. Örnek çözümün kullanıcı-bulut konum gösterimi

Koyu Yazılan Şehirler Kullanıcı Konumları
Altlarındaki Şehirler Atandıkları Konumları Belirtir

Kullanıcı Konumları	**U1** **ZURICH**	**U2** **MOSCOV**	**U3** **PALERMO**	**U4** **GLASGOW**	**U5** **SZEGED**	**U6** **DONETSK**	**U7** **BARCELONA**
Bulut Konumu	DUBLIN	WARSAW	DUBLIN	HAMINA	DUBLIN	VIENNA	STOCKHOLM
Bulut Tipi	GOOGLE	AZURE	AZURE	GOOGLE	AMAZON	AMAZON	GOOGLE

Yukarıda verilen örnek çözümün komşulukları ya her bir kullanıcının mevcuttaki bulut servis sağlayıcısı konumunu, yani Dublin'i Vienna yapmak ya da bulut sağlayıcısını Google iken Amazon yapmak şeklinde çalışmaktadır. Bu varyasyonlar çözüm uzayında dolaşmayı sağlamaktadır. Tüm bu adımların tamamlanması sonucu üretilen ilk çözüme bağlı olarak beklenenden daha iyi veya kötü sonuçlar çıkabilmektedir.

Çizelge 3.20. Çizelge 3.19 ait bir komşu çözüm örneği

Koyu Yazılan Şehirler Kullanıcı Konumları
Altlarındaki Şehirler Atandıkları Konumları Belirtir

Kullanıcı Konumları	**U1** **ZURICH**	**U2** **MOSCOV**	**U3** **PALERMO**	**U4** **GLASGOW**	**U5** **SZEGED**	**U6** **DONETSK**	**U7** **BARCELONA**
Bulut Konumu	***VIENNA***	WARSAW	DUBLIN	HAMINA	DUBLIN	VIENNA	STOCKHOLM
Bulut Tipi	GOOGLE	AZURE	AZURE	GOOGLE	AMAZON	AMAZON	GOOGLE

Tez çalışmasının buradan sonraki bölümünde tez kapsamında önerilen metasezgisel yaklaşımın ve gerçeklenen sezgisel algoritmanın simülasyon ortamında elde edilen performansları tartışılacaktır.

4. ARAŞTIRMA BULGULARI ve TARTIŞMA

Çalışmanın ilk aşamasında algoritmalarda kullanılan maliyet fonksiyonu için gerekli olan fiyat bilgisini belirlemek için öncelikle bulut servis sağlayıcıları belirlenmiştir. Bu servis sağlayıcılar belirlendikten sonra veri seti oluşturmak için kullanıcı konumları ve bulut servis sağlayıcı konumları bilgileri alınıp gecikme verisi elde edilmiştir. Sonrasında replika sunucu yerleşimi problemi modellenip algoritmalar oluşturulmuştur. Kıyaslama metriği olarak tez kapsamında referans olarak kullanılan LUG sezgisel algoritması ile;

- kullanıcı sayısı – işlem süresi,
- kullanıcı sayısı – maliyet parametreleri kıyaslanmıştır.

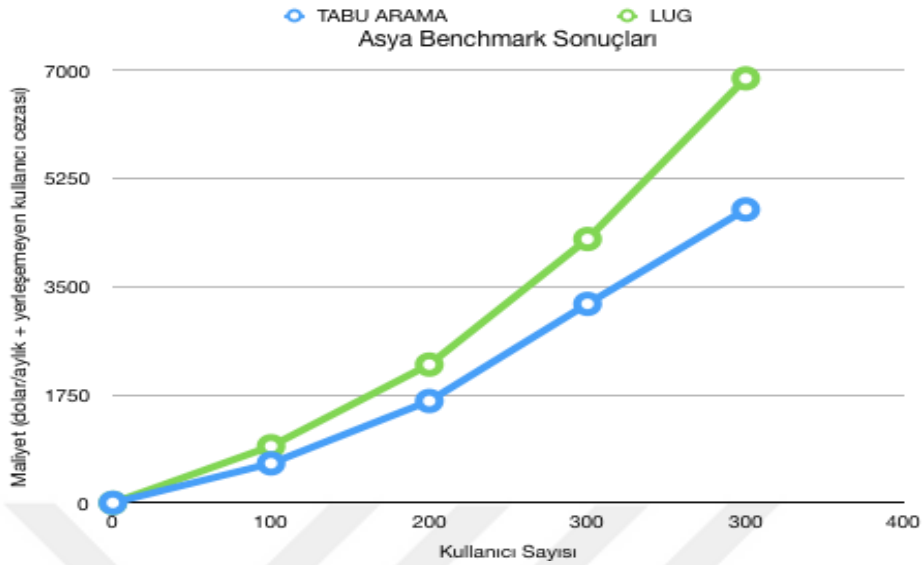
Yapılan testlerde metasezgisel yaklaşım ile sezgisel algoritmadan daha düşük maliyetli sonuçlar almanın mümkün olduğu görülmüştür. Tabu arama algoritması bir çok kez sezgisel algoritmadan daha az maliyetli sonuç çıkarmayı başarmıştır. Ama bazı durumlarda çözümlerin maliyeti minimum olsa da gecikme eşiği değerini ihmal ettiği tespit edilmiş yani fizibil sonuçlar ortaya çıkmamıştır.

Algoritmaların çözüm sürecindeki kritik fonksiyonlarının işlem süreleri de ölçülmüş ve verimlilik açısından karşılaştırılmıştır. Üç ana denektaşından oluşan deney düzeneğinde Avrupa, Asya ve Kuzey Amerika'da belirlenmiş kullanıcı konumları ve üç büyük bulut servis sağlayıcısının bulut konumları ve fiyat bilgileri mevcuttur. Her bir kıta 100, 200 kullanıcının oluşturulduğu farklı üç denektaşı bulundurmakta ve bu denektaşlarının karşılaştırılması aşağıda yapılmaktadır.

Denektaşı	LUG	TABUARAMA	QoS Eşiği
Bench_Asia	781,716	639,85	45ms
Bench_Asia_100	1897,66	1683,60	45ms
Bench_Asia_200	3553,01	3281,89	45ms
Bench_EU	325,99	319,08	45ms
Bench_EU_100	1654,86	1620,26	45ms
Bench_EU_200	3341,10	3225,04	45ms
Bench_USA	631,87	536,87	45ms
Bench_USA_100	2351,31	2351,31	45ms
Bench_USA_200	5034.35	5034,35	45ms

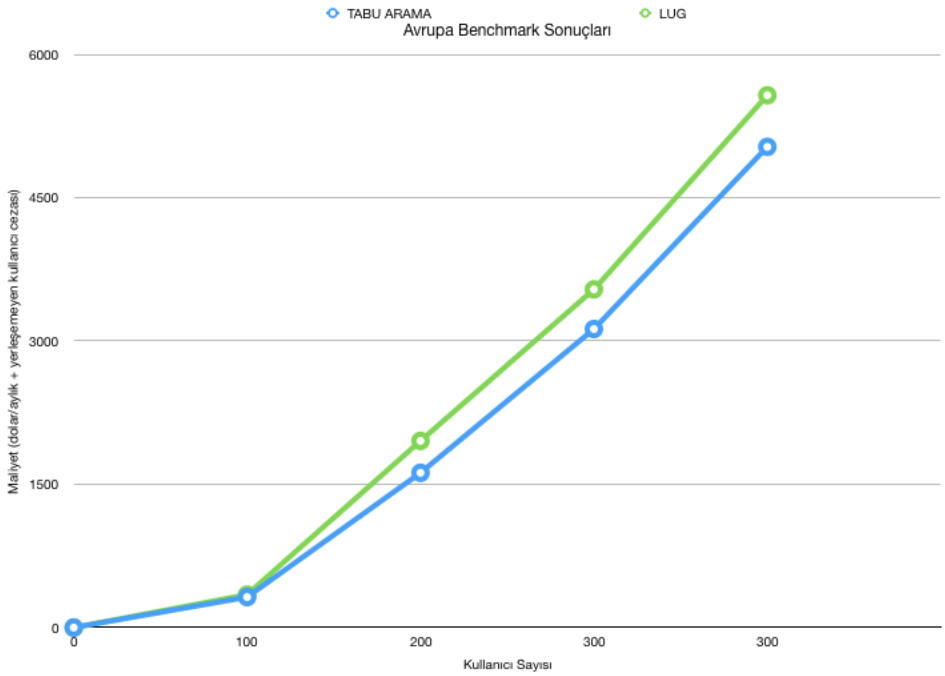
Şekil 4.1. Asya, Avupa ve Kuzey Amerika denektaşları

Şekil 4.1. incelendiğinde testlerin yapıldığı QoS eşiğinin 45 ms olduğu görülmektedir. Yani bir kullanıcı ile atandığı bulut konumu arasındaki gecikme 45 ms'yi geçmemelidir. Bu şartlar altındaki testlerde Tabu Arama Algoritması daha büyük bir çözüm uzayı taradığı için sezgisel LUG algoritmasına göre daha iyi çözümler bulmayı başarmıştır. Tabi tüm bu bulunan sonuçlar defalarca tekrar edilen deneylerin (her denektaşı 30 kere çalıştırıldı) içinde en iyileri alınarak oluşturulmuştur. QoS eşiği daha yüksek seviyelere çekildikçe bazı durumlarda sezgisel algoritmanın Tabu Arama algoritmasına daha yakın sonuçlar verdiği gözlemlenmiştir.



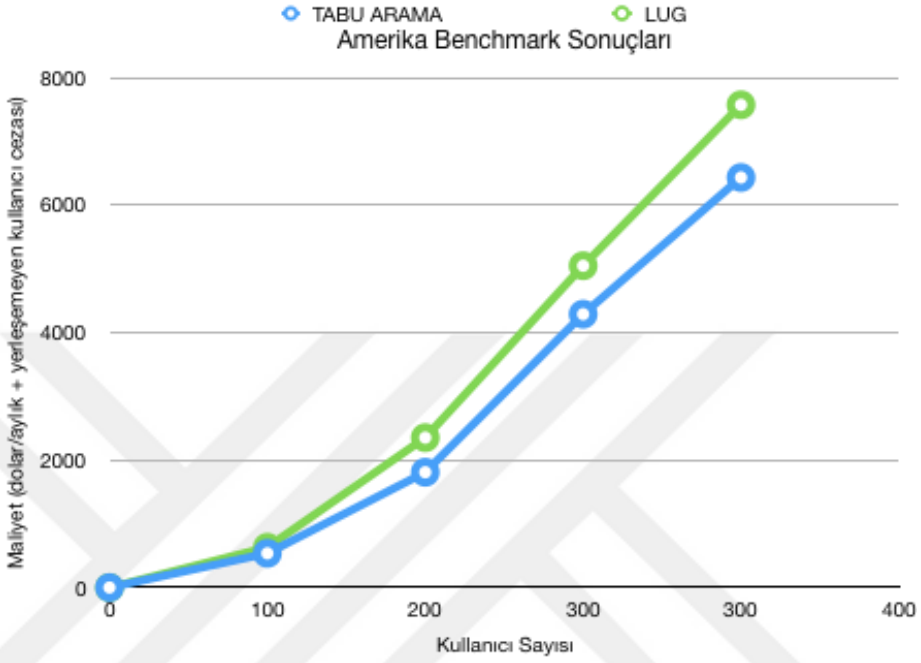
Şekil 4.2. Kullanıcı Sayısına Göre Maliyet

Şekil 4.2. incelendiğinde LUG sezgisel algoritması ve Tabu arama metasezgisel algoritması arasındaki maliyet farkı açık şekilde görülebilmektedir. Tabu arama algoritması çözüm uzayının daha büyük bir kısmını taradığı için sezgisel LUG'a göre daha iyi sonuçlar bulmayı başarmıştır.



Şekil 4.3. Kullanıcı sayısına göre maliyet (Avrupa)

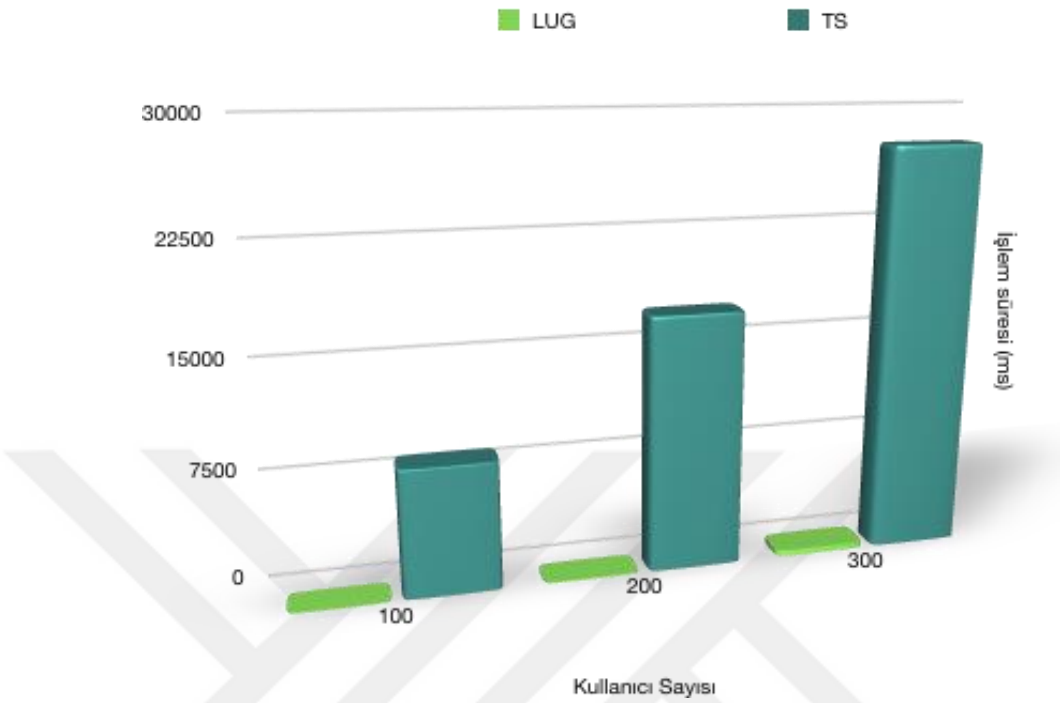
Şekil 4.3. de kullanıcı sayısı artması ile oluşan maliyetlerde Tabu Arama algoritması sezgisel algoritmaya göre daha iyi sonuçlar bulmuştur.



Şekil 4.4. Kullanıcı sayısına göre maliyet (Kuzey Amerika)

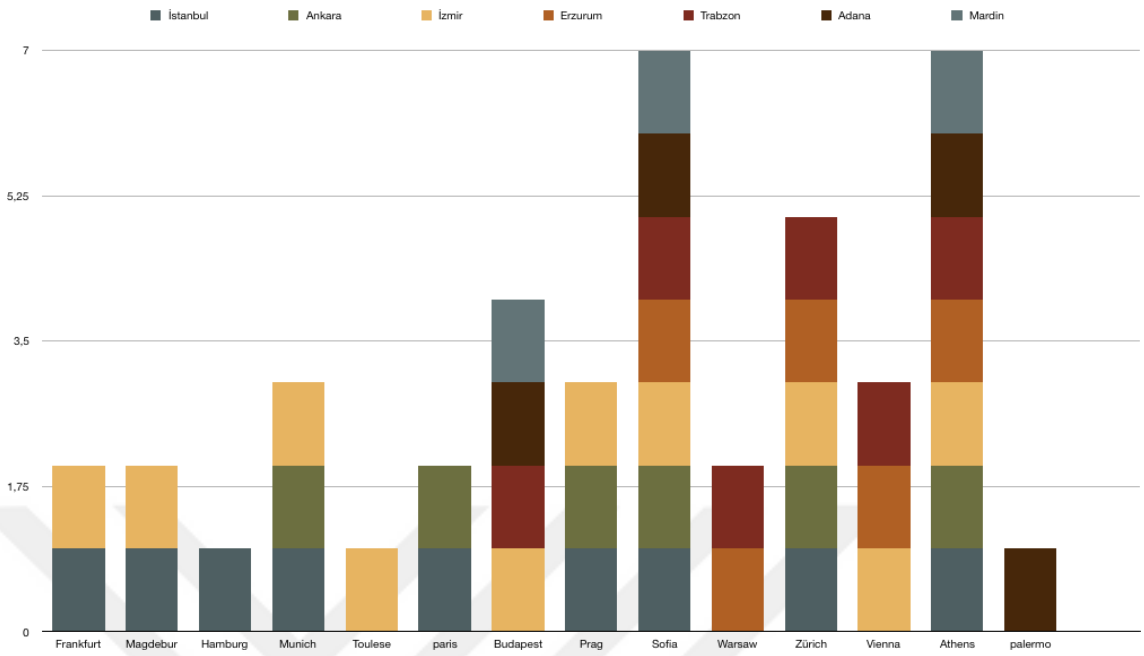
Şekil 4.4.'de Kuzey Amerika verileri incelendiğinde kullanıcı konumlarının bulut konumlarına olan uzaklıkları QoS eşiğinin altında olduklarından maliyet hesaplamaları sonucunda ceza oluşmamış ve maliyet nispeten paralel ilerlemiştir. Problem farklı şekilde yorumlanıp komşu çözümler üreten yapı iyileştirilirse çalışma zamanı olarak da başarılı bir algoritma ortaya çıkacaktır.

Şekil 4.2.'ye bakıldığı zaman kullanıcı sayısına bağlı olarak aylık maliyeti en avantajlı sonuçları üreten algoritma yine metasezgisel algoritmadır. Tabu Arama yaklaşımı LUG'a göre aylık ortalama 60 dolar daha az maliyetlidir. Algoritmaları denemek için yapılan testlerde temsili veriler kullanıldığından için bu altmış dolar, yüzbinleri hatta milyonları bulan kullanıcı sayılarına çıktığı zaman on bin dolarları bulacak ciddi bir tasarruf sağlayacaktır. Tabi ceza puanlarının sadece uzaklığa bağlı gecikme eşiğinden kaynaklandığı durumlarda bu yorum yapılabilir.



Şekil 4.5. Kullanıcı sayısına göre algoritmaların işlem süresi

Şekil 4.5. incelendiğinde sezgisel yaklaşımla metasezgisel yaklaşım arasındaki işlem süresi farkını çok daha net görebiliyoruz. Çünkü Tabu arama algoritmasında problem temsili, kullanıcı sayısı boyutundaki dizilerden oluşmakta ve bu dizilerin kopyalanması ve sıralanması işlemlerindeki döngüler sebebiyle daha fazla işlem süresi olarak geri dönmektedir. Bu süre performans açısından kayıplar doğurmaktadır. İşlem süresinin daha büyük veri setleri için daha da uzun süreler anlamına geldiği açıktır. Ancak sezgisel yaklaşım birkaç döngüde adımları takip edip tek seferde bitirebilmektedir. Yapısı gereği az işlem yaptığı ve probleme özgü olduğu için çalışma süresi daha azdır.



Şekil 4.6. Türkiye'deki bazı şehirlerin QoS eşiğindeki bulut bağlantıları

Şekil 4.6'da görüldüğü üzere Türkiye özelinde metasezgisel algoritma çalıştırılmıştır. Bu denemelerde QoS eşiği 45ms olarak belirlendiğinde yukarıdaki bulut alanları tercih edilmiştir. Bu 40'dan fazla kez test edilmiş ve test sonucunda en çok Atina, Sofia ve Zurich şehirlerindeki bulut alanları Türkiye'deki şehirler için uygun görülmüştür. Bu şehirler içerisinde de çoğunlukla maliyet avantajından dolayı Azure bulut alanları öne çıkmıştır.

5. SONUÇ ve ÖNERİLER

Veri tüketiminin hızla arttığı günümüzde internetten temin edilen verinin büyük kısmını akış servis hizmeti sağlayan servislerin sunduğu video ve müzik dosyaları oluşturmaktadır. Bu hızlı akış karşısında kullanıcı memnuniyetini sağlamak için geliştirilen bulut tabanlı içerik dağıtım ağlarında, replika sunucu yerleşimi hayati bir önem taşımaktadır. Bu probleme literatürde uygulanan sezgisel çözümlerin aksine metasezgisel bir çözüm geliştirilerek daha düşük maliyetli çözümler üretilebileceği ispatlanmıştır.

Bu probleme literatürde temel olarak iki farklı yaklaşım ile yani online veya offline çalışan algoritmalar ile çözümler uygulanmıştır. Bu çalışma kapsamında sunulan metasezgisel algoritma çalışma zamanı bakımından verimsiz gibi gözükmektedir. Yani mevcut problem online talep anında karar verilmesi mecbur olunan bir problem olsaydı en iyi bulut alanı araması esnasında QoS eşiği rahatlıkla aşılacak ve kullanıcı deneyimi olumsuz etkilenecekti. Ancak gelen kullanıcı taleplerinin gerçek zamanlı olmadan analiz için offline şekilde kullanılabilmesi yalnızca Tabu Arama algoritması için değil diğer metasezgisel yaklaşımlarında bu problem üzerinde çalışabileceğini göstermektedir.

Bu tez kapsamında problemlerin modellenip test uygulamasının yazılımında Kotlin programlama dili kullanılmıştır. Bu dilin sahip olduğu esneklik ve pratiklik uygulamanın yazılım ve test sürecinde kolaylık sağlamıştır.

Daha önce literatürde kullanılan çözümlerin sezgisel olmasından dolayı metasezgisel çözümlerin gerekli olup olmayacağı bilinmiyordu. Bu yüzden tez kapsamında daha önce gerçekleştirilen çevrim dışı (offline) sezgisel algoritmaları analiz edip onlardan daha iyi bir çözüm sunduğunu söyleyen Sahoo ve Glitho (2016)'nun geliştirdiği LUG algoritması referans alınmıştır. Test sonuçlarına önerilen algoritmanın sunucu maliyeti noktasında daha başarılı çözümler üretebildiği ortaya çıkmıştır.

Çalışmanın en başında referans alınan sezgisel algoritmanın ürettiği sonuçtan daha iyi bir sonuç mevcut mudur sorusuna cevap bulabilmek için 5 bulut alanı ve 10 kullanıcıdan oluşan bir deney düzeneği hazırlanmıştır. Bu 5 bulut alanının optimum yerleşimi tüm ihtimaller, yani 5^{10} (9.765.625) durum tek tek denenerek elde edilmiştir. Optimum çözümün LUG algoritmasından çok daha düşük maliyetli bir çözüm olduğu gözlemlenmiştir. Bunu sonucunda da meta sezgisel algoritmaların bu gözlemlenen sonucu yakalayabileceği motivasyonu ile problem modellenmiştir, uygulanmıştır ve test edilmiştir.

Ayrıca literatürde yapılan çalışmalar incelendiğinde yapılan testlerin rastgele oluşturulmuş veri setleri ile gerçeklendiği ve bu verilerin paylaşılmadığı gerçeği ortaya çıkmıştır. Bu çalışma kapsamında bu boşluğu doldurabilmek amacıyla herkesin kullanabileceği denektaşları oluşturulmuştur ve literatüre kazandırılmıştır.

Son olarak elde edilen sonuçlar ile kullanılan algoritmaların daha rafine hale gelmesi sağlanarak maliyetlerin daha da düşürebileceği kanısına varılmıştır. Gelecekte yapılacak çalışmalarda komşuluk yapısının geliştirilmesi ve bu sayede maliyeti daha düşük çözümler üretebilen bir metasezgiselin tasarlanabilmesi hedeflenmektedir.

KAYNAKLAR

Anonim, Global Digital Report , https://digitalreport.wearesocial.com 2018

Anonim, Amazon Inc., Bulut Bilişim Nedir ?, https://aws.amazon.com/tr/what-is-cloud-computing/, 29.05.2019a

Anonim, 2019b. Microsoft, Bulut Bilişim Nedir ?, https://azure.microsoft.com/tr-tr/overview/what-is-cloud-computing/, 29.05.2019b

Anonim, "AWS | Amazon CloudFront CDN - Content Delivery Network & Streaming," Amazon Web Services, Inc. , https://aws.amazon.com/cloudfront/,2019c

Anonim Doogal, https://www.doogal.co.uk/MeasureDistances.php, 26.09.2019d

Anonim Google Cloud CDN, https://cloud.google.com/cdn/?hl=tr, 26.09.2019e

Anonim Amazon Cloud CDN, https://aws.amazon.com/tr/cloudfront/, 26.09.2019f

Anonim Microsoft Cloud CDN, https://azure.microsoft.com/tr-tr/services/cdn/, 26.05.2019g

Cayıroglu, İ., 2019. "İleri Algoritma Analizi", http://www.ibrahimcayiroglu.com/Dokumanlar/IleriAlgoritmaAnalizi/IleriAlgoritmaAnalizi-9.Hafta-TabuAramaAlgoritmasi.pdf, (28.05.2019)

Cayıroglu, I., 2019. "Optimizasyon Teknikleri", http://www.ibrahimcayiroglu.com/Dokumanlar/OptimizasyonTeknikleri/OptimizasyonTeknikleri-1.Hafta-C_Sharp_Temelleri.pdf, (10.05.2019)

Chen, F., Guo, K., Lin, J., and La Porta, T., 2010. "Replica Placement in Storage Cloud-based CDNs," Network and Security Research Center, Penn State University, Tech. Rep., http://www.cse.psu.edu/~fachen/cdn.pdf ,

Chen, F., Guo K., Lin, J. and La Porta, T., 2012. "Intra-cloud Lightning: Building CDNs in the Cloud," in Proceedings of IEEE INFOCOM, Orlando, Forida,

Cohen, R., Lewin-Eytan, L., Naor, J. and Raz, D., 2015. "Near optimal placement of virtual network functions," in 2015 IEEE Conference on Computer Communications (INFOCOM) , pp.1346-1354,

Denektaşı Setinin Paylaşım Adresi, https://github.com/Captainwoof/Benchmark-Files/, 02.11.2019.

Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R. and Weihl, B., 2002. "Globally distributed content delivery," Internet Computing, IEEE, vol. 6, no. 5, pp. 50-58,

Douglis, F. and Kaashoek, M.F., 2001. "Scalable Internet Services," IEEE Internet Computing, vol. 5, no. 4, pp. 36–37.

Glover, F. 1995. "Tabu Search Fundamentals And Uses",

Hosanagar, K. et al., 2004. "Optimal pricing of content delivery network services. In Proceedings of the 37th International Conference on System Sciences (Big Island, Hawaii), Jan.

Hu, M., Luo, J., Wang, Y. and Veeravalli B., 2014."Practical Resource Provisioning and Caching with Dynamic Resilience for Cloud- Based Content Distribution Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 8, pp. 2169-2179,

Hu, N. ve Steenkiste, P., 2003. "Evaluation and characterization of available bandwidth probing techniques," Selected Areas in Communications, IEEE Journal on, vol. 21, no. 6, pp. 879-894, 2003

Jamin, S., Jin, C., Kurc, A. R., Raz, D., and Shavitt, Y., 2001. “Constrained mirror placement on the Internet,” in INFOCOM, pp. 31–40.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., 1983. “Optimization by Simulated Annealing”, Science, New Series, Vol. 220, No. 4598, 1983

Küçükkoç, İ., 2019. “Optimizasyona Giriş & Temel Kavramlar, Excel Solver Kurulumu veKullanımı,Örnekler”,http://ikucukkoc.baun.edu.tr/lectures/EMM3208/EMM3208W2.pdf, (10.05.2019)

Laoutaris N., Smaragdakis, G., Oikonomou, K., Stavrakakis, I. and Bestavros, A., 2007. “Distributed Placement of Service Facilities in Large-Scale Networks," in Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM), pp.2144-2152, 6-12 May.

Mobasher, B., Cooley, R., Srivastava, J., 2000 . “Automatic personalization based on Web usage mining.”

Mokhtarian, K. and Jacobsen, H. A., 2015. "Server Provisioning in Content Delivery Clouds," in Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on, pp.65-72, June 27 2015- July 2.

Myers, A., Dinda, P. and Zhang, H., 1999. “Performance characteristics of mirror servers on the internet,” in Proc. IEEE INFOCOM, Mar., pp. 304–312.

Papagianni, C., Leivadeas, A. and Papavassiliou, S., 2013. "A Cloud- Oriented Content Delivery Network Paradigm: Modeling and Assessment," IEEE Transactions on Dependable and Secure Computing, vol. 10, no. 5, pp. 287-300,.

Rappaport, A. and Raz, D., 2013. ”Update aware replica placement,” in International Conferenceon Network and Service Management (CNSM), Zurich.

Salahuddin, A., Sahoo, J., Glitho R., Elbiaze, H., Ajib, W., 2017. “A Survey on Content Placement Algorithms for Cloud-Based Content Delivery Networks”, IEEE Access,

Sahoo, J. ve Glitho, R., 2016. “Greedy Heuristic for Replica Server Placement in Cloud basedContent Delivery Networks”, IEEE Symposium on Computers and Communication

Sahoo, J., Salahuddin, M. A., Glitho, R., Elbiaze H. and Ajib, W., 2016. “A Survey on Replica Server Placement Algorithms for Content Delivery Networks” IEEE Communications Surveys and Tutorials

Şeker, Ş. E., 2019. “Sezgisel Algoritmalar”, http://bilgisayarkavramlari.sadievrenseker.com/2008/12/22/sezgisel-algoritmalar-bulussal-algoritmalar-heuristic-algorithms/, (25.05.2019.)

T. S. Eugene Ng ve H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in IEEE Infocom, pp. 170-179, vol.1, 2002.

Vakali, A. and Pallis, G., 2003. “Content Delivery Networks: Status and Trends”
D. Shmoys, E. Tardos and K. Aardal, Approximation algorithms for facility location problems, In 29th ACM Symposium on Theory of Computing, pages 265-274,.

Wang, F., Liu, J., Chen, M. and Wang, H., 2014 "Migration Towards Cloud-Assisted Live Media Streaming," IEEE/ACM Transactions on Networking,

Zhang, Q., Zhu, Q., Zhani, M., Boutaba, R. and Hellerstein, J., 2013 “Dynamic Service Placement in Geographically Distributed Clouds,” Selected Areas in Communications, IEEE Journal on, vol. 31, no. 12, pp. 762-772.

ÖZGEÇMİŞ

Ömer Faruk YILDIRIM, 1991 yılında Erzurum'da doğdu. İlk ve orta öğrenimini Sabancı İlköğretim Okulu'nda tamamladı. 2005 yılında Erzurum Merkez Anadolu Lisesi'nde lise eğitimine başladı ve 2009 yılında aynı liseden mezun oldu. 2010 yılında Yalova Üniversitesi Bilgisayar Mühendisliği bölümünü kazandı. 2015 yılında aynı üniversiteden Bölüm 3.'sü olarak mezun oldu.