T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM TANIMLI AĞLARDA GENETİK ALGORİTMA TABANLI ÇOKLU DENETLEYİCİ YERLEŞTİRME

Hazırlayan
Eda Nur HASÇOKADAR

Danışman
Doç. Dr. Bilal BABAYİĞİT

Yüksek Lisans Tezi

Haziran 2021
KAYSERİ

T.C.
ERCİYES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YAZILIM TANIMLI AĞLARDA GENETİK ALGORİTMA TABANLI ÇOKLU DENETLEYİCİ YERLEŞTİRME

(Yüksek Lisans Tezi)

Hazırlayan
Eda Nur HASÇOKADAR

Danışman
Doç. Dr. Bilal BABAYİĞİT

Bu çalışma; Erciyes Üniversitesi Bilimsel Araştırma Projeleri Birimi tarafından FYL-2019-9299 proje koduyla desteklenmiştir.

Haziran 2021
KAYSERİ

BİLİMSEL ETİĞE UYGUNLUK

Bu çalışmadaki tüm bilgilerin, akademik ve etik kurallara uygun bir şekilde elde edildiğini beyan ederim. Aynı zamanda bu kural ve davranışların gerektirdiği gibi, bu çalışmanın özünde olmayan tüm materyal ve sonuçları tam olarak aktardığımı ve referans gösterdiğimi belirtirim.

Eda Nur HASÇOKADAR

“Yazılım Tanımlı Ağlarda Genetik Algoritma Tabanlı Çoklu Denetleyici Yerleştirme adlı yüksek lisans tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesine’ ne uygun olarak hazırlanmıştır.

Hazırlayan	**Danışman**
Eda Nur HASÇOKADAR	Doç. Dr. Bilal BABAOĞLU

Bilgisayar Mühendisliği ABD Başkanı

Prof. Dr. Veysel Aslantaş

TEŞEKKÜR

Çalışmalarım boyunca bilimsel araştırma teknikleri ve bilgisi ile bana her konuda yardımcı olan, ilgi ve yardımlarını hiç esirgemeyen değerli danışmanım Doç. Dr. Bilal BABAYİĞİT' e çok teşekkür ederim.

Hayatım boyunca her alanda benden desteklerini esirgemeyen ve eğitimim için daima arkamda duran ailem; Fikriye HASÇOKADAR, Bekir HASÇOKADAR ile APAYDIN ve KARADAYI ailelerine çok teşekkür ederim.

Çalışmalarım süresince sabırla beni destekleyen yol arkadaşım Mustafa SEZEN'e teşekkürederim.

Yüksek lisans eğitimim süresince bana destek olan ve bilgilerini paylaşmaktan hiç kaçınmayan iş ailem Kayseri şeker Fabrikası Yazılım Departmanı çalışanlarına ve sevgili bilgi teknolojileri müdürüme, üniversiteye iş günlerinde gelip tez çalışmamı sürdürmeye yardımcı olduğu için yönetim ekibimize çok teşekkür ederim.

Eğitimim süresince çalışmalarıma yardımcı olmuş tüm bölüm hocalarıma teşekkür ederim.

Eda Nur HASÇOKADAR

Haziran 2021, KAYSERİ

YAZILIM TANIMLI AĞLARDA GENETİK ALGORİTMA TABANLI ÇOKLU DENETLEYİCİ YERLEŞTİRME

Eda Nur HASÇOKADAR

Erciyes Üniversitesi, Fen Bilimleri Enstitüsü
Yüksek Lisans Tezi, Haziran 2021
Danışman: Doç. Dr. Bilal BABAYİĞİT

ÖZET

Geleneksel ağlar, gelişen teknoloji ve işlenecek veri boyutunun sürekli artmasıyla ihtiyaçları karşılamada zorlanmaktadır. Yazılım Tanımlı Ağ (YTA) yaklaşımı, geleneksel ağlara bir alternatif olarak ortaya çıkmıştır. YTA, kontrol ve veri düzlemini birbirinden ayırır, ağı esneklik ve maliyet açısından sağladığı avantajlar ile kontrol düzlemi üzerinden yönetir. Büyük veri akışına sahip ağlarda tek bir denetleyici ile ağı yönetmek kesintilerine ve veri kaybına neden olabileceğinden çoklu denetleyici kullanmak yararlı olacaktır. Çoklu denetleyici kullanılan YTA'larda denetleyicilerin yerleştirilmesi başlıca problemler arasında yer almaktadır.

Bu tez çalışması YTA'larda çoklu denetleyici yerleştirme problemini çözmek üzere gerçekleştirilmiştir. Topology Zoo veri tabanı bünyesindeki Colt ve ULAKNET veri setlerine Genetik Algoritma ile Dijkstra Algoritması birlikte uygulanarak hem Avrupa hem de Türkiye için denetleyicilerin en iyi konumu belirlenmiştir. Ayrıca Türkiye'de kurulması gereken pandemi hastaneleri için YTA modellemesi sunulmuştur. Sunulan modellemeye göre pandemi hastanelerinin kurulacağı şehirler kontrol katmanına karşılık gelirken bu hastanelerden yararlanacak olan diğer tüm şehirler veri katmanına karşılık gelmektedir. Pandemi hastanelerinin kurulum maliyeti dikkate alındığında stratejik noktalarda konumlandırılması, bu hastanelere daha geniş bir kapsama alanı sağlayacaktır. Pandemi hastanelerinden maksimum fayda sağlanabilmesi için koronavirüslü hasta sayısı yoğun fakat diğer tüm şehirlere minimum uzaklıktaki lokasyonlara hastane kurulması önerilmektedir.

Anahtar Kelimeler: Yazılım Tanımlı Ağ, Çoklu Denetleyici Problemi, Genetik Algoritma, COVID-19, Pandemi Hastanesi

GENETIC ALGORITHM-BASED MULTI-CONTROLLER PLACEMENT IN SOFTWARE-DEFINED NETWORKS

Eda Nur HASÇOKADAR

Erciyes University, Graduate School of Natural and Applied Sciences
M.Sc. Thesis, June 2021
Supervisor: Assoc. Prof. Dr. Bilal BABAYİĞİT

ABSTRACT

Traditional networks have difficulty in meeting the needs with the developing technology and the continuous increase in the size of the data to be processed. The Software Defined Network (SDN) approach has emerged as an alternative to traditional networks. SDN separates the control and data plane from each other and manages the network over the control plane with the advantages it provides in terms of flexibility and cost. In networks with large data flows, it is useful to use multiple controllers, as managing the network with a single controller can cause interruptions and data loss. Placement of controllers in SDNs using multiple controllers is among the main problems.

This thesis study has been carried out to solve the problem of multiple controller placement in SDNs. By applying the Genetic Algorithm and Dijkstra Algorithm to Colt and ULAKNET data sets within the Topology Zoo database, the best position of the controllers has been determined for both Europe and Turkey. In addition, SDN modeling has been presented for pandemic hospitals that need to be established in Turkey. According to the presented modeling, the cities where pandemic hospitals will be established correspond to the control layer, while all other cities that will benefit from these hospitals correspond to the data layer. Positioning pandemic hospitals at strategic points, considering the cost of establishment, will provide these hospitals with a wider coverage area. In order to obtain maximum benefit from pandemic hospitals, it is recommended to establish hospitals in locations with a high number of patients with coronavirus, but at a minimum distance from all other cities.

Keywords: Software-Defined Network, Multi-Controller Placement, Genetic Algorithm, COVID-19, Pandemic Hospital

İÇİNDEKİLER

YAZILIM TANIMLI AĞLARDA GENETİK ALGORİTMA TABANLI ÇOKLU DENETLEYİCİ YERLEŞTİRME

3. BÖLÜM

UYGULAMA DETAYLARI ve DENEYSEL SONUÇLAR

4. BÖLÜM

SONUÇ, TARTIŞMA ve GELECEK ÇALIŞMA

KISALTMALAR

YTA : Yazılım Tanımlı Ağ

ÇDYP : Çoklu Denetleyici YerleĢtirme Problemi

GA : Genetik Algoritma

DA : Dijkstra Algoritması

CNPA : Clustering-Based Network Partition Algorithm

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

GİRİŞ

Günümüzdeki teknolojik geliŞmeler ve ağ ile yönetilebilen cihazların artması, ağ yönetimini daha karmaŞık hale getirmiŞtir. Yeni cihazların ağa katılması, ağdan çıkarılması ve cihazların birbirleriyle haberleŞmesi hususunda yaŞadığı zorluklar geleneksel ağ mimarisini, Yazılım Tanımlı Ağ (YTA) mimarisine yönlendirmiŞtir. YTA yapısında kontrol ve veri katmanı birbirinden ayrılmıŞtır. Bu sayede kontrol katmanındaki ağı yönetme özelliğine sahip denetleyici ya da denetleyicilerle tüm ağ kontrol altına alınabilmektedir. Kontrol katmanında tek bir denetleyici kullanılabileceği gibi ihtiyaç durumuna göre birden fazla denetleyici de kullanılabilir. YTA'larda tek denetleyici kullanımı yük dengesizliği, aŞırı yüklenme, ağ kopması, yanıt gecikmesi gibi problemlere sebep olabilmektedir. Bu tarz problemlerin çözümü için kontrol katmanında birden fazla denetleyici kullanmak önerilmektedir. YTA'larda birden fazla denetleyici kullanmak ise çözülmesi gereken bazı problemleri beraberinde getirmektedir. Kontrol katmanında kaç denetleyici kullanılacağı, denetleyicilerin birbirleri arasında ileti Şiminin nasıl sağlanacağı, denetleyicilerin tüm ağ yükünü aralarında nasıl paylaŞacağı ve denetleyicilerin nasıl yerleŞtirilmesi gerektiği bu problemler arasında yer almaktadır.

Bu çalıŞmada YTA'larda Çoklu Denetleyici YerleŞtirme Problemi (ÇDYP) çözümü için Genetik Algoritma(GA) ile Dijkstra Algoritması(DA) birlikte uygulanmıŞtır. Türkiye ve Avrupa ağ haritalarındaki Şehirlerin, maksimum Şekilde faydalanabileceği denetleyici yerleŞtirmesi yapılmıŞtır. Bunun yanı sıra koronavirüs ile mücadele için kurulacak olan pandemi hastanelerinin Türkiye'de yerleŞtirilmesi, YTA'larda ÇDYP olarak modellenmiŞtir. Bu modellemeye göre pandemi hastanelerinin kurulması için uygun olan Şehirler kontrol katmanını oluŞtururken bu hastanelerden faydalanacak olan diğer tüm Şehirler veri katmanını oluŞturmaktadır. Deneyler sonucunda ortaya çıkan veriler ile pandemi hastanelerinin kurulması gereken Şehirler belirlenmiŞ ve sonuçlar

yorumlanmıştır. Önerilen modellemeye göre pandemi hastaneleri tüm şehirlere minimum uzaklıkta fakat hastalığın fazla olduğu şehirlere yakın yerlerde kurulmalıdır.

Tezin Amacı

Günümüzde ağa katılan cihaz sayısının sürekli artması ile ağ yönetimi daha da karmaşık hale gelmiştir Geniş kapsamlı ağların yönetilmesi için YTA mimarisinin kullanılması hem literatür çalışmalarında ele alınmış hem de firma uygulamalarında yerini almaya başlamıştır.

Bu tez çalışmasının amacı YTA'ların ÇDYP'ni GA ve DA kullanarak çözmektir. Çalışmada Türkiye ve Avrupa ağ haritalarında kullanılacak denetleyicileri maksimum kapsam sağlayacak şekilde yerleştirmek amaçlanmıştır. Ayrıca Türkiye'de koronavirüs salgını ile mücadelede kurulması gereken pandemi hastanelerini YTA mimarisi ile modellemeyi amaçlar. Kurulması gereken pandemi hastanelerinin olduğu şehirler, diğer tüm şehirlere minimum mesafede ve maksimum koronavirüs vaka sayısına sahip çözüm kümelerinden oluşmaktadır. Pandemi hastanelerinin kurulum maliyeti göz önünde bulundurulduğunda kurulabilecek hastanelerden maksimum fayda sağlanması amaçlanmaktadır. Çalışma Türkiye için gerçekleştirilmiş olup gerekli verilerin temin edilmesiyle tüm ülkeler için kurulması gereken pandemi hastanelerinin konumu konusunda yol gösterici olmayı amaçlamaktadır.

Tezin İçeriği

Bu tez kapsamında yapılan çalışmalar bölümler halinde sunulmuştur

Birinci bölümde çalışmanın temelinde kullanılan YTA mimarisi anlatılmıştır. YTA mimarisinin avantaj ve dezavantajlarından bahsedilmiş olup geleneksel ağlarla karşılaştırılmıştır. YTA mimarisinde tek denetleyici veya çoklu denetleyici kullanımından bahsedilmiş ayrıca YTA'larda ÇDYP ayrıntılı bir şekilde verilmiştir. Pandemi hastaneleri ve COVID-19 salgını ile ilgili bilgi verilmiş ve bölüm sonunda ilgili literatür sunulmuştur

İkinci bölümde tez çalışmasında kullanılan GA ve DA tanıtılmış ve bu algoritmaların çalışma şekli adımları ve özellikleri verilmiştir.

Üçüncü bölümde YTA yapısı ile gerçekleştirilen iki deney ayrıntılı olarak verilmiştir. Birinci deneyde YTA'larda ÇDYP'ne GA ve DA uygulanarak çözüm önerilmiştir. Türkiye ve Avrupa ağ haritalarındaki koordinatlar kullanılarak elde edilen çehir haritalarından, maksimum kapsam için ayrı ayrı üçer adet denetleyici yerleşim yerleri grafiklerle sunulmuştur. İkinci deneyde ise birinci deneydeki koordinat değerlerine ek olarak Türkiye'deki ağ haritası veri dosyasına koronavirüs katsayısı eklenerek Türkiye'de kurulacak pandemi hastaneleri için YTA mimarisi modelli sunulmuştur. Deney sonucunda elde edilen veriler grafiklerle gösterilmiş, Türkiye'de pandemi hastanelerinin kurulması için uygun çehirler tespit edilmiştir.

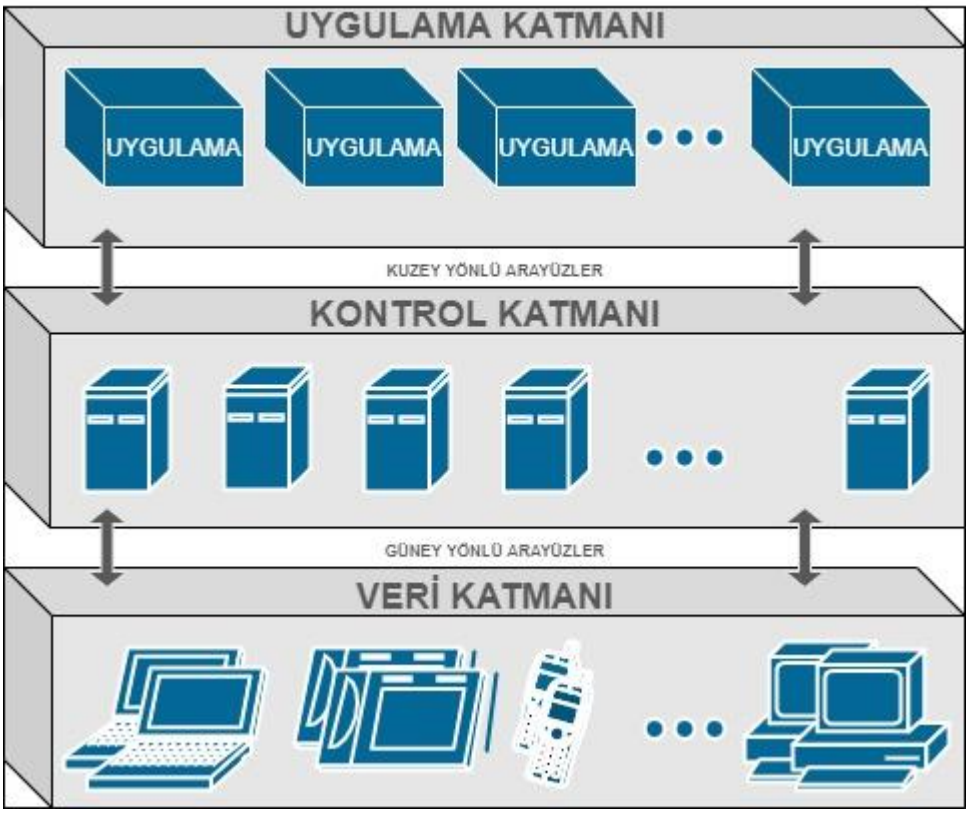
Dördüncü bölümde ise çalışma sonuçları, tartışma ve gelecek çalışmalar değerlendirilmiştir.

1. BÖLÜM

GENEL BİLGİLER ve LİTERATÜR ÇALIŞMASI

1.1. Yazılım Tanımlı Ağ

Yazılım Tanımlı Ağ (YTA), mevcut ağ mimarisinin [1] performansını iyileştirmek için ortaya çıkan yeni bir ağ modellemesidir. Uygulama katmanı, kontrol katmanı ve veri katmanından oluşan YTA mimarisinde katmanlar arası iletişim Kuzey ve Güney yönlü arayüzler aracılığıyla gerçekleştirilir. Temel YTA mimarisi şekil 1'de gösterilmektedir.



Şekil 1. Temel YTA Mimarisi

Şekil 1'de de görüldüğü üzere YTA mimarisinde üç adet katman ve iki adet arayüz bulunmaktadır. Kontrol, kontrol katmanında mantıksal yönetime sahip denetleyici veya denetleyiciler yardımıyla sağlanır. Denetleyiciler; ağ politikalarını uygulama, ağ yapılandırması, topoloji yönetimi, bağlantı bulma, akış tablosu giriş gibi işlemlerden sorumludur. YTA'lar, kontrol düzlemini veri düzleminden ayırdığı için veri düzlemindeki anahtarlar basit yönlendirme cihazları haline gelmektedir. Temel YTA mimarisinde bulunan katmanların ve arayüzlerin özellikleri [1] şu şekildedir:

Veri Katmanı: Veri katmanı YTA özellikli anahtarlardan ve diğer ağ elemanlarından oluşur. Veri katmanı güneye bakan arayüzler sayesinde kontrol katmanındaki denetleyicilere akış iletme bilgisi taşır. Veri katmanı, kontrol katmanının verdiği kararlara göre ağ trafiğini yönlendirir. Veri katmanında bulunan anahtarlar üç bölümden oluşur. Bunlar:

OpenFlow protokolü: Kontrol ve veri katmanı arasında standart bir güney arayüzdür. Denetleyicilerin birçoğu OpenFlow ile uyumludur. Birçok ağ anahtarı satıcısı da OpenFlow protokolü ile uyumlu anahtarlar üretmektedir. OpenFlow standartlarına uygun cihaz üretilmesi, kontrol katmanının daha esnek olmasını sağlamaktadır. Openflow, veri katmanındaki bir anahtarın kontrol katmanındaki birden çok denetleyici ile bağlanmasına olanak sağlar. Anahtar ile denetleyiciler üç durumda bulunabilir. Bunlar eşit, slave (köle) ve master (efendi) şeklindedir. Anahtar ile denetleyicinin eşit olması durumunda denetleyiciler anahtarlar üzerinde tüm erişim iznine sahiptir. Bir anahtar eşit durumdaki denetleyiciye eşzamansız olarak ileti (paket yollama ya da akışı geri çekme bilgisi) gönderebilir. Denetleyiciler eşit durumdaki anahtarlara geçiş durumunu değiştirmek için kontrol komutu göndererek köle ya da efendi olma isteği yollayabilir. Eğer denetleyici köle durumunda ise anahtar üzerinden salt okunur yetkiye sahiptir. Denetleyicinin anahtar üzerinde efendi olması, eşit olması ile aynı yetkilerdedir. Fakat bir anahtar aynı anda birden fazla eşit ya da köle denetleyiciye bağlanabilirken sadece bir efendi denetleyiciye bağlanabilir.

Güvenli kanal: Her bir OpenFlow özellikli anahtarı, denetleyiciye bağlayan arabirimdir.

Akış tablosu: Yönlendirme politikalarından oluşan bir gruptur.

Güney Yönlü Arayüz: Veri katmanı ile kontrol katmanını birbirine bağlayan iletiğim arayüzüdür. OpenFlow en popüler güney yönlü arayüzlerdendir.

Kontrol Katmanı: YTA'larda ağın yönetildiği bu katman bir ya da daha fazla denetleyici cihazdan oluğur Denetleyiciler ağ ilkelerini belirlemek için genel ağ bilgisine sahip olur ve daha sonra anahtarları güvenli bir kanal üzerinden yapılandırabilirler. Denetleyiciler kendi aralarında iletiğimi doğu-batı arayüzleriyle, uygulamalarla iletiğimi kuzey yönlü arayüzlerle ve veri katmanındaki anahtarla iletiğimi güney yönlü arayüzler ile sağlamaktadır.

Kuzey Yönlü Arayüz: Uygulama katmanı ile kontrol katmanını birbirine bağlamaktadır. Amaç uygulamaların ağ kaynaklarını ve bu kaynakların yetkilerini kolayca yönetmesini sağlamaktır. Güneydeki arayüzlerden faklı olarak belli bir standartları yoktur. Çünkü mevcut denetleyiciler ve uygulamaların birleğtirilip bir standartta bağlanması zordur.

Uygulama Katmanı: Kullanıcı ihtiyaçları için tasarlanmığuygulama ve hizmetlerden oluğur. Uygulamalar kontrol katmanı sayesinde ağ elamanlarına eriğilebilir.

1.1.1. Yazılım Tanımlı Ağlar ile Geleneksel Ağların Karşılaştırılması

Bilgi teknolojilerinin sürekli geliğmesiyleinternet, insanların çalığma ve yağam tarzını derinden etkileyen karmağık ve büyük ölçekli bir altyapıya dönüğmüğtür [2]. Ancak geleneksel ağın karmağıklığı, yöneticilerin özellikle dinamik ve karmağık ağ yapılarında çalığmasını ve bu ağları yönetmesini zorlağtırır. Mevcut ağların bu sorunlarını iyileğtirmek için yeni bir teknolojiye ve yönteme ihtiyacı vardır. YTA, bu sorunları çözmek için kontrol ve veri katmanlarını birbirinden ayrılması esasına dayanan, ağın yazılımsal olarak kolayca yönetilmesini sağlayacak ğekilde oluğturulmuğ yeni bir ağ modellemesidir. Tabloda geleneksel ağlar ile YTA'ların bazı yönlerden karğılağtırılması yapılmığtır.

Tablo 1. Geleneksel Ağlar ile YTA'ların Karşılaştırılması

GELENEKSEL AĞ	YAZILIM TANIMLI AĞ
Yönetim Karmaşıklığı: Ağ yapısı katmanlara ayrılmadığı için büyük ölçekli ağ yapılarının yönetimi zordur. Ağ yönetimi manuel olarak yapılmaktadır.	**Yönetim Kolaylığı:** Kontrol ve veri katmanı birbirinden ayrı olduğu için ağ yönetimi kolaydır. Ağ içerisindeki trafik ve ağ içerisinde eklenip çıkarılacak cihazlar kontrol katmanındaki denetleyiciler tarafından rahat bir şekilde yönetilebilir.
Düşük Erişilebilirlik: Ağ içerisinde herhangi bir sorun ile karşılaşıldığında problemin kaynağının bulunması zordur ve problemin çözüm süresi uzundur.	**Yüksek Erişilebilirlik:** Sistemde herhangi bir sorun ile karşılaşıldığında kontrol katmanı hızlı ve etkili bir çözüm sunar.
Aşırı Yüklenme: Ağ içerisinde yoğunluk yaşanması durumunda cihazlar aşırı yüklenebilir. Aşırı yüklenme tıkanıklıklara yol açarak haberleşme sıkıntısına neden olabilir.	**Yük Dengeleme:** Ağ trafiğinde yaşanılan yoğunluk denetleyici katmanı programlanması ile daha az yoğunluğa sahip alanlara paylaştırılabilir. Bu sayede cihazların yük dengelemesi sağlanmış olup haberleşmede sorun yaşanmaz.
Statik Yapılandırma: Ağ içerisindeki yapılandırmalar manuel olarak yapıldığı için zaman kaybına ve yanlışlıklara neden olabilir.	**Dinamik Yapılandırma:** Kontrol katmanı ile tek merkezden standart ve hızlı bir şekilde veri katmanı elemanları eklenebilir, güncellenebilir ya da devre dışı bırakılabilir.

1.1.2. Yazılım Tanımlı Ağlarda Çoklu Denetleyici Yerleştirme Problemi

YTA mimarisinde kontrol katmanında tek bir denetleyici kullanabileceği gibi ihtiyaç durumuna göre birden fazla denetleyici de kullanılabilir. Kontrol katmanında tek bir denetleyici kullanmak ağın tek bir merkez tarafından yönetilmesi ile tutarlılık açısından avantaj sağlar.[3] Fakat büyük ölçekli ağlarda tek denetleyici kullanmak yeterli olmayabilir. Tek bir kontrol noktasının tek bir hata noktası olması, aynı anda birden fazla isteğe yanıt vermek zorunda olabileceği için darboğaz ve yanıt süresinde gecikme gibi dezavantajları vardır. YTA'lardaki bu tür sorunları çözmek için kontrol katmanında çoklu denetleyici yapısı önerilmiştir. Ancak YTA'larda çoklu denetleyici kullanımı da bazı problemleri beraberinde getirmektedir. Bunlar:

- Kontrol katmanında kullanılacak denetleyici sayısının kaç olması gerektiği,
- Kontrol katmanındaki denetleyicilerin maksimum verimle çalışabilmesi için nasıl konumlandırılması gerektiği
- Kontrol katmanındaki denetleyicilerin teknik özellikleri farklı olabileceği için aralarındaki iletişimin nasıl sağlanacağı,
- Kontrol katmanındaki denetleyicilerin etki alanlarının farklı olması nedeniyle ağ durum bilgisinin nasıl ve ne sıklıkta eşitleneceği problemleridir.

YTA'larda Çoklu Denetleyici kullanımının çeşitli avantaj ve dezavantajları bulunmaktadır. Avantajları;

Yönetim: Farklı denetleyiciler farklı özellikleri nedeniyle tek bir etki alanında ya da bölünmüş etki alanlarında daha etkin bir yönetim sağlayabilir.

Ölçeklenebilirlik: Kontrol katmanında tek bir denetleyici kullanılması ağ yoğunluğuna bağlı olarak ağırı yüklemeye sebep olur. Kontrol katmanında çoklu denetleyici kullanımı ağırı yüklenmeyi engelleyerek dar boğaz olayını engeller. Çoklu denetleyici kullanımı artan kaynak kullanımı ile kapasite artırımına yardımcı olur.

Gecikme Azaltma: Tüm ağ işlemleri sıralı olarak tek bir denetleyici tarafından yanıtlanırsa denetleyici ağırı yüklenir ve yanıt gecikmesine sebep olur. Çoklu

denetleyici kullanımında bu trafik farklı denetleyicilere dağıtılabildiği için yanıt gecikmesi azalır.

Sağlamlık: Çoklu denetleyici kullanımı tek bir denetleyici kullanımının tek bir hata noktası olmasını önler. Denetleyicilerden biri ya da birkaçı arızalandığında diğer denetleyiciler devreye girer. Bu sayede ağ olası durumlar için daha sağlam hale gelmiş olur.

Kararlılık: Çoklu Denetleyici kullanımı ile denetleyicilerde yaşanan problemler, diğer denetleyiciler sayesinde kısa sürede çözülebildiğinden ağ kesintiye uğramadan kararlığını korur.

Çoklu denetleyici dezavantajları ise şu şekildedir:

Tutarsızlık: YTA'larda çoklu denetleyici kullanımında ağ durum bilgisinin nasıl eşitleneceği önemli sorunlardan biridir. Denetleyiciler arası tutarsızlık yanıt gecikmelerini artırıp hatalara neden olabilir.

Yerleştirme: YTA mimarisinde kaç denetleyiciye ihtiyaç olduğu ve bu denetleyicilerin en uygun yerleşim yeri önemli sorunlardan biridir.

Zaman Kaybı: Aşırı yüklemiş olan bir denetleyicinin en kısa sürede yük dengelenmesinin yapılması gerekmektedir. Aşırı yüklenmiş olan denetleyicilerin yüklerinin diğer denetleyicilere dağıtılması zaman alabilir ve ağ işleyişinin aksamasına sebep olabilir.

1.2. COVID-19 ve Pandemi Hastaneleri

1.2.1. COVID-19

COVID-19 veya SARS-CoV-2 olarak da adlandırılan koronavirüs, şiddetli akut solunum yolu sendromuna neden olan bulaşıcı bir solunum yolu hastalığıdır [4]. Çin Halk Cumhuriyeti'nde ortaya çıkan ve çeşitli ülkelerden bildirimi yapılan koronavirüs hastalığı tüm dünya için halk sağlığı tehdidi oluşturmaktadır.

Özellikle son aylarda COVID-19 salgınının hayatımızdaki mali ve sosyal etkileri oldukça yıkıcıdır. Bu amaçla, hükümetler COVID-19'un yayılmasını yavaşlatmak için

kan testi kitleri satın alma, maske takma ve sosyal mesafe koyma kuralları, esnek çalışma saatleri, çevrimiçi okullar vb. gibi önlenmeler almaktadır. Bunların dışında hükümetlerin etkili kaynak yönetimi ve koordinasyona ihtiyacı vardır. Özellikle, pandemi hastanelerin nereye yerleştirileceği ve yerleştirilecek pandemi hastane sayısının ne olması gerektiği gibi temel zorluklar hükümetler tarafından etkin bir şekilde ele alınmalıdır.

1.2.2. Pandemi Hastaneleri

Pandemi; dünyada birden fazla ülkede veya kıtada, çok geniş bir alanda yayılan ve etkisini gösteren salgın hastalıklara verilen genel isimdir.

Pandemiye neden olan hastalığı taşıyanların tedavisi için özel olarak kurulan hastaneler ise pandemi hastanesi olarak bilinmektedir. Pandemi Hastaneleri; bünyesinde Enfeksiyon Hastalıkları ve Klinik Mikrobiyoloji, Göğüs hastalıkları, Göç Hastalıkları uzmanı hekimlerden en az ikisinin bulunduğu ve 3. seviye erişkin yoğun bakım yatağı bulunan hastanelerdir[5].

Koronavirüs gibi tüm dünya yüzeyi gibi çok geniş bir alanda yayılan ve tüm insanlığı etkisi altına alan pandemik hastalıklarla mücadelede pandemi hastaneleri hayati önem taşımaktadır.

1.3. Literatür Araştırması

YTA'larda ÇDYP, literatürde yoğun olarak incelenmiştir. Makaleler [6-18], YTA'ların faydalarını ve uygun tek denetleyici veya çoklu denetleyicilerin dağıtımları için denetleyici yerleştirme problemlerini, buldukları çözümleri açıklamaktadır.

[6] çalışmasında yazarlar, ÇDYP'ye bir çözüm olarak GA ile Parçacık Sürü Optimizasyonu Algoritmasını hibrit olarak kullanarak büyük ölçekli bir ağ için yanıt gecikmelerini en aza indiren bir çözüm sunmaktadır. Hibrit algoritma sayesinde en iyi konum, sadece GA kullanımına kıyasla daha kısa sürede elde edilir.

[7] çalışmasında, ağ yükünün dağıtımı için ÇDYP farklı senaryolar uygulayarak dağıtılmış denetleyicilerin yerleştirme problemine çözüm önerilmiştir.

Öğretim gecikmesi, kontrol düzlemi kullanımı ve denetleyici işyükü dağıtımı gibi farklı yönleri olan ÇDYP [8]'de incelenmiştir. GA ile bir hibrit gradyan iniş optimizasyon yöntemi, iki farklı ağ senaryosunda uygulanmış ve kontrol düzleminin kullanımı ile ağ yanıt süresi arasındaki dengeyi etkili bir şekilde kurmuştur

[9] çalışmasında, farklı denetleyici yerleştirme modelleri ve bunların doğruluğu incelenmiştir. Çalışmanın diğer hedefleri, denetleyici kapasitesini, güvenilirliği ve esnekliği en üst düzeye çıkarırken ağ gecikmesini, dağıtım maliyetini ve enerji tüketimini en aza indirmektir. ÇDYP ile ilgili olarak, yazarlar, denetleyici yerleştirme modelleri tarafından kullanılan arama yöntemlerini göstermek için literatürde benimsenen bir özellik seçme tekniğini sunmuştur

[10] çalışmasında, ÇDYP için hiyerarşik bir K-medyan Algoritması sunmuşlardır. Ağ içerisinde yük dengelemeyi düşünerek ağ alanlarını kontrol alanlarına bölerek alanın verimli denetimini sağlamaya çalışırlar.

[11]'de yazarlar, geniş alan ağları için denetleyiciler ve anahtarlar arasındaki paketleri değerlendirerek yayılma gecikmesini azaltmanın önemli olduğunu savunmaktadır. Küme Tabanlı Ağ Bölümü Algoritması (clustering-based network partition algorithm: CNPA) adı verilen yeni bir yaklaşım önermişlerdir. CNPA, denetleyici ve denetleyici arasındaki yayılma gecikmesini kısaltmak için ağı alt ağlara böler ve ilgili anahtarları bir alt ağa yerleştirebilir. Ağ gecikmesini en aza indirmeyi, güvenilirliği ve dayanıklılığı en üst düzeye çıkarmayı, dağıtım maliyetini ve enerji tüketimini en aza indirmeyi amaçlamaktadırlar. ÇDYP için denetleyiciler arasında işbirliğine, çoklu denetleyicilerin kullanımıyla maliyet farkındalığına ve optimizasyona dayalı olacak parametrelerin sayısını artırmaya odaklanmaktadırlar.

[12] 'de yazarlar ÇDYP için çözüm tekniklerini ve sınırlamalarını analiz ederler. Çözüm teknikleri genellikle nesnel işlevlere dayanmaktadır. Optimum çözümlerin elde edilmesinde, anahtarlar ve denetleyiciler arasındaki yayılma gecikmesi gibi çeşitli faktörler ve denetleyicilerin ve anahtarların kapasitesi gibi kısıtlamalar dikkate alırlar.

[13] çalışmasında, YTA mimarisini kullanarak İnternette işlenen suçlarla mücadele etmek için uygun maliyetli gerçek zamanlı bir izleme ve kayıt sistemi sunulmuştur [14]'te, yazarlar küçük ölçekli ve düşükmaliyetli bir YTA tasarımı uygulanmıştır. İki

farklı yük dengeleme algoritmasının (Round Robin ve Dijkstra) performansı, [15]'te ise YTA yük dengeleme problemini ele almak için gidiş-dönüş süreleri kullanılarak incelenmiştir.

[16]'da yazarlar, denetleyici yerleştirme probleminde denetleyicilerin yük faktörünü incelemiştir. Bunun için Kapasiteli ÇDYP tanımlanmıştır. Değerlendirme, yeni stratejinin gerekli denetleyici sayısını önemli ölçüde azaltabileceğini ve maksimum yük kontrol cihazının yükünü azaltabileceğini göstermektedir.

[17] çalışmasında yazarlar, YTA'larda ÇDYP için yeni bir matematiksel model önermektedir. Modelin amacı, farklı kısıtlamaları göz önünde bulundurarak ağın maliyetini en aza indirmektir. Simülasyon sonuçları, modelin küçük ölçekli YTA'ları planlamak için kullanılabileceğini göstermektedir.

[18]'de, ağ operatörlerinin planını tanıtmak ve YTA altyapısını güncellemek için yeni bir genişleme modeli sunulmuştur Belirli bir ağ tasarımı ve anahtarlar için model, yeni ağın topolojisini tasarlamak için kaç tane, nereye ve hangi denetleyicilerin kurulacağını bulur.

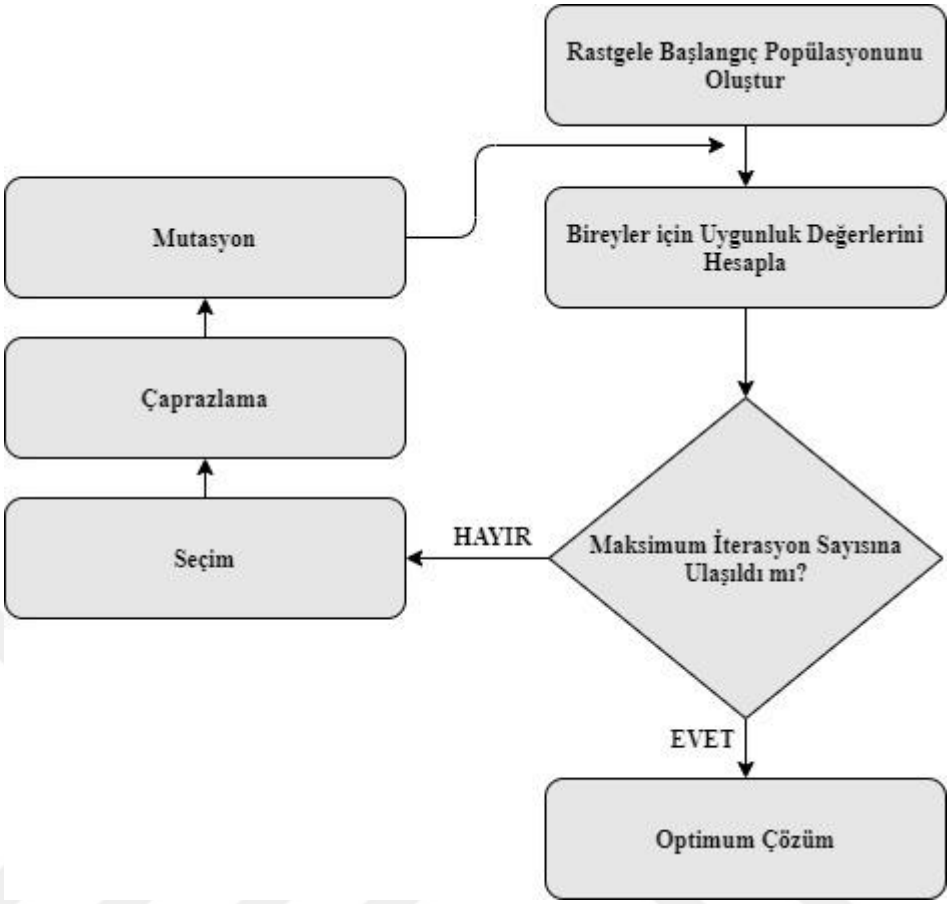
2. BÖLÜM

YÖNTEM VE MATERYALLER

2.1. Algoritmalar

2.1.1. Genetik Algoritma

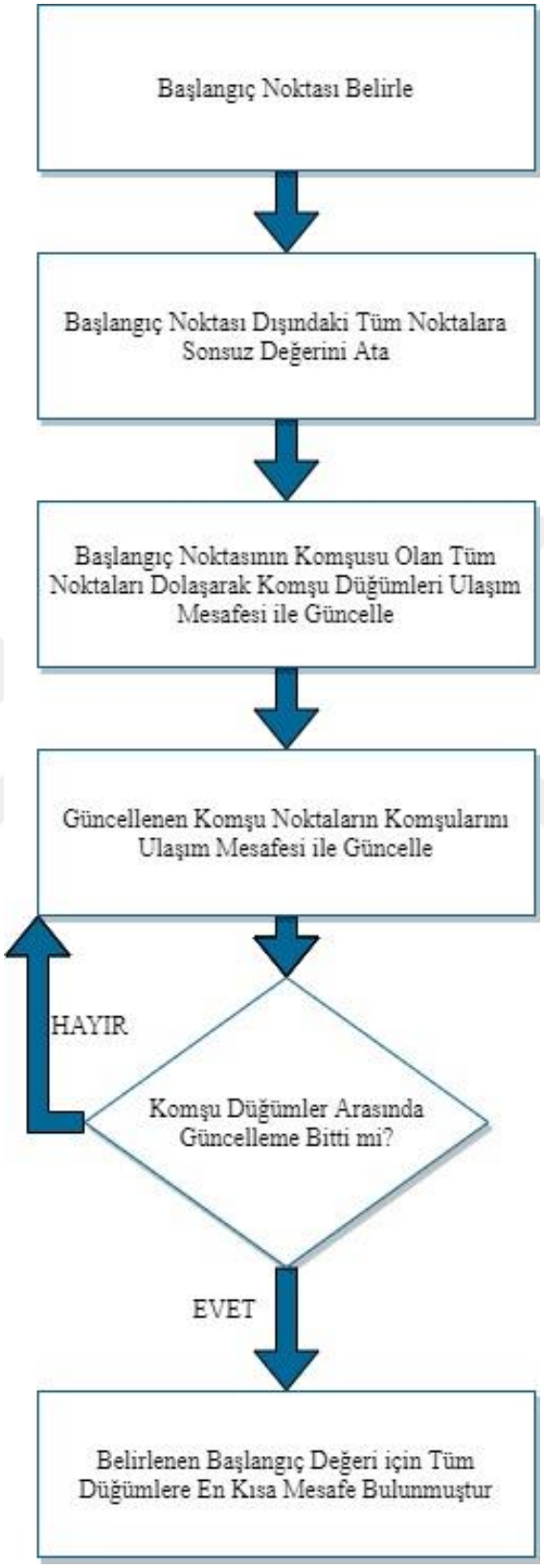
GA, çaprazlama ve mutasyon işlemleriyle yeni bir birey oluşturmak için mevcut nüfustan daha yüksek formda olan ebeveynleri seçen bir popülasyonun doğaya dayalı evrimidir. [19] GA'da temel olarak çaprazlama, mutasyon ve başarılı gen seçimi olmak üzere üç adım bulunmaktadır. GA süreci, rastgele bir popülasyon oluşturulmasıyla başlar. Başlangıç popülasyonunu oluşturduktan sonra her bireyin uygunluk değerini belirlemek gerekir. Uygunluk değeri, sorunun türüne göre çözüme olan yakınlıkla doğru orantılıdır. Uygunluk değeri ne kadar yüksekse, bir bireyin hayatta kalma ve yeniden üreme şansı o kadar yüksektir. Yüksek uygunluk değerine sahip bireyler çaprazlama adımı için seçilir. Çaprazlama adımında, ebeveyn olarak iki birey seçilir. Seçilen bu bireylerden tek noktalı, çok noktalı, tek tip vb. gibi farklı çaprazlama operatörleri kullanılarak bir veya daha fazla birey üretilir. Çaprazlama işleminin ardından gen havuzunu zenginleştirmek üzere mutasyon işlemi uygulanır. Bu adımlar tamamlandıktan sonra yeni bir popülasyon üretilir. Bu işlemler, belirli bir sonlandırma koşulu karşılanana kadar tekrar edilerek en iyi çözüme gitmek amaçlanmaktadır. GA'nın blok diyagramı şekil 2'de verilmiştir.



Şekil 2. GA Blok Diyagram

2.1.2. Dijkstra Algoritması

DA, en kısa yol araması için en sık kullanılan algoritmalardan biridir. DA'nın amacı, bir grafikteki düğümler arasındaki en kısa mesafeyi bulmaktır [20]. Uygulandığı sorunlara çözüm ararken her adımda en uygun çözümü bulmayı hedeflediği için açgözlü arama prensibini kullanır. DA'nın girdisi ağırlıklı grafiklerdir. Çıktısı ise başlangıç düğümünden grafikteki her bir tepe noktasına giden en kısa yoldur. DA genel işleyişi şekil 3'te verilmiştir [21].



ġekil 3. DA Blok Diyagram

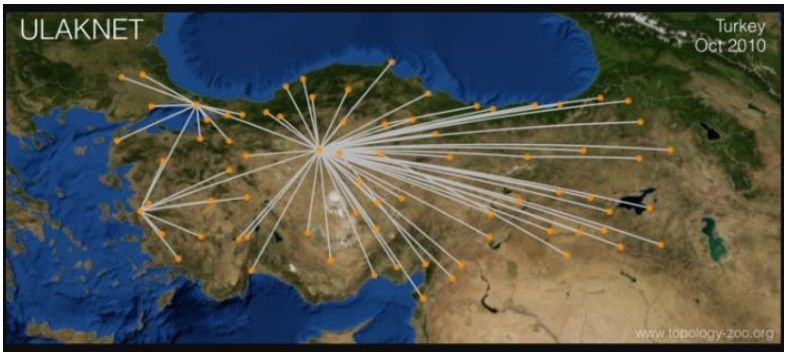
3. BÖLÜM

UYGULAMA DETAYLARI ve DENEYSEL SONUÇLAR

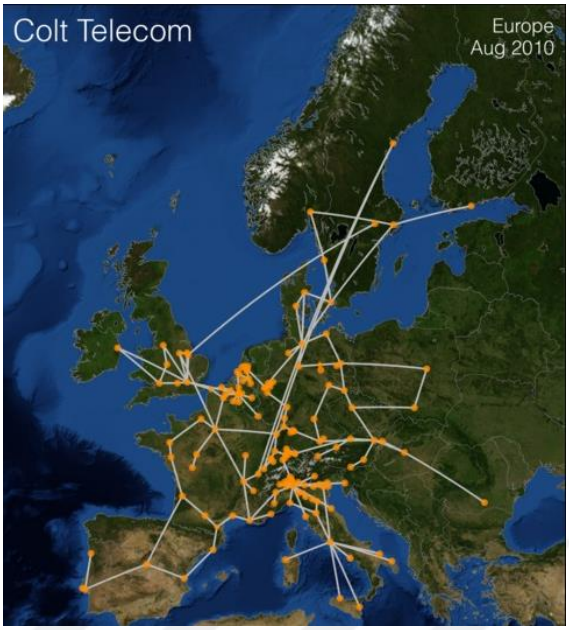
Deneyler, Linux tabanlı Ubuntu 16.04 işletim sistemi, 8 GB RAM ve Core i7, 2.0 GHz işlemciye sahip bir makinede Python kodları ile gerçekleştirilmiştir. Python kodlarını derlemek için Pycharm platformu kullanılmıştır.

3.1. Deney 1: YTA'larda DA tabanlı GA ile ÇDYP

Bu deneyde YTA'larda ÇDYP için DA tabanlı DA yaklaşımı kullanılarak denetleyici yerleşimi sağlanmıştır. Bu deneyde şekil 4 ve şekil 5'te verilen Topology Zoo veri tabanı bünyesinde bulunan Colt ve ULAKNET veri setlerine ait Avrupa ve Türkiye Ağ Haritaları kullanılmıştır.

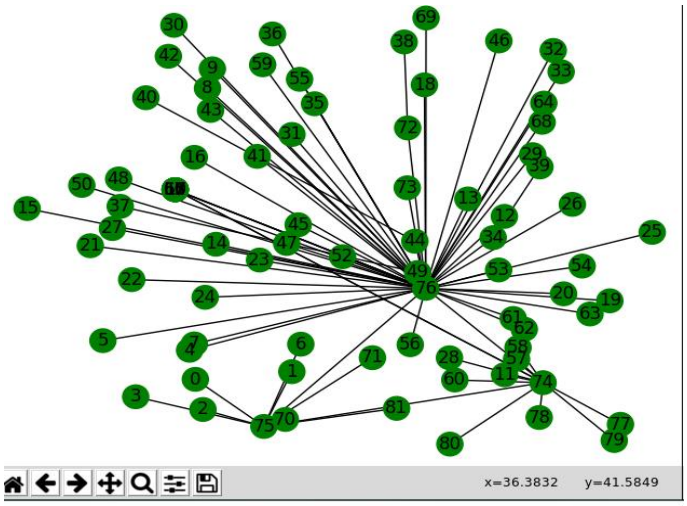


Şekil 4. ULAKNET Türkiye Haritası



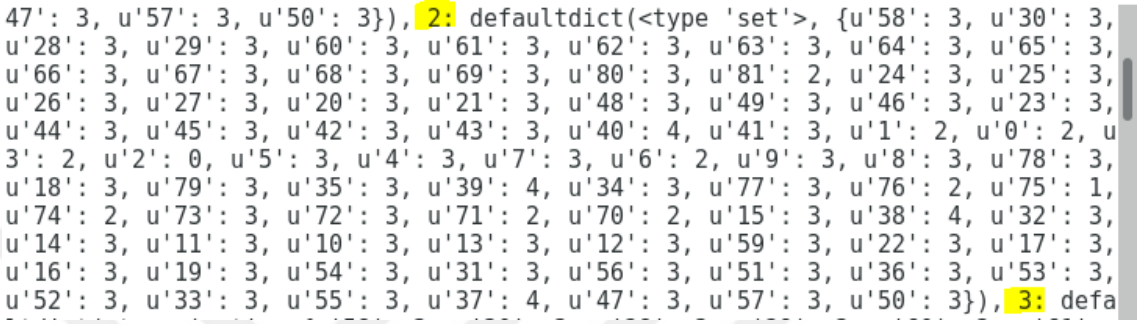
Şekil 5. ULAKNET Avrupa Haritası

Topology Zoo [22] veri tabanı bünyesinde bulunan ULAKNET ve Colt veri setlerindeki enlem ve boylam değerleri kullanılarak Türkiye ve Avrupa için sabit birer düğüm haritası çizilmiştir. Türkiye için çizilen sabit düğüm haritasında şekil 6'da görebileceği gibi her bir şehir bir numara ile temsil edilmektedir. Örneğin İstanbul için 74, Ankara için 76, Kayseri için 45, Erzurum için 18 numara tahsis edilmiştir. Bu aşamada henüz denetleyici yerleri belirlenmemiştir.



Şekil 6. Çoklu Denetleyicilerin Yerleştirilmesinden Önce ULAKNET Türkiye şehir Haritası

şehirlerin koordinatlarına göre Türkiye ağ haritasında DA kullanarak haritalardaki her bir şehrin diğer tüm şehirlere olan en kısa mesafelerinin olduğu bir matris oluşturulmuştur Bu matris şekil 7'de 2 numaralı şehiriçin gösterilmiştir. Bu değerlere göre örneğin 2 numaralı şehrin 34 numaralı şehre uzaklığı 3 birim uzaklıktayken 76 numaralı şehre olan uzaklığı 2 birim uzaklıktadır.



Şekil 7. şehir Mesafe Matrisi

YTA'larda ÇDYP için tercih edilip uygulanan GA parametreleri Tablo 2'de gösterilmiştir. Çalışmada uygulanan modellemeye göre popülasyondaki birey sayısı, ÇDYP çözüm kümesi sayısına karşılık gelmektedir. Bir bireydeki özellik sayısı çözüm kümelerinde kullanılacak denetleyici sayısına karşılıkgelmektedir. Ebeveyn yüzdesi, mutasyon oranı ve Rastgele seçim oranı temel GA adımları için kullanılmaktadır. Jenerasyon sayısı ise GA'yı kaç kere çalıştıracağımız anlamına gelmektedir.

Tablo 2. Birinci Deneyde Kullanılan GA Parametreleri

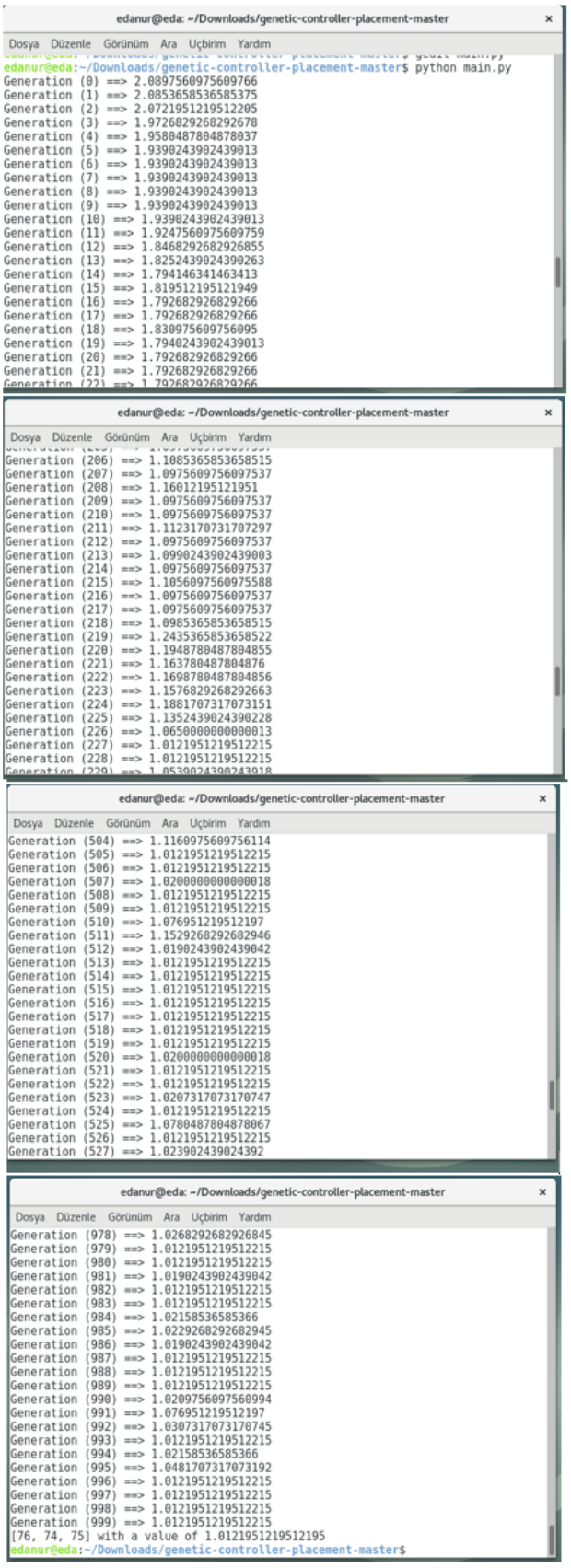
Parametre	Değer
Popülasyondaki Birey Sayısı	100
Bireydeki özellik Sayısı	3
Ebeveyn Yüzdesi	0.1
Mutasyon Oranı	0.05
Rastgele Seçim Oranı	0.01
Jenerasyon Sayısı	1000

Tablo 2'deki GA parametre değerlerine göre başlangıçta rastgele üç noktanın seçildiği 100 adet 1*3'lük çözüm matrisleri oluşturulmuştur Şekil 8'de gösterilen çözüm kümeleri, [0 9] Aralığında bulunan sayılardan rastgele seçilerek oluşturulmuştur.

Şekil 8. Rastgele Çözüm Kümesi

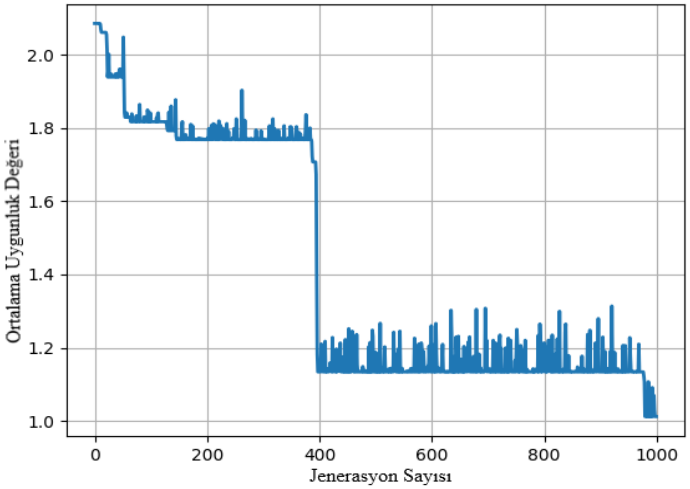
Rasgele çözüm kümelerinin oluşturulmasının ardından bu çözüm kümelerine GA uygulanacaktır. GA'nın çalışabilmesi için bireylerin uygunluk değerlerine ihtiyaç duyulur. Uygunluk değerleri her jenerasyonun sonunda uygunluk fonksiyonu ile hesaplanır. Uygunluk fonksiyonunun değeri bir şehrin diğer şehirlere olan uzaklıkları ile ters orantılıdır. Bir çözüm kümesindeki şehirlerin diğer şehirlere uzaklıkları ne kadar düşükse uygunluk değeri o kadar yüksektir. Her bir jenerasyonda bir çözüm kümesi için ortalama bir uygunluk fonksiyonu değeri belirlenir. Bir jenerasyonda bulunan çözüm, yerel optimum noktası ve bulunan ortalama değer ise yerel optimum değeridir. Tüm jenerasyonlar tamamlandıktan sonra yerel optimum noktası ve yerel optimum değerleri arasından en optimum değer seçilerek global optimum olarak atanır.

Çalışma 1000 jenerasyon çalıştırılmıştır. ULAKNET veri tabanında gerçekleştirilen deneyin bazı jenerasyon sonuçları şekil 9'da gösterilmiştir. Şekil 9'a göre örneğin 17. iterasyonda 1,7926 yerel optimum değeri elde edilmişken 228. Jenerasyonda 1,0121 yerel optimum değeri elde edilmiştir. Bu jenerasyonlarda elde edilen bu ortalama değerler, o iterasyondaki elde edilen optimum çözüm kümesindeki denetleyici şehirlere bağlıdır. Çalışma 1000 jenerasyon çalıştırıldığında tüm jenerasyonlardan elde edilen yerel optimum değerlerinden minimumu global optimum değerine karşılık gelir.

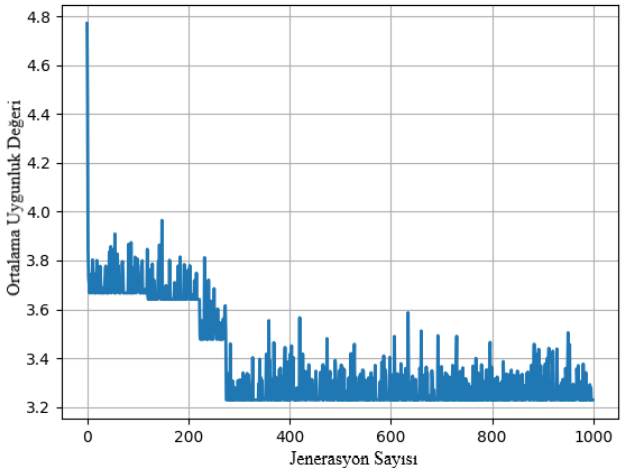


Şekil 9. ULAKNET Jenerasyon ÇalıĢma Örnekleri

Her bir jenerasyon sonucunda sonucu global optimum değere götüren yerel optimum değerlerinden elde edilen yakınsama grafiği ULAKNET veri tabanında şekil 10'da, Colt veri tabanında şekil 11'de gösterilmiştir.



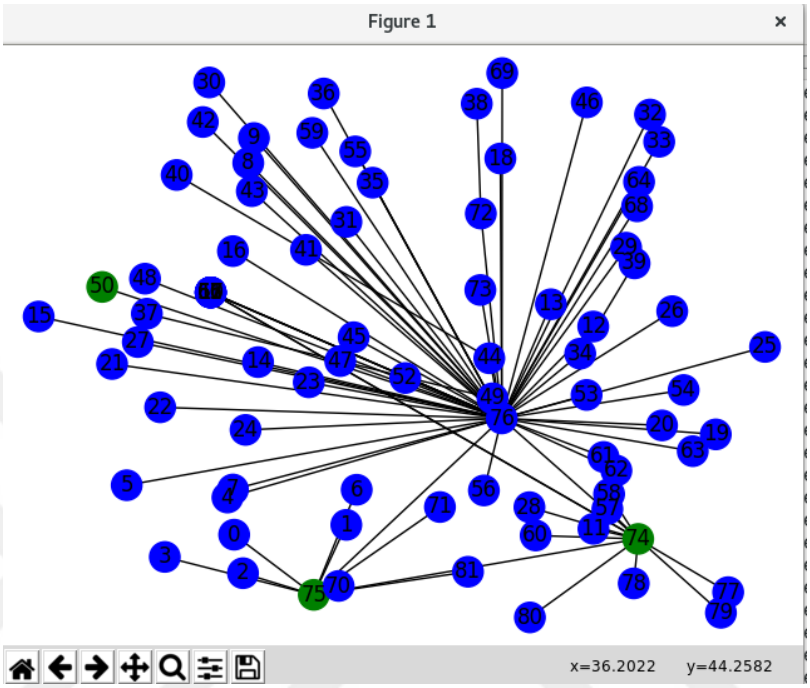
Şekil 10. ULAKNET Veri Tabanı için Ortalama Fitness Fonksiyon Değeri Yakınsama Grafiği



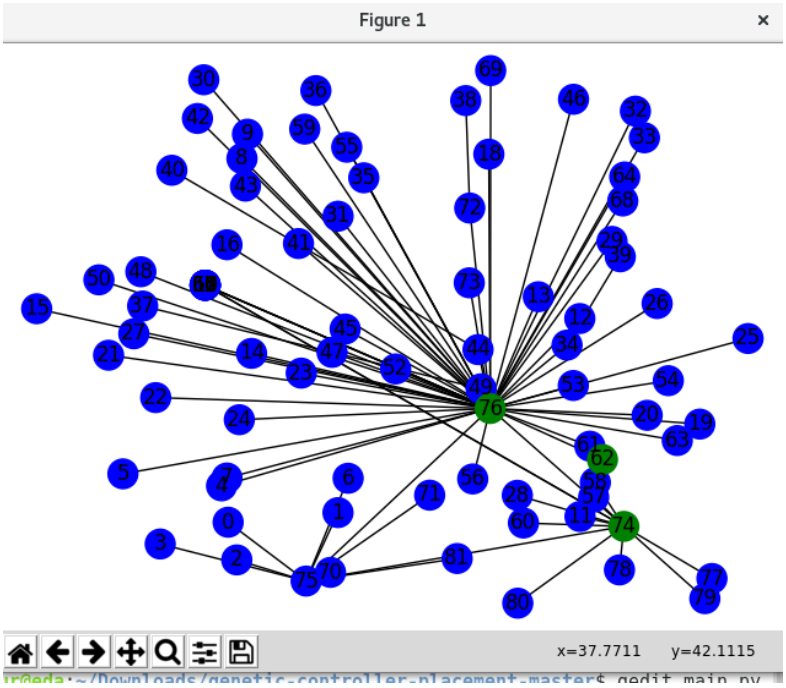
Şekil 11. Colt Veri Tabanı için Ortalama Fitness Fonksiyon Değeri Yakınsama Grafiği

ULAKNET veritabanına GA uygulanarak farklı zamanda gerçekleştirilen 4 örnek deney sonucu ile elde edilen üç denetleyicinin yerleşim şehri şekil 12, 13, 14 ve 15'te gösterilmiştir. ULAKNET veritabanı ile Türkiye ağ haritasına algoritmalarımız uygulandığında en iyi yerleşim yerleri şekil 12'de 50, 74 ve 75 numaralara karşılık gelen Kilis, İstanbul ve İzmir şehirleri iken şekil 13'de 62, 74 ve 76 numaralara karşılık

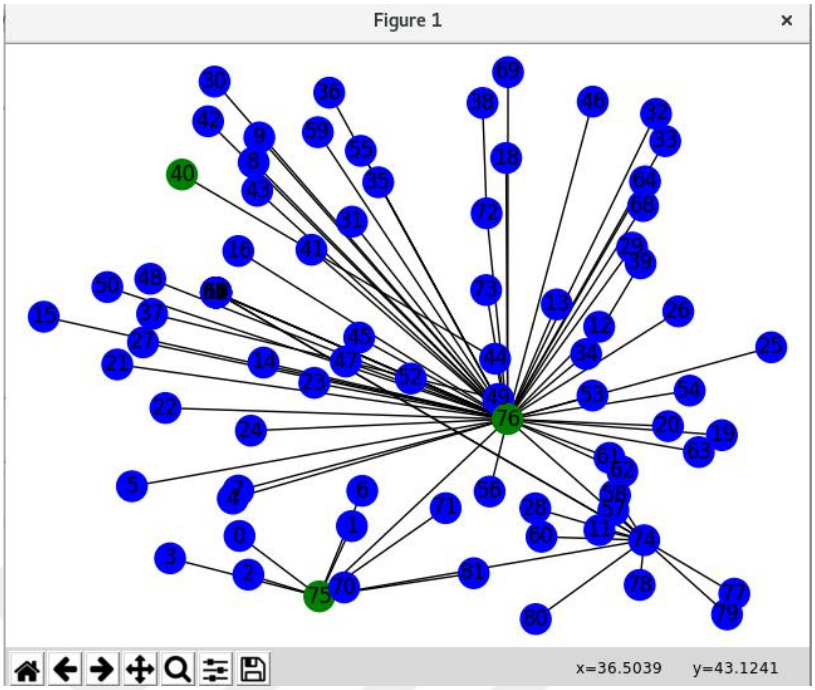
gelen Düzce, İstanbul ve Ankara şehirleridir. Şekil 14'te bulunan en iyi yerleşim yerleri 40, 75 ve 76 numaralara karşılık gelen Mardin, İzmir ve Ankara şehirleri iken şekil 15'te 71, 74 ve 76 numaralara karşılık gelen Kütahya, İstanbul ve Ankara şehirleridir.



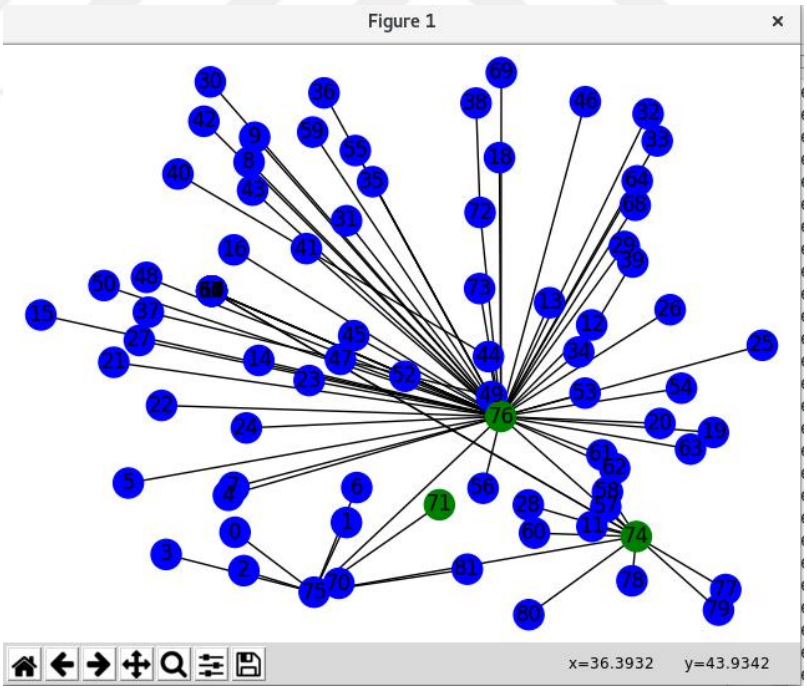
Şekil 12. ULAKNET Denetleyici Yerleşim Örneği-1



Şekil 13. ULAKNET Denetleyici Yerleşim Örneği-2



Şekil 14. ULAKNET Denetleyici YerleĢimÖrneği-3



Şekil 15. ULAKNET Denetleyici YerleĢimÖrneği-4

1000 jenerasyon sonucu elde edilen global optimum çözümler ve buna bağlı olarak elde edilen değerler ULAKNET veri tabanı için Şekil 16'da, Colt veri tabanı için Şekil 17'de gösterilmiĢtir.

```
Generation (980) ==> 1.0121951219512215
Generation (981) ==> 1.0121951219512215
Generation (982) ==> 1.0481707317073192
Generation (983) ==> 1.0121951219512215
Generation (984) ==> 1.0121951219512215
Generation (985) ==> 1.1073170731707338
Generation (986) ==> 1.0175609756097581
Generation (987) ==> 1.0121951219512215
Generation (988) ==> 1.0121951219512215
Generation (989) ==> 1.024146341463416
Generation (990) ==> 1.0121951219512215
Generation (991) ==> 1.0735365853658556
Generation (992) ==> 1.0913414634146361
Generation (993) ==> 1.0121951219512215
Generation (994) ==> 1.021951219512197
Generation (995) ==> 1.0697560975609774
Generation (996) ==> 1.0296341463414647
Generation (997) ==> 1.0121951219512215
Generation (998) ==> 1.0121951219512215
Generation (999) ==> 1.0121951219512215
[74, 75, 76] with a value of 1.0121951219512195
```

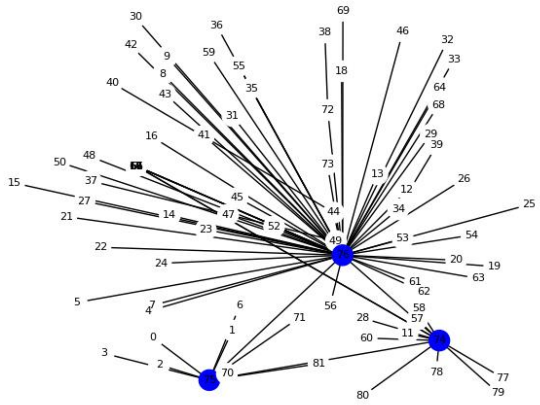
Şekil 16. ULAKNET Veri Tabanı Jenerasyonlar sonucu çözüm kümesi

```
Generation (980) ==> 3.376405228758169
Generation (981) ==> 3.22875816993464
Generation (982) ==> 3.32797385620915
Generation (983) ==> 3.320915032679738
Generation (984) ==> 3.22875816993464
Generation (985) ==> 3.341176470588234
Generation (986) ==> 3.22875816993464
Generation (987) ==> 3.22875816993464
Generation (988) ==> 3.22875816993464
Generation (989) ==> 3.22875816993464
Generation (990) ==> 3.22875816993464
Generation (991) ==> 3.22875816993464
Generation (992) ==> 3.22875816993464
Generation (993) ==> 3.2918954248366017
Generation (994) ==> 3.22875816993464
Generation (995) ==> 3.278758169934639
Generation (996) ==> 3.22875816993464
Generation (997) ==> 3.22875816993464
Generation (998) ==> 3.22875816993464
Generation (999) ==> 3.22875816993464
[51, 30, 152] with a value of 3.2287581699346406
```

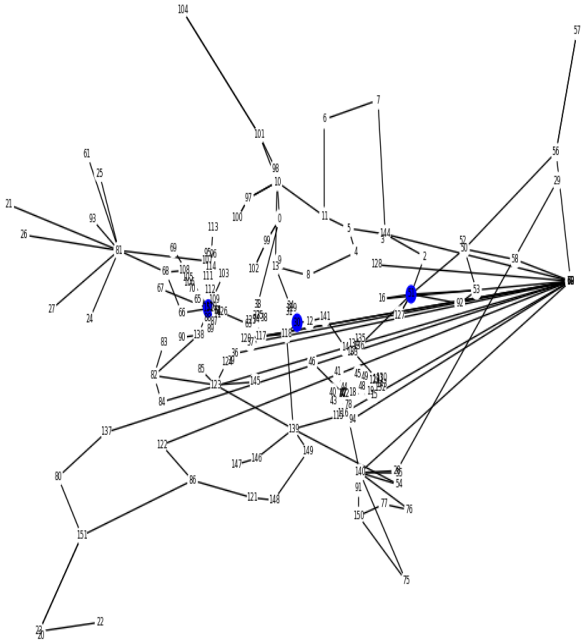
Şekil 17. Colt Veri Tabanı Jenerasyonlar sonucu çözüm kümesi

Şekil 16'da 1000 jenerasyon sonucunda ULAKNET veritabanı için en optimum değer olan 1.0122 değerini veren 74,75 ve 76. Şehirlerinden oluşan çözüm kümesi ile elde edilmiştir. 76 Ankara, 74 İstanbul ve 75 ise İzmir'i temsil etmektedir. Colt veritabanı için ise şekil 17'de en optimum değer olan 3.2288 değerini veren 51,30 ve 152. şehirlerden oluşan çözüm kümesidir. 51 Almanya'daki Hamburg Şehrini, 30 Almanya'daki Karlsruhe Şehrini ve 152 İtalya'daki Milan Şehrini temsil etmektedir.

Deney sonucunda haritada DA ve GA kullanılarak elde edilen denetleyici yerleşim yerleri ULAKNET veri tabanı için şekil 18'de, Colt veri tabanı için şekil 19'da gösterilmiştir.



Şekil 18. ULAKNET Veri Tabanı için Optimize EdilmiĢ Denetleyici YerleĢim Yerleri

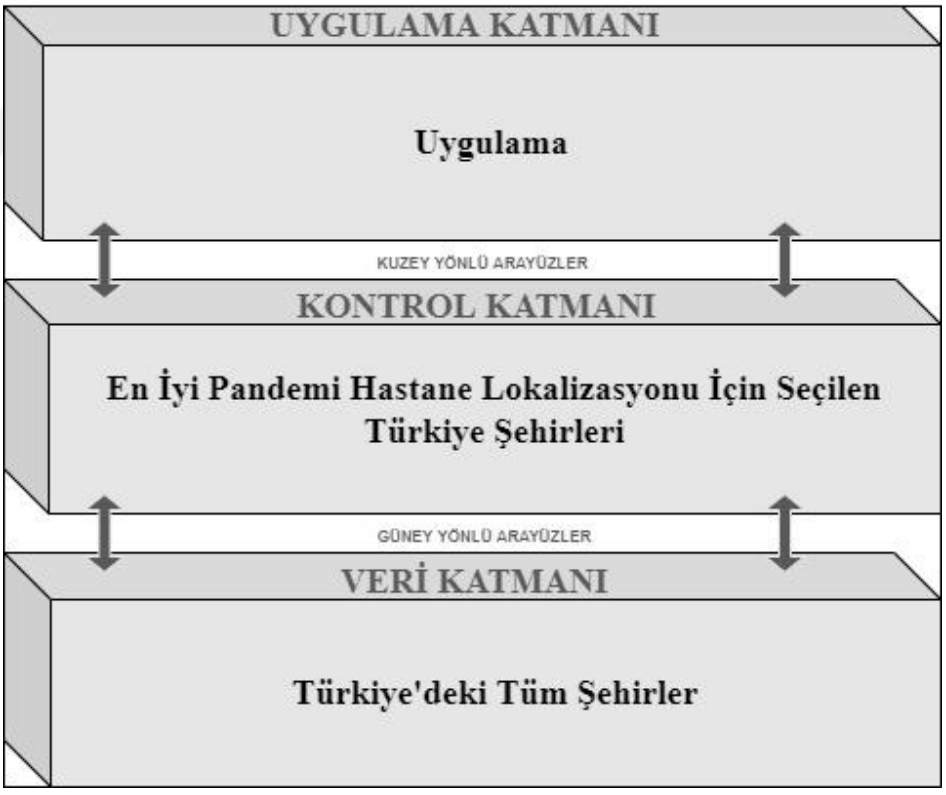


Şekil 19. Colt Veri Tabanı için Optimize EdilmiĢ Denetleyici YerleĢim Yerleri

Bu çalıĢmada YTA'larda ÇDYP, DA temelli GA ile çözülmüĢtür DA'nın her bir çalıĢmada en yakın mesafeyi bulma özelliğinden ve GA'nın her bir çalıĢma zamanında en iyi çözümü bulma özelliğinden yararlanılarak bir sistem tasarlanmıĢ, bu sistem iki farklı veri setine uygulanmıĢ ve elde edilen optimum çözümler grafla oluĢturulan harita üzerinde gösterilmiĢtir.

3.2. Deney 2: Koronavirüsle Mücadelede En İyi Pandemi Hastanesi Lokalizasyonu için YTA Yaklaşımı

Bu deneyde, tüm dünyayı etkisi altına alan koronavirüs salgınıyla mücadelede gerekli olan pandemi hastanelerinin Türkiye'nin hangi şehirlerine kurulması gerektiği ile ilgili çözüm önerilmiştir. Hastane yerleştirme sorunu için GA ve DA, YTA modeline birlikte uygulanmıştır. Pandemi hastanesi yerleştirme probleminin YTA altyapısında nasıl modellendiği şekil 20'de gösterilmiştir. Bu modellemeye göre kontrol katmanı Türkiye'de pandemi hastanesinin kurulması için seçilen şehirlerden oluşmaktadır. Veri katmanında ise Türkiye'deki tüm şehirlerden oluşmaktadır. Kontrol katmanında seçilen şehirler veri katmanında bulunan şehirler için denetleyici rolü oynamaktadır. Amaç kontrol katmanında seçilen şehirlerdeki pandemi hastanelerin diğer tüm şehirlerin pandemi hastanesi ihtiyacını maksimum oranda karşılamasıdır.

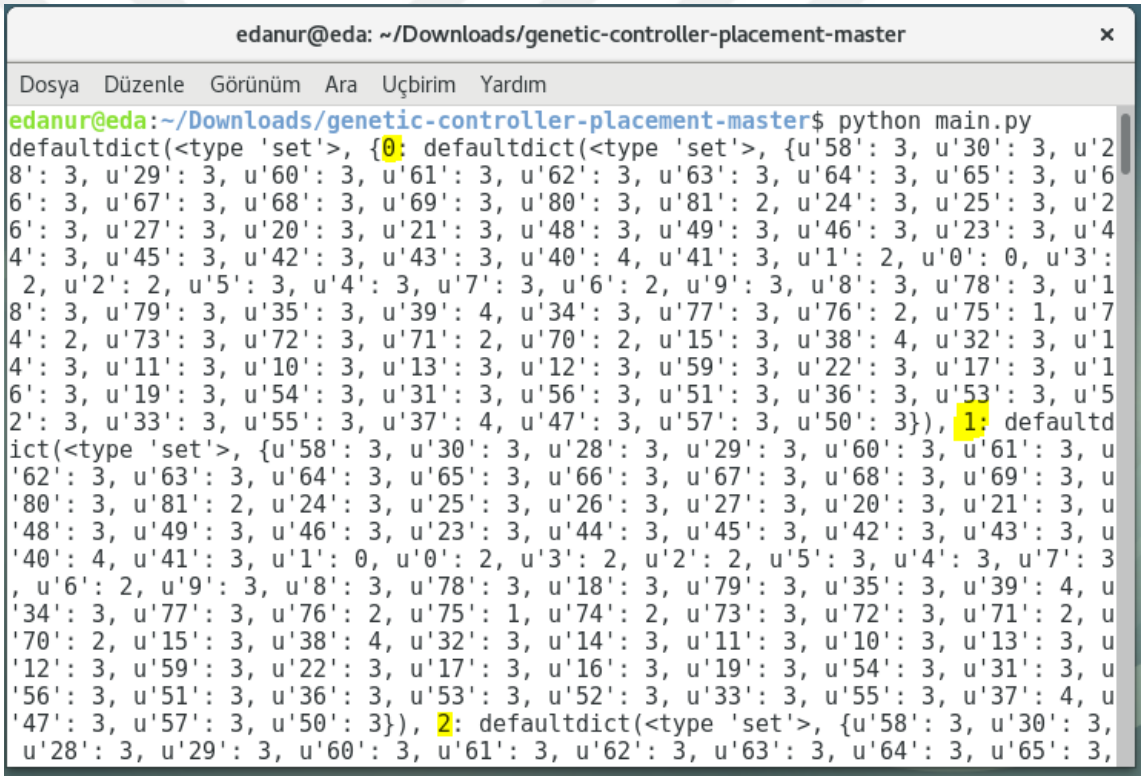


Şekil 20. En İyi Pandemi Hastane Lokalizasyonu için Kullanılan YTA Yapısı

ULAKNET veri setindeki enlem ve boylam değerleri kullanılarak elde edilen Türkiye için sabit bir düğüm haritası şekil 6'da gösterilmiştir. Bu ağ haritasında her bir şehir bir

numara ile temsil edilmektedir. Bu aşamada henüz denetleyici yerleri belirlenmemiştir. Örneğin İstanbul için 74, Ankara için 76, Kayseri için 45, Erzurum için 18 numara tahsis edilmiştir.

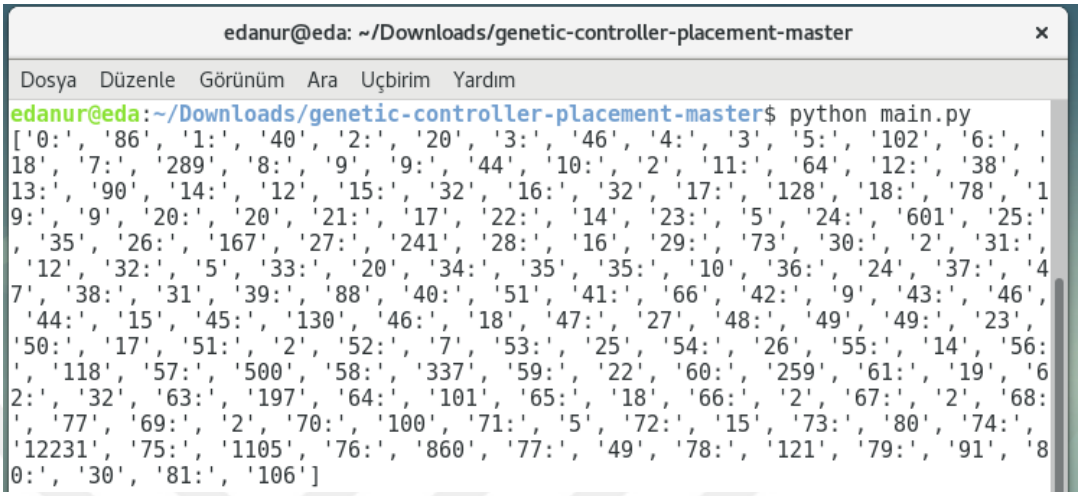
Türkiye'deki şehirlerin koordinat değerlerine göre her bir şehrin diğer şehirlere olan minimum uzaklıklar DA ile hesaplanır. Şekil 21'de her bir şehir için diğer tüm şehirlere olan minimum uzaklıklarından oluşan matris gösterilmiştir. Şekil 21'deki sarı ile belirtilen 1. şehir olan Uşak için 25. şehir olan Sinop şehrine minimum uzaklığı 3 birim iken 40 numaralı şehir olan Mardin'e minimum uzaklığı 4 birimdir.



Şekil 21. şehir Mesafe Matrisi

Bir şehirdeki belli bir tarihteki koronavirüslü hasta sayısı o şehrin koronavirüs katsayısı olarak kabul edilir. ULAKNET veritabanı dosyasına enlem ve boylam değerlerine ek olarak 03/04/2020 tarihinde Türkiye'deki şehirlerin koronavirüs katsayı değerleri [23] veri dosyasındaki şehir özelliklerine eklenmiştir. Koronavirüs katsayı değerleri şekil 22'de gösterilmiştir. Örneğin 74 ili İstanbul'a karşılık gelirken 76 Ankara'ya

karşılık gelir ve İstanbul ve Ankara'nın koronavirüs katsayıları 03/04/2020 tarihinde sırasıyla 12231 ve 860'tır.



Şekil 22. Türkiye şehirlerinin Koronavirüs Matrisi

Veri dosyamızda İstanbul'a ait enlem, boylam ve koronavirüs katsayı verileri aşağıdaki gibidir:

<node id="74">
<data key="d29">1</data>
<data key="d30">41.01384</data>
<data key="d31">Turkey</data>
<data key="d32">Red Colour</data>
<data key="d33">74</data>
<data key="d34">28.94966</data>
<data key="d35">Istanbul</data>
<uuid name="uuid">12231</uuid>
<sehir_no name="sehir_no">74</sehir_no>
</node>

Türkiye'deki tüm şehirler için veriler aynı şekilde enlem, boylam ve koronavirüs katsayısı ile doldurulmuştur

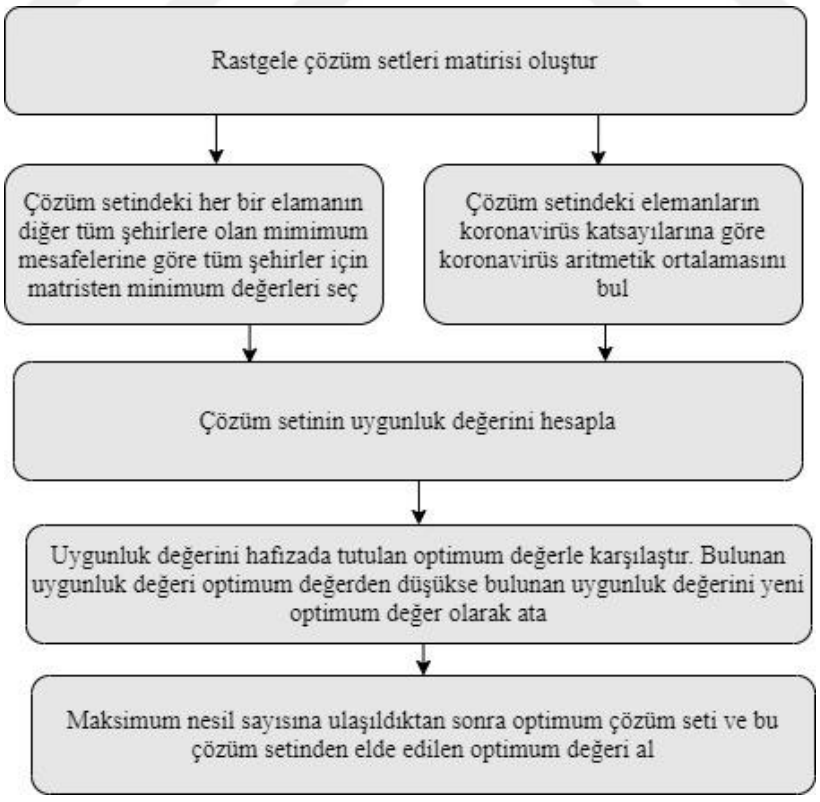
Bu çalışmada ULAKNET veri setine uygulanan GA parametre değerleri Tablo 3'te listelenmiştir. Çalışmada uygulanan modellemeye göre popülasyonundaki birey sayısı,

ÇDYP'de çözüm kümesi sayısına karşılık gelmektedir. Bir bireydeki özellik sayısı çözüm kümelerinde kullanılacak denetleyici sayısına karşılık gelmektedir. Ebeveyn yüzdesi, mutasyon oranı ve rastgele seçim oranı temel GA adımları için kullanılmaktadır. Jenerasyon sayısı ise GA'yı kaç kere çalıştıracağımız anlamına gelmektedir.

Tablo 3. İkinci Deneyde Kullanılan GA Parametreleri

Parametre	Değer
Popülasyondaki Birey Sayısı	100
Bireydeki özellik Sayısı	10
Ebeveyn Yüzdesi	0.1
Mutasyon Oranı	0.05
Rastgele Seçim Oranı	0.01
Jenerasyon Sayısı	1000

YTA'larda ÇDYP'ye göre modellediğimiz hastane yerleştirme problemi için şekil 23'te akış şeması sunulmuştur.



Şekil 23. Hastane Yerleştirme Problemi için Akış şeması

Şekil 23'teki akış Şeması'nın ilk adımı olan rasgele çözüm kümesi Tablo 3'te listelenen parametreler kullanılarak 100 adet rastgele 1x10'luk çözüm kümesi oluşturulmuştur. Rastgele üretilen çözüm kümeleri şekil 24'te gösterilmektedir.

Şekil 24. Rastgele Başlangıç Çözümleri

Her bir rastgele çözüm kümesinin uygunluğu, uygunluk fonksiyonu kullanılarak hesaplanır. Uygunluk fonksiyonunun iki amacı vardır:

- Haritadaki tüm Şehirleri kapsayacak Şekilde minimum mesafelerde çözüm kümeleri seçmek
- Koronavirüs katsayıları daha yüksek olan çözüm setlerini seçmek

Tüm Şehirlerin uygunluk değerlerindeki mesafe için DA'ya göre hesaplanan mesafe matrisi kullanılır. Rastgele çözüm setindeki elemanların içinden diğer tüm Şehirlere minimum uzaklığı olan Şehir seçilir. Çözüm setindeki minimum değerlere göre tüm Şehirlere optimum mesafe hesaplandıktan sonra, çözüm setinden tüm Şehirlere olan minimum mesafenin ortalaması hesaplanır. Örneğin rastgele çözüm setlerinden biri olan [8 6 2 3 4 5 0 7 1 9] için matristeki tüm Şehirlere minimum mesafede olan Şehrin 5 numaralı Şehir olduğunu varsayalım. 5 numaralı Şehrin diğer tüm Şehirlere olan minimum uzakları ortalaması alınır. Elde edilen değer bu çözüm kümesinin minimum uzaklığı değeri olarak elde tutulur. Bu mesafeler tüm çözüm setleri için hesaplanır.

Deneyde kullanılan algoritmada kullanılan parametreler ve formüller şu şekildedir

n denetleyici sayısı; kurulacak pandemi hastanesi sayısına karşılık gelmektedir ve bu değer 10 olarak belirlenmiştir.

i = 1, 2, 3,…, 10 olacak şekildedir ve x_i çözüm kümesidir.

x_j, çözüm kümesindeki elemanların koronavirüs katsayılardır.

$\bar{x}$, çözüm setindeki elemanların koronavirüs katsayılarının aritmetik ortalamasıdır.

h şehirlerin ağırlıklı grafiğidir.

f (x, h), DA fonksiyonudur.

A, seçilen çözüm kümesinin diğer tüm şehirlere en kısa mesafelerle hesaplanan değeridir.

L_Opt ve G_Opt, sırasıyla lokal optimum ve global optimum için değerleridir. Denklem (1) çözüm setinde yer alan elamanların koronavirüs katsayılarının aritmetik ortalamasını gösterir ve (2) çözüm kümesindeki elemanlardan diğer şehirlere minimum mesafede olan şehrin tüm şehirlere olan uzaklarının ortalamasını göstermektedir. (1) ve (2). Fonksiyonları 100 çözüm kümesi için hesaplama yapar. (1) ve (2) hesaplandıktan sonra, (2) nolu formülden elde edilen değer (1) nolu değerden elde edilen değere bölünerek seçilen çözüm setinin yerel optimum değeri elde edilir (3). (3) nolu formüle göre lokal optimum değerinin uygun olabilmesi için A değerinin yani diğer şehirlere olan mesafenin düşük fakat $\bar{x}$ değerinin yani şehirlerin koronavirüs katsayı ortalamasının yüksek olması gerekmektedir.

$$\bar{x} = \frac{\sum_{j=1}^{n} x_j}{n} \quad (1)$$

$$A = ortalama(\min(f(x_i, h), i = 1, \dots, n)) \quad (2)$$

$$\text{L\_Opt} = \frac{A}{\bar{x}} \quad (3)$$

Başlangıçta, global bir optimum değer varsayılan olarak yüksek bir değer olarak (4) ile tanımlanır.

$$G\_Opt = Maxint \quad (4)$$

Her yinelemede, global optimum değer, (3) 'teki değer ile karşılaştırılır. (3) 'teki değer, global optimum değerden daha iyiyse, yeni global optimum değer bu yeni değer (5) olacaktır.

"L_Opt" değeri "G_Opt" dan düşükse, o zaman:

$$G_{Opt} = L\_Opt \quad (5)$$

Deneyin amacı, ortalama mesafenin minimum ve koronavirüs aritmetik ortalaması maksimum olduğunda optimum değere ulaşmaktır. Böylelikle pandemi hastanelerin kurulmasında iki kriter dikkate alınmaktadır:

• Haritadaki şehirlere yakın olan ve maksimum şehir sayısını kapsayacak çözüm kümeleri,

• Koronavirüs sayısı yüksek şehirlerden oluşan ve hastanelere daha çok ihtiyaç duyan çözüm kümeleridir.

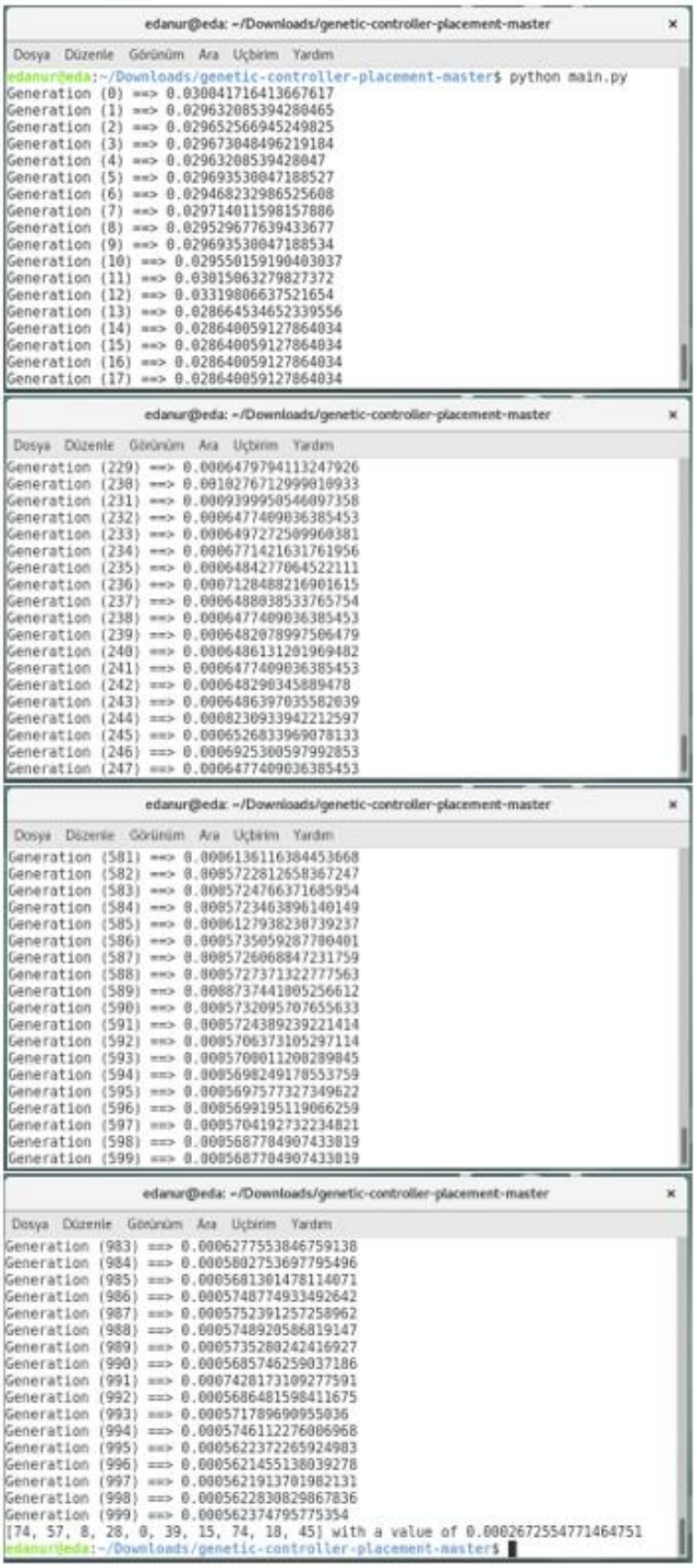
Her deney 1000 iterasyon için çalıştırılır. Maksimum iterasyona ulaşıldıktan sonra optimum çözüm seti ve optimum değer baz alınır. Çalışmada 30 kez yürütülmüştür 30 çalışmadan elde edilen en iyi uygunluk değerleri Tablo 3'te gösterilmiştir. İlk kolon çözümün kaçıncı çalıştırmadan elde edildiğini, ikinci kolon çalıştırmadan elde edilen çözüm setini yani pandemi hastanesi kurulacak şehirleri temsil eden numaraları, üçüncü kolon ise o çözüm kümesinin uygunluk değerini göstermektedir. Tablo uygunluk değeri en yüksek olandan en düşükolan çalıştırmalara göre sıralanmıştır.

Tablo 4. Deney Sonuçları

Çalıştırma No	**Pandemi Hastanesi Kurulacak Şehir Numaraları**	**Uygunluk Değeri**
9	[74, 57, 8, 28, 0, 39, 15, 74, 18, 45]	0.0002672554771464751
12	[79, 24, 37, 76, 40, 1, 74, 75, 58, 39]	0.0003217251423851137
1	[76, 74, 57, 58, 74, 38, 28, 39, 75, 40]	0.0003219547001035476
14	[60, 42, 39, 49, 12, 40, 74, 24, 76, 33]	0.00032287621623980107

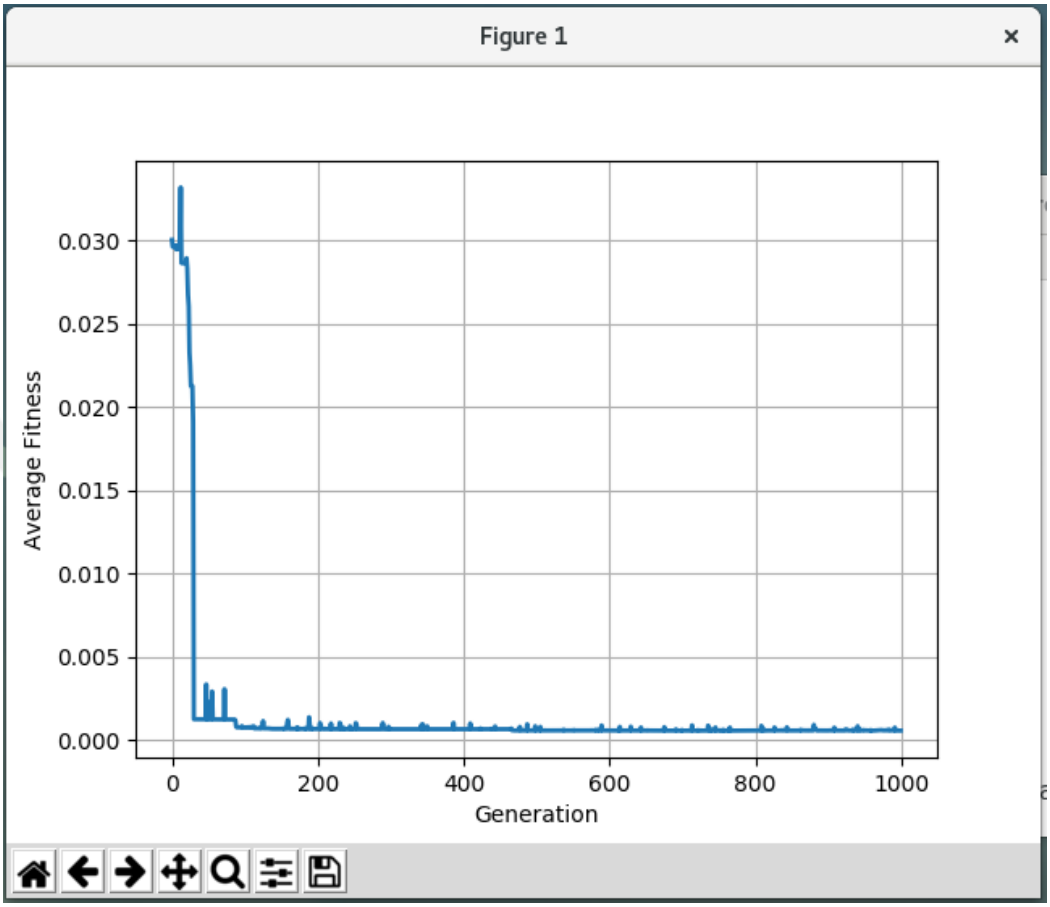
21	[65, 10, 24, 76, 74, 75, 13, 44, 63, 58]	0.0003229917768039527
11	[28, 57, 24, 43, 75, 45, 38, 19, 76, 74]	0.0003231074201182398 3
5	[51, 74, 52, 62, 53, 57, 58, 38, 7, 76]	0.0003235352480868282
8	[74, 24, 63, 57, 40, 3, 42, 76, 79, 59]	0.0003235708226569537
28	[74, 40, 58, 24, 76, 7, 57, 54, 75, 34]	0.0003238789470047675
17	[24, 58, 18, 75, 76, 37, 40, 7, 74, 39]	0.0003241519484160358 7
18	[18, 75, 7, 57, 58, 37, 44, 74, 24, 74]	0.0003242233769377750 5
20	[76, 74, 60, 74, 14, 2, 22, 40, 57, 78]	0.0003242233769377750 5
10	[44, 18, 76, 34, 58, 5, 74, 74, 30, 34]	0.0003243383497664764
22	[76, 24, 74, 74, 27, 37, 38, 58, 75, 57]	0.0003245685402205335
13	[75, 60, 74, 39, 30, 40, 57, 27, 24, 74]	0.0003247990576496674
29	[18, 40, 67, 63, 24, 81, 64, 25, 74, 76]	0.0003249144391976779 4
19	[64, 38, 24, 75, 76, 74, 62, 16, 39, 50]	0.0003249690401117191
15	[74, 24, 40, 49, 16, 59, 11, 2, 76, 7]	0.0003250861038869755
7	[75, 76, 39, 74, 24, 57, 46, 49, 7, 63]	0.0003252610762238490 4
30	[57, 24, 75, 58, 40, 74, 76, 7, 27, 74]	0.0003258893348427155
23	[74, 39, 49, 57, 68, 38, 7, 75, 76, 58]	0.0003259079178007381 4
2	[76, 58, 35, 74, 37, 57, 75, 60, 17, 27]	0.0003262334629682094 4
24	[7, 1, 27, 74, 76, 75, 39, 24, 58, 40]	0.0003263054392941361
16	[74, 39, 40, 60, 57, 38, 74, 75, 34, 27]	0.0003270891715803711 4
6	[24, 81, 0, 57, 37, 41, 74, 74, 39, 75]	0.0003281787392685552
3	[58, 75, 67, 76, 55, 31, 74, 7, 39, 60]	0.0003281973329648310 7
26	[76, 52, 60, 76, 24, 27, 23, 74, 58, 7]	0.0003281973329648310 7
4	[58, 9, 76, 57, 49, 39, 61, 79, 74, 63]	0.0003696841205081152
27	[27, 37, 39, 57, 39, 80, 74, 76, 51, 7]	0.0003725414508469707 6
25	[39, 58, 17, 75, 76, 75, 57, 7, 40, 24]	0.0005330035817840696

Yapılan deney sonuçlarına göre en yüksek uygunluğa sahip 9. çalıştırmaya ait 1000 iterasyon adımından rastgele seçilerek bazıları şekil 25'te gösterilmektedir.



Şekil 25. Optimum Sonucu Veren 9. ÇalıGıtırmanın Bazı Adımları

9. çalıştırmaya ait yerel optimum değerlerinden elde edilen yakınsama grafiği şekil 26'da gösterilmektedir.



Şekil 26. Optimum Sonucu veren 9. Çalıştırmanın Yakınsama Grafiği

Global optimum değeri veren çözüm setindeki her sayı bir şehre karşılık gelir ve pandemi hastanesi yapımı için en uygun şehirleri temsil eder. Optimum çözüm setini veren 9. çalıştırmadan elde edilen şehirler: [74, 57, 8, 28, 0, 39, 15, 74, 18, 45] numaralı şehirlerdir. Elde edilen bu şehirlerin isimleri şu şekildedir:

74: İstanbul;

57: Kocaeli;

8: Çanakkale;

28: Bilecik;

0: Denizli;

39: Ordu;

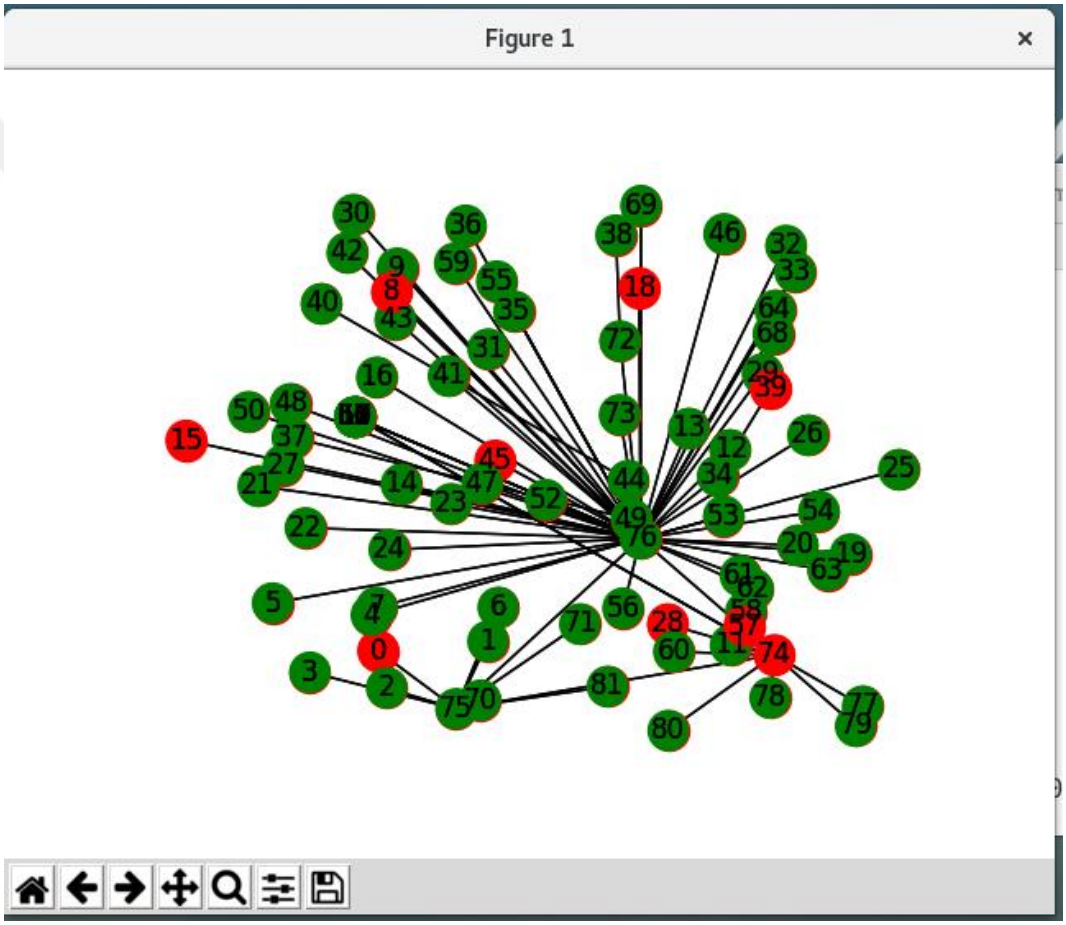
15: Hatay;

74: Ġstanbul;

18: Erzurum;

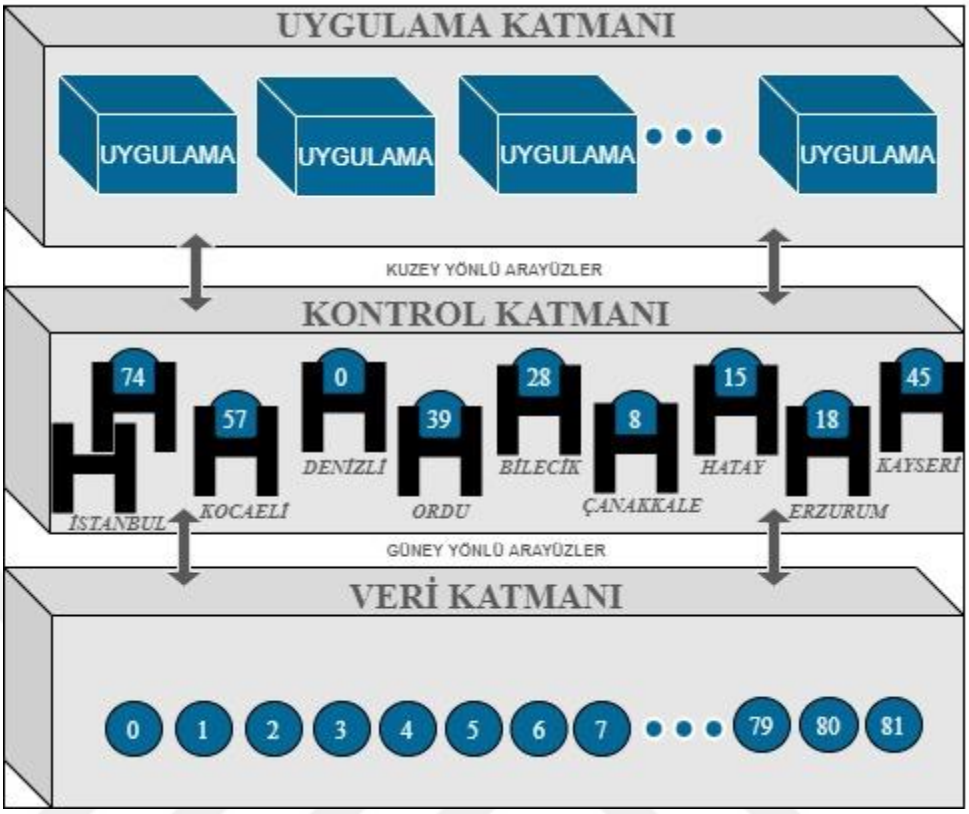
45: Kayseri

Elde edilen Ģehirler Ģekil 27'de ULAKNET veri setinden elde edilen Türkiye ağ haritası üzerinde kırmızı renk ile belirtilmiĢtir.



Şekil 27. En Ġyi Hastane Lokalizasyonları

Deney sonucunda elde edilen değerlerin YTA yapısı mimarisinde tam olarak anlaĢılabilmesi için Ģekil 28'deki modelleme oluĢturulmuĢtur Burada Türkiye'de pandemi hastanesi yerleĢimi için Tablo 4'teki verilere göre bulunan 9. Deneyden elde edilen uygun Ģehirler YTA mimarisinde kontrol katmanına yerleĢtirilmiĢtir. Türkiye'deki diğer tüm Ģehirler ise veri katmanına yerleĢtirilmiĢtir. Veri katmanındaki Ģehirler kontrol katmanında bulunan pandemi hastanelerinden faydalanacaktır.



Şekil 28. YTA Mimarisinde Pandemi Hastaneler İçin Elde Edilen Şehirler

Deney sonucunda elde edilen çözüm kümesi iki adet İstanbul Şehriniiçermektedir. Bu, İstanbul'un iki pandemi hastanesine ihtiyaç duyduğu anlamına gelir. En optimum değeri veren 9. deneyde başkent Ankara elde edilen Şehirlerden değildir. Fakat 30 deneyin 24'ü Ankara'da pandemi hastanesinin kurulması gerektiğini göstermektedir. Deneyde pandemi hastanesi kurulması için seçilen Şehirler ekonomik, sosyal ve kültürel seviyelerine göre elde edilmemektedir. Deneylerden elde edilen değerler, tek bir Şehir olarak değil çözüm setindeki tüm Şehirlere uygulanan algoritmadan elde edilen sonuçlarla birlikte değerlendirilmelidir.

4. BÖLÜM

SONUÇ, TARTIŞMA ve GELECEK ÇALIŞMA

4.1. Sonuç

Bu tez çalışmasında Yazılım Tanımlı Ağlarda (YTA) Çoklu Denetleyici Yerleştirme Problemine (ÇDYP) çözüm önerilmiştir. Topology Zoo veri tabanı bünyesinde bulunan Colt ve ULAKNET veri setlerine ait Avrupa Ağ Haritası ve Türkiye Ağ Haritası kullanılmıştır. Kullanılan ağ haritaları birer YTA mimarisi olarak modellenmiştir. YTA olarak modellenen bu ağ haritalarında belirlenen sayıda denetleyicilerden maksimum kapsam sağlanabilmesi ve en uygun yerleşimini bulabilmek için Dijkstra Algoritması (DA) ve Genetik Algoritma (GA) birlikte kullanılarak çözüm önerilmiştir. DA'nın uygulandığı sorunlara çözüm ararken her adımda en uygun çözümü bulmayı hedeflemesi özelliğinden faydalanılarak ağ haritalarında her bir şehre en kısa mesafede bulunan şehri seçmesi sağlanmıştır. GA için belirlenen parametre değerlerine göre rastgele çözüm kümeleri oluşturulmuşolup her bir jenerasyonda GA'nın temel adımları olan mutasyon, çaprazlama ve seçme adımlarına göre uygunluk değerleri hesaplanmıştır. Her bir jenerasyonda elde edilen yerel optimum değerlerden en optimum olan değer global optimum değer olarak seçilerek elde etmek istediğimiz çözüm olarak seçilmiştir Böylece global optimum çözümde bulunan elemanlar ağ haritasındaki denetleyicileri yerleştireceğimiz şehirler iken bulunan çözüm değeri o çözüm kümesinin uygunluğudur.

Ağ haritalarının YTA mimarisi ile modellenmesinin ardından Türkiye'de kurulması gereken pandemi hastanelerinin yerleştirilmesi problemi YTA mimarisi ile modellenmiştir. Pandemi hastanelerinin kurulum maliyetinin yüksek olması nedeniyle istenilen her lokasyonda kurulması mümkün olmadığından nasıl yerleştirileceği büyük önem taşımaktadır. Pandemi hastanesi yerleştirme problemini çözmek için ULAKNET veri setinden elde edilen Türkiye ağ haritası koordinat verilerine her bir şehrin

koronavirüs katsayısı, parametre olarak eklenmiştir. Elde edilen veri dosyasına yine DA ile GA birlikte uygulanmıştır. Deneysel sonuçlardan bulunan hastane konumları, koronavirüsün yüksek olduğu yerlerdeki tüm şehirleri kapsayacak en yakın mesafede olma özelliğine sahiptir. Böylece kurulan pandemi hastaneleri hep daha çok koronavirüslü hastaya hitap edecek hem de ulaşımı kolay olacaktır.

4.2. Tartışma

Günümüzdeki teknolojik gelişmeler ve ağ ile yönetilebilen cihazların artması ile ağ yönetimi daha karmaşık hale gelmiştir. Geleneksel ağ mimarisinin yeni cihazların ağa katılması, ağdan çıkarılması ve cihazların birbirleriyle haberleşmesi hususunda yaşadığı zorluklar YTA mimarisine yöneltmiştir. YTA mimarisinde optimal ağ planlarken kontrol katmanında bulunan denetleyicilerin yerleşimi oldukça önemlidir. Bu denetleyicilerin en uygun yerleşim yeri için GA ve DA'nın birlikte kullanılmıştır. GA'nın yeni bir birey oluşturmak için mevcut nüfustan daha yüksek formda olan ebeveynleri seçen bir popülasyonun doğaya dayalı evrimi olması YTA'larda ÇDYP için kullanılması oldukça mantıklıdır. Çünkü GA'nın bu özelliği sayesinde gerçekleştirilen deneylerin her bir iterasyonunda daha yüksek formda çözüm kümeleri elde edilmiştir. Kullanılan DA'nın her bir aşamada en yakın mesafeyi bulması özelliği sayesinde denetleyiciler, diğer denetleyicilere ve ağ elemanlarına maksimum kapsam ile fayda sağlayacak şekilde yerleştirilmiştir. GA ile DA'nın birlikte kullanılması ise denetleyicilerin en uygun yerleşim için daha da uygunluk sağlamaktadır.

Yapılan çalışmada Türkiye'de pandemi hastanelerinin nasıl yerleştirilmesi gerektiği YTA mimarisi ile modellenmiştir. Bu modelleme ile pandemi hastanesinin kurulum yerlerinin seçimi dinamik olarak yönetilebilmektedir. Ülkelerin ve şehirlerin koronavirüs sayılarına net bir şekilde ulaşılması durumunda DA ve GA'nın ÇDYP'nin çözümünde başarımı daha net bir şekilde anlaşılacaktır.

4.3. Gelecek Çalışma

Gelecek çalışma olarak ülkelerin koronavirüs verilerine net bir şekilde ulaşılmasının mümkün olması durumunda tüm ülkeler için pandemi hastanelerinin en uygun yerleşim yerleri bulunacaktır. Bunun için belli zaman aralığındaki koronavirüs bulaşmış hasta

sayısının yanı sıra o şehirlerin nüfus miktarı, sosyal ve ekonomik şartları göz önünde bulundurularak bir algoritma geliştirmek planlanmaktadır.

Ayrıca YTA'larda ÇDYP'ye Yapay Arı Koloni Algoritması uygulanacak olup tez çalışması sonunda elde edilen sonuçlarla karşılaştırılacaktır.

KAYNAKÇA

1. Zhang, Y., Cui, L., Wang, W., Zhang, Y., 2018. A survey on software defined networking with multiple controllers. **Journal of Network and Computer Applications**, **103**, 101-118.
2. Lu, J., Zhang, Z., Hu, T., Yi, P., Lan, J., 2019. A survey of controller placement problem in software-defined networking. **IEEE Access**, **7**, 24290–24307.
3. Guo, Z., Liub, R., Xuc, Y., Gushchind, A., Walide, A., 2017. STAR: Preventing flow-table overflow in software-defined networks. **Computer Network**, **125**, 15–25.
4. Coronavirus Disease (COVID-19) Pandemic. (Web Sayfası: **www.who.int/covid-19**), (EriĢimTarihi: Haziran 2020).
5. Pandemi Hastaneleri. (Web Sayfası: **https://hasta.saglik.gov.tr/Eklenti/36907/0/pandemi-hastaneleripdf.pdf)**, (EriĢimTarihi: Temmuz 2020).
6. Liao, L., Leung, V.C.M., 2017. Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in SDNs. **IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)**, 1-6.
7. Mouawad, N., Naja, R., Tohme, S., 2018. Optimal and dynamic SDN controller placement. **2018 International Conference on Computer and Applications (ICCA),** 1-9.
8. Huang, V., Chen, G., Fu, Q., Wen, E., 2019. Optimizing controller placement for software-defined networks. **International Conference on Cloud Computing and Big Data**, 224-232.
9. Adebayo, I.O., Adigun, M.O., Mudali, P., 2018. Feature selection strategies for the controller placement problem in SDNs: A review. **2018 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD**, 1-5.
10. Kuang, H., Qiu, Y., Li, R., Liu, X., 2018. A hierarchical K-Means algorithm for controller placement in SDN-based WAN architecture. **2018 10th International**

Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 263–267.

11. Wang, G., Zhao, Y., Huang, J., Wang, W., 2017. The controller placement problem in software defined networking: a survey. **IEEE Network**, **31**(5), 21-27.

12. Singh, A.K., Srivastava, S., 2018. A survey and classification of controller placement problem in SDN. **International Journal of Network Management**, **28**(3), 1-25.

13. Babayigit, B., Karakaya, S., 2018. Cost-effective logging using SDN architecture. **International Conference on Advanced Technologies**, 1.

14. Babayigit, B., Karakaya, S., Ulu, B., 2018. An implementation of software defined network with Raspberry Pi. **2018 IEEE 26th Signal Processing and Communications Applications Conference**, 1-4.

15. Babayigit, B., Ulu, B., 2018. Load balancing on software defined networks. **2nd International Symposium on Multidisciplinary Studies and Innovative Technologies**, 1-4.

16. Yao, G., Bi, J., Li, Y., Guo, L., 2014. On the capacitated controller placement problem in software defined networks. **IEEE Communications Letters**, **18**(8).1339-1342.

17. Sallahi, A., St-Hilaire, M., 2015. Optimal model for the controller placement problem in software defined networks**. IEEE Communications Letters**, **19**(1), 30-33.

18. Sallahi, A., St-Hilaire, M., 2017. Expansion model for the controller placement problem in software defined networks. **IEEE Communications Letters**, **21**(2), 274-277.

19. Genetik Algoritmalar (Genetic Algorithms). (Web sayfası: **https://bilgisayarkavramlari.com/2009/02/16/genetik-algoritmalar-genetic-algorithms/?highlight=genetik**) (Erişim Tarihi: Haziran 2019)

20. Dijkstra, E. W., 1959, **A note on two problems in connexion with graphs. Numerische Mathematik**, **1**(1), 269–271.

21. Dijkstra Algoritması. (Web Sayfası: **bilgisayarkavramlari.com/2010/05/13/dijkstra-algoritmasi-2/#:~:text=Bilgisayar%20bilimlerinde%20kullan%C4%B1lan%20ve%20algoritmay%C4%B1,shortest%20path)%20bulmak%20i%C3%A7in%20kullan%C4%B1l%C4%B1r.&text=Dijkstra%20algoritmas%C4%B1%20herhangi%20bir%20%C5%9Fekildeki,giden%20en%20k%C4%B1sa%20yolu%20hesaplar.**), (EriĢim Tarihi: Haziran 2019)

22. The Internet Topology Zoo. (Web Sayfası: **www.topology-zoo.org**), (EriĢim Tarihi: Mayıs 2019)

23. Hangi ilde ne kadar coronavirüs vakası var? Vefat Sayıları. (Web Sayfası: **https://ajansspor.com/haber/hangi-ilde-ne-kadar-corona-virus-vakasi-var-vefat-sayilari-362702, Access Date: June 2020.**) (EriĢimTarihi: Ağustos 2020)

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı :Eda Nur HASÇOKADAR

Uyruğu :TC

Doğum Tarihi ve Yeri :

Tel :

E-Posta :

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Yüksek Lisans	Erciyes Üniversitesi, Bilgisayar Mühendisliği	2021
Lisans	Fırat Üniversitesi, Bilgisayar Mühendisliği	2016
Lise	Nuh Mehmet Baldöktü Anadolu Lisesi	2012

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2017-Halen	Kayseri şeker Fabrikası	Bilgisayar Müh.

YABANCI DİL

İngilizce

TEZDEN YAPILAN YAYINLAR

1. B. Babayigit**, E.N. Hasçokadar,** A Software-Defined Network Approach for The Best Hospital Localization Against Coronavirus (COVID-19), Advances in Science, Technology and Engineering Systems Journal, 5(6), 1537–1544, 2020**,** doi: 10.25046/aj0506184
2. B. Babayiğit, B. Ulu**, E.N. Hasçokadar, "**Solving multi-controller placement problem in software defined networks with a genetic algorithm," in 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, 666–670, 2019, doi: 10.1109/UBMK.2019.8907199.