
GAZİ ÜNİVERSİTESİ
Gazi
1926

GRAF TEORİNİN YAPAY ZEKÂ ÜZERİNE UYGULAMALARI

Kübra Nur CANBAY

YÜKSEK LİSANS TEZİ
MATEMATİK ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2024

ETİK BEYAN

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

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu, bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Kübra Nur CANBAY

23/01/2024

GRAF TEORİNİN YAPAY ZEKÂ ÜZERİNE UYGULAMALARI

(Yüksek Lisans Tezi)

Kübra Nur CANBAY

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ocak 2024

ÖZET

Graf teorisi ve yapay zekâ, son zamanlarda önemli gelişmeler gösteren birbiriyle yakından ilişkili iki alandır. Graf teorisi, karmaşık ağları temsil etmek ve analiz etmek için güçlü bir matematiksel çerçeve sunarken, yapay zekâ teknikleri verileri işleme, analiz etme ve iç görüler çıkarma konusunda devrim niteliğinde bir etki ortaya çıkarmıştır. Bu tez, graf teorisi ile yapay zekâ arasındaki ilişkiyi keşfeder ve bu iki alanın birbirini nasıl etkilediğini vurgular. Graf teorisinin; graflar, düğümler, kenarlar ve çeşitli graf algoritmaları gibi temel kavramlarına genel bir bakış sunar. Ayrıca graf teorisinin rolünü, makine öğrenmesi algoritmalarının, doğal dil işleme ve bilgisayarla görme gibi alanlarda nasıl kullanıldığını açıklar. Bu iki alanın güçlü yönlerini birleştirerek araştırmacılar, karmaşık verileri analiz etmek ve anlamak için yeni teknikler geliştirebilir. Bu çalışma, graf teorisinin matematiksel temelleri ve Graf Konvolüsyon Ağları üzerine odaklanmaktadır. GCN modelinin olası bir İstanbul depremi senaryosunda kurtarma ekibi rotalarını oluşturmak için nasıl kullanıldığı, modelin performansı ve elde edilen sonuçlar açıklanmıştır.

Bilim Kodu	:	20401
Anahtar Kelimeler	:	Graf teorisi, çizge kuramı, yapay zekâ, yapay zekâ uygulamaları, graf algortiaları, graf konvolüsyon ağları
Sayfa Adedi	:	62
Danışman	:	Dr. Öğr. Üyesi Gül ÖZKAN KIZILIRMAK

APPLICATIONS OF ARTIFICIAL INTELLIGENCE ON GRAPH THEORY

(M. Sc. Thesis)

Kübra Nur CANBAY

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

January 2024

ABSTRACT

Graph theory and artificial intelligence are two closely related fields that have seen significant developments recently. While graph theory provides a powerful mathematical framework for representing and analyzing complex networks, artificial intelligence techniques have revolutionized the way we process, analyze, and extract insights from data. This thesis explores the relationship between graph theory and artificial intelligence and highlights how these two fields influence each other. Graph theory: It provides an overview of basic concepts such as graphs, nodes, edges, and various graph algorithms. It also explains the role of graph theory and how machine learning algorithms are used in fields such as natural language processing and computer vision. By combining the strengths of these two fields, researchers can develop new techniques for analyzing and understanding complex data. This study focuses on the mathematical foundations of graph theory and Graph Convolution Networks. How the GCN model is used to create rescue team routes in a possible Istanbul earthquake scenario, the performance of the model, and the results obtained are explained.

Science Code	:	20401
Key Words	:	Graph theory, artificial intelligence, artificial intelligence applications, Graph algorithms, Graph convolution networks
Page Number	:	62
Supervisor	:	Assist. Prof. Dr. Gül ÖZKAN KIZILIRMAK

TEŞEKKÜR

Bu çalışma boyunca bana desteğini sağlayan Siber Güvenlik Mühendisi Osman Kaan KARS'a, tez yazımını, bilimsel araştırmayı ve kaynak göstermenin önemini öğreten, CERN'de çalışan Fizikçi Ümit SÖZBİLİR'e, bu alanda var olmama vesile olan matematiksel düşünceler ile ufkumu açan dostum, Matematikçi Merve KAYA'ya, yüksek lisans eğitimim boyunca sağladıkları imkanlar için çalışmakta olduğum şirketim Kron Teknoliji'ye, hayatım boyunca eğitimimi destekleyen, her koşulda yanımda olan babam Kenan CANBAY'a, annem Fatma CANBAY'a, ve canım kardeşim Halil İbrahim CANBAY'a, son olarak desteğinin ve sevgisinin sağladığı motivasyonu hayatım boyunca hissedeceğim biricik babaannem Kübra CANBAY'a teşekkürü borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET iv

ABSTRACT v

TEŞEKKÜR vi

İÇİNDEKİLER vii

ŞEKİLLERİN LİSTESİ ix

SİMGELER VE KISALTMALAR xi

1. GİRİŞ 1

2. GRAF TEORİSİNİN MATEMATİKSEL TEMELLERİ 3

2.1. Grafın Tarihçesi 3

2.2. Graf Kavramı 3

2.3. Komşuluk 4

2.4. Graf Çeşitleri 8

2.5. Grafların Matris Gösterimi 14

2.6. Ağırlıklı Grafların Matris Gösterimi 17

3. GRAF ALGORİTMALARI 21

3.1. Önce Derinlik Araması (DFS) 21

3.1.1. DFS uygulamaları 21

3.2. Önce Genişlik Araması (BFS) 23

3.2.1. BFS uygulamaları 24

4. GRAF MAKİNE ÖĞRENİMİ 27

4.1. Denetimli Graf Makine Öğrenimi 28

4.2. Denetimsiz Graf Makine Öğrenimi 29

4.3. Sıkıştırma ve Gömme 31

5. GRAF TABANLI DERİN ÖĞRENME 33

5.1. Graf Sinir Ağları (GNN) 33

Sayfa

5.1.1. Graf konvolüsyonel ağları (GCN) 35
6. ANALİZ 39
6.1. Veri Seti 39
6.2. Araçlar ve Kütüphaneler 39
6.3. GCN Modeli 40
7. SONUÇ VE ÖNERİLER 45
KAYNAKLAR 47
EKLER 48
ÖZGEÇMİŞ 62

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil 2.1. 4 noktalı, 5 kenarlı bir graf. 4

Şekil 2.2. G grafının komşulukları. 5

Şekil 2.3 G grafı. 6

Şekil 2.4. İlmek ve katlı kenara sahip graf. 7

Şekil 2.5. İzole nokta ve uç nokta içeren graf. 8

Şekil 2.6. Basit graf. 8

Şekil 2.7. Çoklu graf 9

Şekil 2.8. G grafı 10

Şekil 1.9. İki parçalı graf. 12

Şekil 2.10. Ağırlıklı graf 13

Şekil 2.11. G grafı ve etki matrisi 14

Şekil 2.12. G grafı ve bitişiklik matrisi 15

Şekil 2.13. G grafı ve derece matrisi 15

Şekil 2.14. G graf ve Laplacian matrisi. 16

Şekil 2.15. G grafı ve uzaklık matrisi. 17

Şekil 2.16. G grafı ve ağırlıklı bitişiklik matrisi 18

Şekil 2.17. G grafı ve ağırlıklı Laplacian matrisi. 18

Şekil 2.18. G grafı ve ağıtlıklı grafın uzunluk matrisi. 19

Şekil 3.1. G grafı 22

Şekil 4.1. Yapay zekâ-makine öğrenimi Venn şeması. 27

Şekil 4.2. Nokta özelliği tahmini 28

Şekil 4.3. Bağlantı tahmini. 29

Şekil 4.4. Graf özelliği tahmini 29

Şekil 4.5. Temsil öğrenme 30

Şekil 4.6. Benzerlik 30

Şekil **Sayfa**

Şekil 4.7. Topluluk belirleme..... 30

Şekil 4.8. Merkez nokta belirleme 30

Şekil 6.1. Analiz Şeması 39

Şekil 6.2. Softmaks fonksiyonu. 41

Şekil 6.3. ReLu fonksiyonu. 42

Şekil 6.4. Optimal graf..... 43

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
BFS	Önce Genişlik Araması
DFS	Önce Derinlik Araması
GCN	Graf Konvolüsyon Ağları
GML	Graf Makine Öğrenimi
GNN	Grad Sinir Ağları

1. GİRİŞ

Modern çağda, sadece şehirlerin yollarını ya da sosyal medya platformlarının kullanıcılarını değil, moleküllerin ve proteinlerin bağlantılarını bile graf yapılarıyla temsil ediyoruz. Ancak bu karmaşık bağlantı ağları, geleneksel yapay zekâ yöntemleriyle işlenmek için çok karmaşıktır. Neredeyse her yerde bulunan bu graf yapıları hem fiziksel hem de dijital dünyalarımızın anlaşılmasında hayati bir rol oynamaktadır.

Dünyamızın her köşesinde, tıptan teknolojiye, ekonomiden ekolojiye kadar graf yapılarıyla karşılaşıyoruz. Bu yapıları daha iyi anlamak, geleceğin bilimine ve teknolojisine yön verecektir. Geleneksel Konvolüsyonel Sinir Ağları, genellikle düzenli, ızgara benzeri yapıya sahip görüntülerle mükemmel çalışırken, düzensiz ve karmaşık graf yapılarıyla başa çıkmakta yetersiz kalmaktadır. Bu, sosyal ağlardan biyoinformatiğe kadar birçok alanda bir sorun haline gelmiştir. Peki, bu karmaşık yapıları nasıl anlamlandırabiliriz?

Bu tez, graf teorisinin matematiksel temelleri ve Graf Konvolüsyon Ağları üzerine odaklanmaktadır ve tezin ana amacı, Graf Konvolüsyon Ağları (GCN) gibi graf-tabanlı derin öğrenme yaklaşımlarının matematiksel temellerini keşfederek, graf veri yapılarını daha derinlemesine anlamamıza yardımcı olmaktır. Esas problem, GCN modelinin olası bir İstanbul depremi senaryosunda kurtarma ekibi rotalarını oluşturmak için nasıl kullanıldığı olup, modelin performansı ve elde edilen sonuçlar son bölümde açıklanmıştır.

2. GRAF TEORİSİNİN MATEMATİKSEL TEMELLERİ

Bu bölümde Graf teorinin tarihçesi, matematiksel temelleri ve ifadeleri açıklanmıştır.

2.1. Grafın Tarihçesi

Graf teorinin ortaya çıkışı veya temellendirildiği olay için genel görüş 1736 yılında Euler tarafından çözüme kavuşturulmuş daha doğrusu yanlışlığı doğrulanmış olan Königsberg Köprüsü problemidir [1]. Bu problem Prusya'da yaşayan meraklı halk tarafından ortaya çıkmıştır. Ortaya atılan problem şöyledir: Bütün köprülerden bir ve yalnız bir defa geçmek koşulu ile bir yürüyüş yapılabilir mi? 1736'da Euler incelemeleri ile böyle bir gezintinin mümkün olmadığını kanıtlamış ve hangi durumlar olsaydı mümkün olurdu üzerinde çalışmalar yaparak problemin çözümünü dönemin ünlü matematik ve bilim dergilerinde yayımlatmıştır. Bu problem Leonhard Euler'in araştırmaları ile matematikte tamamıyla yeni bir dal olan graf teorinin ilk teoremi ve topolojinin keşfinin habercisi olmuştur.En basit anlamda ifade ile Euler'in çözümü köşelerin dereceleri üzerinde temellenmiştir. Başlangıç köşesi eğer son nokta olma şartı altında değilse (çıkış), (çıkış, dönüş, çıkış), (çıkış, dönüş, çıkış, dönüş, çıkış), şeklinde tek sayıda derece sahip olması gerekiyor ve aynı zamanda son nokta da başlangıç noktasının ikizi gibi davranmalı ve tek dereceli olmalıdır. Orta köşeler ise çift dereceli olmalıydı ki çıkış ve dönüş yolları eşit miktarda olabilsin. Sonuç olarak sahip olduğumuz grafta 2 adet tek sayı dereceli köşe ve diğer köşeler de çift sayı dereceli ise bütün köşeler köprülerin her birinden bir ve sadece bir kez geçilme koşulu altında gerçekleşebilir.

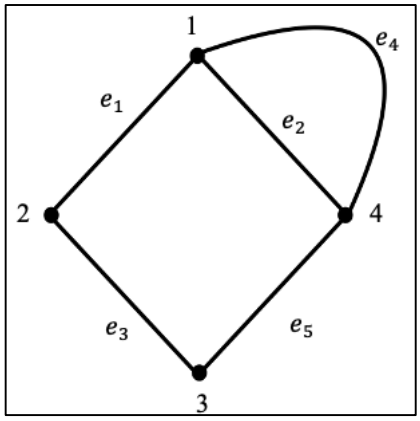
2.2. Graf Kavramı

2.2.1. Tanım

Graf, elemanları nokta olarak adlandırılan sonlu, boş olmayan $V = \{1,2, \dots, n\}$ noktalar kümesi ve elemanları kenar olarak adlandırılan sonlu $E = \{\{i,j\} : i,j \in V\}$ kenarlar kümesinden oluşan $G = (V,E)$ ikili yapısına denir ve kısaca $G = (V,E)$ ile gösterilir.

Ayrıca $\forall\, i,j \in V$ için E'nin i ve j noktalarına karşılık gelen elemanları e_{ij} veya e_{ji} şeklinde gösterilir [2].

Örnek



Şekil 2.1. 4 noktalı, 5 kenarlı bir graf.

Şekil 2.1.'de verilen G'nin noktalar ve kenarlar kümesi:

$V = \{1,2,3,4,\}$
$E = \{e_1, e_2, e_3, e_4, e_5\}$

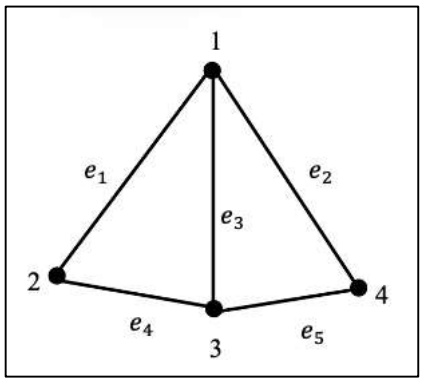
2.3. Komşuluk

2.3.1. Tanım

$G = (V, E)$ bir graf olmak üzere, G'nin herhangi i ve j noktaları arasında en az bir kenar bulunuyorsa i ve j noktalarına komşudur, denir ve $i \sim j$ ile gösterilir [2].

2.3.2. Tanım

Bir i noktasına komşu olan noktaların kümesine i'nin komşuluklar kümesi denir ve N_i ile gösterilir.



Şekil 2.2. G grafının komşulukları.

G grafının komşulukları ve komşuluk kümeleri aşağıdaki gibidir:

$i = 1$ için,
$1\sim2$, $1\sim 3$, $1\sim4$ olmak üzere $N_1 = \{2,3,4\}$
$i = 2$ için,
$2\sim1$, $2\sim 3$ olmak üzere $N_2 = \{1,3\}$
$i = 3$ için,
$3\sim1$, $3\sim 4$ olmak üzere $N_3 = \{1,4\}$
$i = 4$ için,
$4\sim1$, $4\sim 3$ olmak üzere $N_4 = \{1,3\}$

2.3.3. Tanım

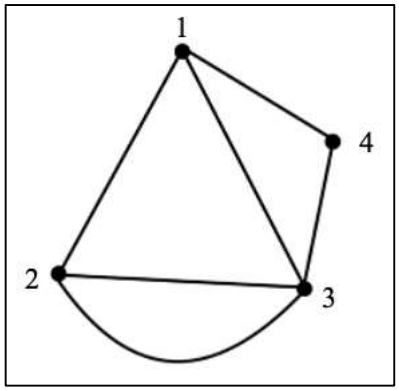
$G = (V, E)$ grafının herhangi bir i noktasına bağlı kenar sayısına i'nin derecesi denir ve $d_G(i)$ ile gösterilir.

2.3.4. Tanım

Bir i noktasına komşu olan noktaların derecelerinin ortalamasına i noktasının ortalama derecesi denir, m_i ile gösterilir ve aşağıdaki şekilde ifade edilir.

$$m_i = \frac{1}{d_G(i)} \sum_{j:\, j\sim i} d_G(j)$$

Örnek



Şekil 2.3. G grafı.

$d_G(1) = 3$, $d_G(2) = 3$, $d_G(3) = 4$ ve $d_G(4) = 2$ olmak üzere,

$$m_1 = \frac{1}{d_G(1)}\big(d_G(2) + d_G(3) + d_G(4)\big) = \frac{8}{3}$$

$$m_2 = \frac{1}{d_G(2)}\big(d_G(1) + d_G(3)\big) = \frac{7}{2}$$

$$m_3 = \frac{1}{d_G(3)}\big(d_G(1) + d_G(2) + d_G(4)\big) = \frac{8}{3}$$

$$m_4 = \frac{1}{d_G(4)}\big(d_G(1) + d_G(3)\big) = \frac{7}{2}$$

2.3.5. Tanım

Bir G grafının maksimum (1) ve minimum (2) ifadeleri aşağıdaki şekilde tanımlanır:

$$\delta(G) = max\{d_G(i): i \in V\} \quad (1)$$

$$\Delta(G) = min\{d_G(i): i \in V\} \quad (2)$$

2.3.6. Tanım

Bir G grafının ortalama derecesi $|V|$, V nin eleman sayısı olmak üzere aşağıdaki şekilde tanımlanır:

$$d(G) = \frac{1}{|V|}\sum_{i \in V} d_G(i)$$

2.3.7. Tanım

$|E|$, E'nin eleman sayısı olmak kenarlar kümesinin eleman sayısı aşağıdaki şekilde tanımlanır:

$$|E| = \frac{1}{2}\sum_{i \in V} d_G(i) = \frac{1}{2} d(G)|V|$$

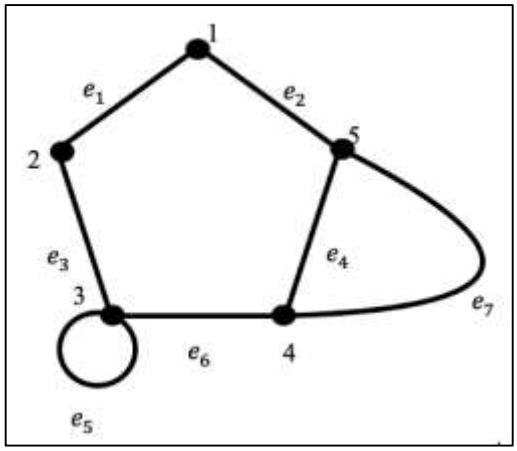
2.3.8. Tanım

Başlangıç ve bitiş noktası aynı olan kenara ilmek denir.

2.3.9. Tanım

Grafın her iki noktası arasında birden fazla kenar bulunuyor ise bu kenarlara katlı kenar, çoklu kenar ve paralel kenar denir.

Örnek



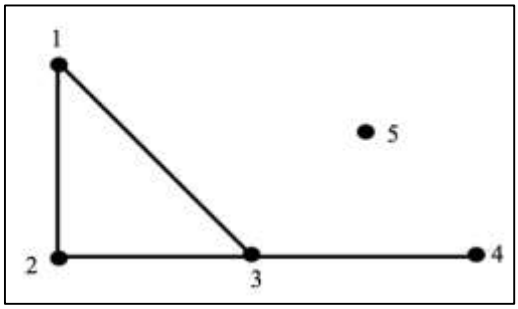
Şekil 2.4. İlmek ve katlı kenara sahip graf

G'nin e_4 ve e_7 kenarları katlı kenar olup e_5 kenarı da ilmektir.

2.3.11. Tanım

G bir graf olmak üzere, $d_G(i) = 1$ olan noktaya uç nokta, denir.

Örnek



Şekil 2.5. İzole nokta ve uç nokta içeren graf.

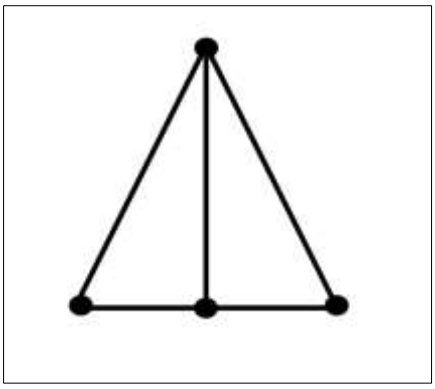
G'nin 5, izole ve 4, uç noktasıdır.

2.4. Graf Çeşitleri

2.4.1. Tanım

Herhangi iki noktası arasında en fazla bir kenar bulunan ve ilmek içermeyen graf, basit graf olarak adlandırılır.

Örnek

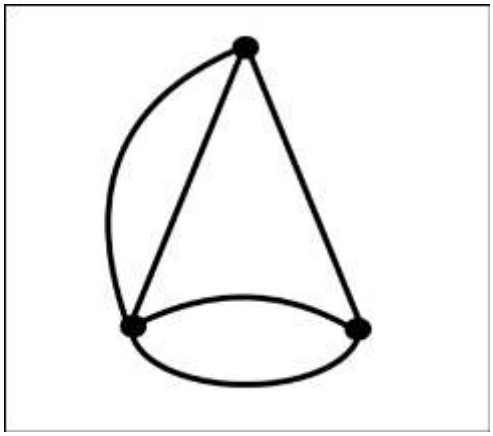


Şekil 2.6. Basit graf

2.4.2. Tanım

Herhangi iki noktası arasında birden fazla kenar bulunan graf, çoklu graf olarak adlandırılır.

Örnek



Şekil 2.7. Çoklu graf

2.4.3. Tanım

Basit bir grafın herhangi iki noktası arasında bir kenar bulunuyor ise bu grafa, tam graf denir ve n noktalı bir tam graf K_n ile gösterilir.

K_n grafının kenar sayısı:

$$C\binom{n}{2} = \frac{n(n-1)}{2}$$

2.4.4. Tanım

Yalnızca izole nokta içeren grafa boş graf denir ve n noktalı bir boş graf N_n ile gösterilir.

2.4.5. Tanım

Bir grafın her bir noktası aynı derece sahip ise bu grafa, regüler graf denir.

2.4.6. Tanım

Bir grafın sonlu sayıdaki birbiri ile bağlantılı noktalarından ve kenarlarından oluşan diziye yürüme denir, W ile gösterilir.

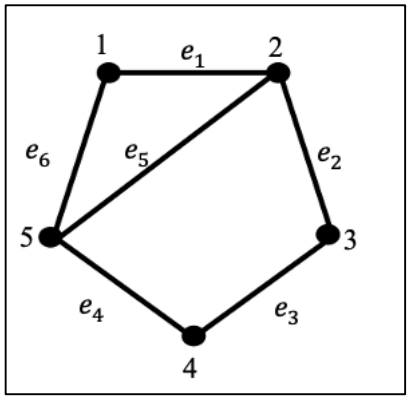
2.4.7. Tanım

Her bir kenarın ve noktanın en fazla bir kez kullanıldığı yürümeye yol denir, P ile gösterilir.

2.4.8. Tanım

Başlangıç ve bitiş noktası aynı olan yola devir denir, C_n ile gösterilir. Devirdeki her bir i noktası için $d_i = 2$'dir.

Örnek



Şekil 2.8. G grafı

G'nin yürüme, yol ve deviri aşağıdaki gibidir:

$W = 1e_1 2e_2 3e_3 4e_4 5e_5 2e_1 1e_6$

$P = 1e_6 5e_4 4e_3 3$

$C_n = 1e_1 2e_2 3e_3 4e_4 5e_6 1$

2.4.9. Tanım

Herhangi iki noktası arasında yol bulunan grafa bağlantılı graf denir

2.4.10. Tanım

İçinde devir bulundurmayan bağlantılı grafa ağaç graf veya ağaç denir, n noktalı bir ağaç T_n ile gösterilir.

2.4.11. Tanım

$n \geq 3$ olmak üzere C_n devrine diğer noktalar ile komşu olacak yeni bir noktanın eklenmesi sonucu oluşan grafa tekerlek denir, W_n ile gösterilir. W_n, $n+1$ nokta ve $2n$ kenar içerir.

2.4.12. Tanım

0 ve 1'lerden oluşan sayı dizisine bit dizisi denir.

2.4.13. Tanım

n uzunluğundaki 2^n bit dizileri ile temsil edilen noktalardan oluşan grafa N-Küp denir, Q_n ile gösterilir.

Q_n, 2^n tane nokta ve $n2^n$ tane kenar içerir.

Q_n, $(n-1)$- dereceli regüler bir graftır.

2.4.14. Tanım

Grafın noktalar kümesi, her bir noktası aynı kümedeki diğer noktalar ile komşu olmayacak şekilde U ve W ayrık kümelerine bölünebiliyorsa iki parçalı graftır. Bu graf;

$U \cap W = \emptyset$ ve $U \cup W = V$

olmak üzere

$G = (U, W, E)$

şeklinde gösterilir.

2.4.15. Tanım

$|U| = m$ ve $|V| = n$

olmak üzere

$G = (U, W, E)$ için $\forall i \in U$

noktasının derecesi

$d_i = n$ ve $\forall j \in W$

noktasının derecesi

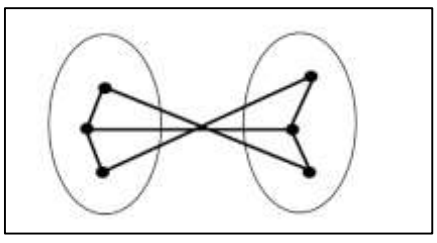
$d_j = m$

ise bu graf, parçalı tam graftır.

$K_{m,n}$

ile gösterilir.

Örnek



Şekil 1.9. İki parçalı graf

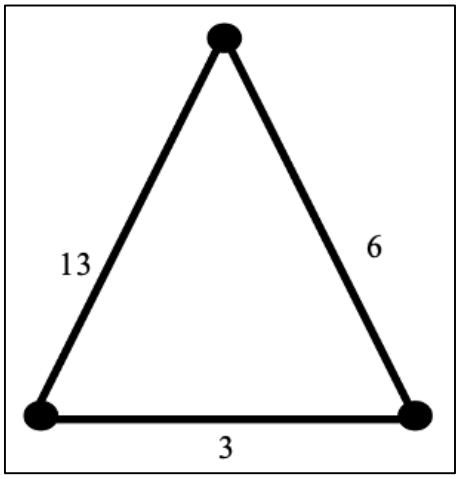
2.4.16. Tanım

Her bir kenarına yön verilerek oluşturulan grafa yönlü graf denir. Yön, kenarın başlangıç noktasında bitim noktasına doğru verilir.

2.4.17 Tanım

Her bir kenarına reel sayı veya kare matrisler ağırlık olarak atanarak oluşturulan grafa ağırlıklı graf denir.

Örnek



Şekil 2.10. Ağırlıklı graf

2.4.18. Tanım

16 köşe ve 24 kenardan oluşan grafa möbiüs graf denir. Simetrik bipartit kübik bir graftır.

2.4.19. Tanım

G bir graf olmak üzere aşağıda verilen koşulları sağlıyor ise $L(G)$'ye line graf denir.

$L(G)$'nin her bir noktası G'nin bir kenarını temsil eder.

$L(G)$'nin her iki noktasının komşu olabilmesi için gerek ve yeter koşul onların yöndeş kenarlarının G'de ortak bir bitiş noktası vardır.

2.5. Grafların Matris Gösterimi

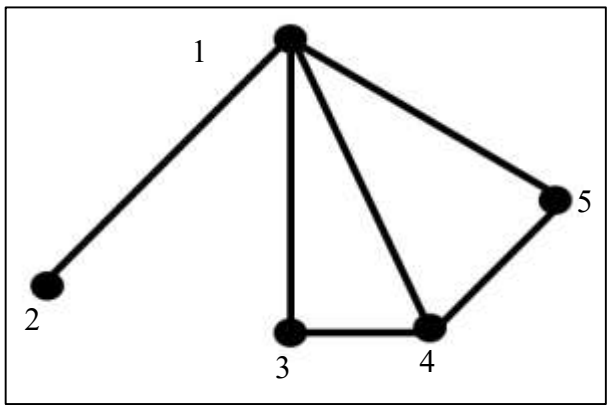
2.5.1. Tanım

G, n noktalı ve m kenarlı basit bir graf olsun. G'nin etki matrisi, elemanları i noktasının e_j kenarının bitim noktası ise 1, diğer durumlar için 0 olarak tanımlıdır,

$$M(G) = \left(m_{ie_j}\right)_{nxm}$$

ile gösterilir [2].

Örnek



$$\mathrm{M(G)} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Şekil 2.11. G grafı ve etki matrisi

2.5.2. Tanım

G, n noktalı basit bir graf olsun, G'nin bitişiklik matrisinin elemanları,

$$(a_{ij})_{nxn} = \begin{cases} 1; & i \sim j \\ 0; & aksi\ hal \end{cases}$$

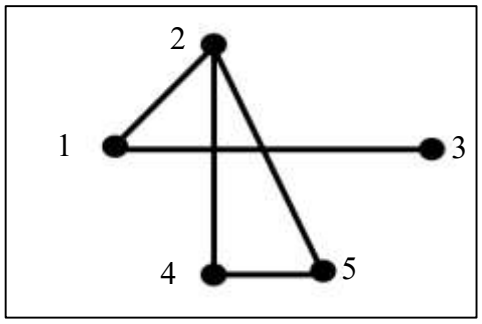
olmak üzere,

$A(G) = (a_{ij})_{nxn}$

şeklinde tanımlanır [2].

Bitişiklik matrisi reel, simetrik bit matris olduğundan tüm öz değerleri reeldir.

Örnek



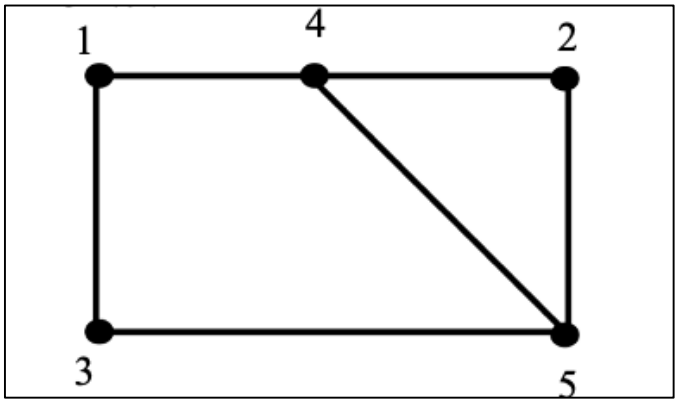
$$\mathrm{A}(G) = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Şekil 2.12. G grafı ve bitişiklik matrisi

2.5.3. Tanım

G, n noktalı bir graf olsun. G'ye ait her bir noktanın derecesinin oluşturduğu köşegen matrise G'nin derece matrisi denir ve $DG) = k\ddot{o}ş(d_1, d_2, d_3, \ldots . d_n)$ şeklinde ifade edilir [2].

Örnek



$$\mathrm{D}(G) = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Şekil 2.13. G grafı ve derece matrisi

2.5.4. Tanım

G, n noktalı bir graf olsun, G'nin Laplacian matrisi,

$$l_{ij} = \begin{cases} d_i; & i = j \\ -1; & i \sim j \\ 0; & aksi\ hal \end{cases}$$

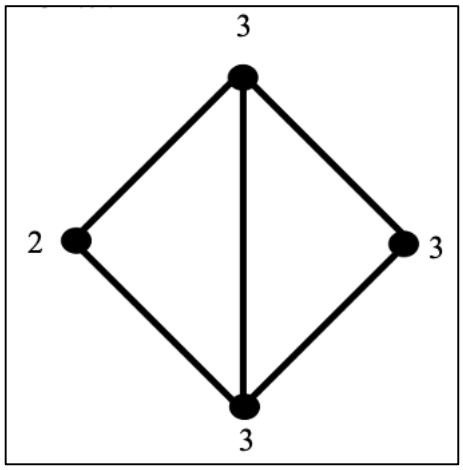
olmak üzere,

$$L(G) = (l_{ij})_{nxn}$$

şeklinde tanımlanır.

Bitişiklik matrisi reel, simetrik bit matris olduğundan tüm öz değerleri reeldir.

Örnek



$$\mathrm{D}(G) = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

Şekil 2.14. G graf ve Laplacian matrisi.

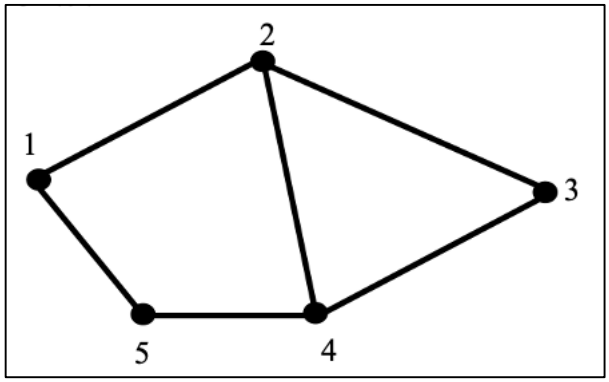
2.5.5. Tanım

G, bağlantılı bir graf $V(G) = \{v_1, v_2, \dots, v_n\}$ ve d_{ij}, i ve j arasındaki en kısa yolun uzaklığı olmak üzere G'nin uzaklık matrisi,

$$D(G) = \begin{cases} d_{iJ}; & i \neq j \\ 0; & i = j \end{cases}$$

şeklinde tanımlanır.

Örnek



$$\mathrm{D}(G) = \begin{bmatrix} 0 & 1 & 2 & 2 & 1 \\ 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 \\ 1 & 2 & 2 & 1 & 0 \end{bmatrix}$$

Şekil 2.15. G grafı ve uzaklık matrisi.

$D(G)$, reel ve simetrik bir matristir.

2.6. Ağırlıklı Grafların Matris Gösterimi

2.6.1. Tanım

G, n noktalı, basit, bağlantılı ve ağırlıklı bir graf olmak üzere G'nin bitişiklik matrisi,

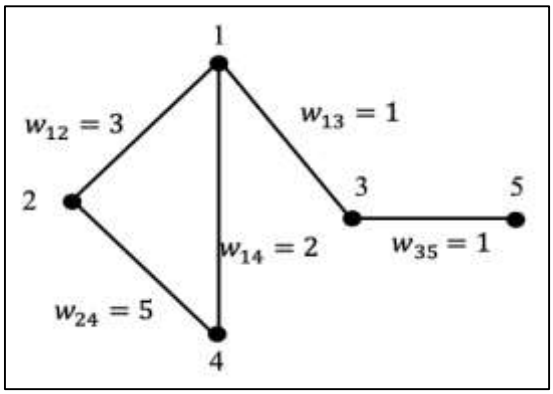
$$a_{ij} = \begin{cases} w_{ij}; & i \sim j \\ 0; & aksi\ durumlarda \end{cases}$$

olmak üzere,

$$A(G) = (a_{ij})_{nxn}$$

şeklinde tanımlanır.

Örnek



$$\mathrm{A}(G) = \begin{bmatrix} 0 & 3 & 1 & 2 & 0 \\ 3 & 0 & 0 & 5 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Şekil 2.16. G grafı ve ağırlıklı bitişiklik matrisi

2.6.2. Tanım

G, n noktalı, basit, bağlantılı ve matris ağırlıklı bir graf olsun, G'nin Laplacian matrisi,

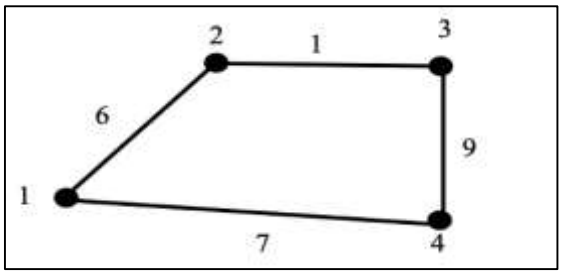
$$l_{ij} = \begin{cases} w_i; & i = j \\ w_{ij}; & i \sim j \\ 0; & aksi\ hal \end{cases}$$

olmak üzere,

$$L(G) = (l_{ij})_{nxn}$$

şeklinde tanımlanır.

Örnek



$$\mathrm{L}(G) = \begin{bmatrix} 13 & -6 & 0 & -7 \\ -6 & 7 & -1 & 0 \\ 0 & -1 & 10 & -9 \\ -7 & 0 & -9 & 16 \end{bmatrix}$$

Şekil 2.17. G grafı ve ağırlıklı Laplacian matrisi.

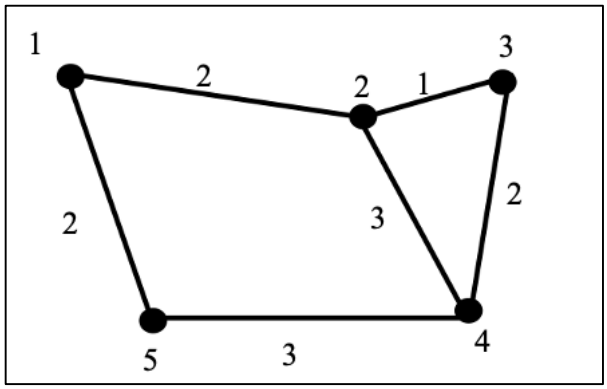
2.6.3. Tanım

G ağırlıklı grafının uzunluk matrisi,

$$\left[D_{e_w}\right]_{ij} = \begin{cases} w(P_{iJ}); & i \neq j \\ 0; & diğer\ durumlar \end{cases}$$

şeklinde tanımlanır.

Örnek



$$\left[D_{e_w}\right]_{ij} = \begin{bmatrix} 0 & 2 & 3 & 5 & 2 \\ 2 & 0 & 1 & 3 & 4 \\ 3 & 1 & 0 & 2 & 5 \\ 5 & 3 & 2 & 0 & 3 \\ 2 & 4 & 5 & 3 & 0 \end{bmatrix}$$

Şekil 2.18. G grafı ve ağıtlıklı grafın uzunluk matrisi.

3. GRAF ALGORİTMALARI

Üçüncü bölümde graf algoritmalarından Önce Derinlik Araması (DFS) ile Önce Genişlik Araması (BFS) ve çalışma modelleri açıklanmıştır.

3.1. Önce Derinlik Araması (DFS)

DFS, bir graf ya da ağaç veri yapısını keşfetmek veya geçmek için kullanılan bir graf geçiş algoritmasıdır. Genellikle kök olarak adlandırılan belirli bir noktadan başlar ve her kenarda geriye doğru izleme yapmadan önce mümkün olduğunca ileriye doğru keşfe devam eder [3].

DFS şu şekilde çalışır:

1. Bir Noktadan Başla: Grafta belirli bir başlangıç noktası seçilir.
2. Komşuları Keşfet: Mevcut noktasının bitişik, gidilmemiş bir noktasına gidilir.
3. Tekrarla: Gidilmemiş komşu kalmayana kadar yeni nokta için 2. Adım tekrarlanır.
4. Kontrol Et: Gidilmemiş komşu kalmadığında hâlâ gidilmemiş komşuları olan önceki noktalara geri izleme yapılır.
5. Tekrarla: Tüm noktalara gidilene kadar 2-4 arasındaki adımlar tekrarlanır.
6. DFS'nin bağlantısız bir graftaki tüm noktalara gitmeyebilir, bu nedenle tüm noktalara gidildiğinden emin olmak için tüm noktaların yinelenmesi gibi teknikler ile kullanılır. Başlangıç noktası seçimi ve graf temsili DFS'nin performansını etkileyebilir.

3.1.1. DFS uygulamaları

Yol Bulma: DFS, bir harita veya labirent içinde iki nokta arasındaki en kısa yolun bulunmasında kullanılır. Özellikle ızgara tabanlı oyunlarda veya harita yönlendirmesinde sıkça kullanılır [3].

Graf İşleme: DFS, bir graf içindeki tüm düğümleri veya kenarları ziyaret etmek için kullanılır. Örneğin, bir sosyal ağdaki ilişkileri incelemek veya web sayfalarını bağlamak için kullanılabilir.

Bağlantıları Bulma: DFS, bir ağdaki bağlantıları veya alt ağları bulmak için kullanılabilir. Örneğin, bir sosyal ağdaki arkadaşlık gruplarını veya internet üzerindeki web sitesi alt bağlantılarını bulabilirsiniz.

Sürücü İşleme: DFS, bir dosya sistemi içindeki dosyaları ve dizinleri gezme işlemlerinde kullanılır. Örneğin, bir bilgisayarın dosya sisteminde belirli türdeki dosyaları bulmak için DFS kullanılabilir.

Oyun Ağaçları: Oyunların yapay zekâ tarafından analiz edilmesi için DFS kullanılabilir. Örneğin, satranç gibi oyunlarda hamle seçeneklerini değerlendirmek için kullanılır.
Mantık ve Yapay Zekâ: DFS, mantıksal sorunları ve yapay zekâ algoritmalarını çözmek için kullanılabilir. Örneğin, bir bilgi tabanı sorgusunu çözmek veya bir durum uzayını keşfetmek için kullanılabilir.

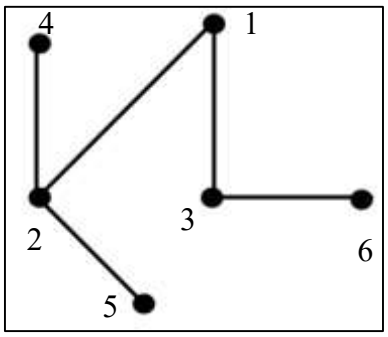
Graf Analizi: Büyük veri analizi sırasında, DFS, bir ağın yapısını ve özelliklerini anlamak için kullanılır. Örneğin, bir sosyal ağın ağ merkezlerini veya topluluklarını belirlemek için kullanılabilir.

Örnek

$G = (V, E)$ bir graf olmak üzere,

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (1,3), (2,4), (2,5), (3,6)\}$



Şekil 3.1. G grafı

Nokta 1'den başlayın.

Nokta 2 olan bitişik bir noktaya gidin.

Nokta 1'e bağlı gidilmemiş başka nokta olmadığı için nokta 2'nin komşusu olan nokta 4'e devam edin.

Nokta 4'ün gidilmemiş bitişik noktası olmadığı için 2. noktaya geri dönün.

Nokta 2'den nokta 5'e gidin.

Nokta 5'in gidilmemiş komşuları olmadığı için nokta 2'ye ve ardından nokta 1'e geri dönün.

Nokta 1'den nokta 3'e gidin.

Son olarak, nokta 3'ten nokta 6'ya gidin.

3.2. Önce Genişlik Araması (BFS)

BFS, bir graf ya da ağaç veri yapısını keşfetmek veya geçmek için kullanılan başka bir graf geçiş algoritmasıdır. BFS genellikle bir kuyruk veri yapısı kullanılarak uygulanır. Bu, noktalara keşfedildikleri sırayla gidilmesini sağlar [4].

BFS şu şekilde çalışır:

1. Başlangıç Noktası Seç: Grafta bir başlangıç noktası seçilir.
2. Komşu Noktaları Keşfet: Mevcut noktanın tüm bitişik, gidilmemiş noktalarına gidilir.
3. Tekrarla: Yeni gidilen noktaların her biri için bu işleme devam edilir.
4. Sonraki Seviyeye Geç: Geçerli seviyedeki tüm noktalara gidildikten sonra bir sonraki nokta seviyesine geçilir.

3.2.1. BFS Uygulamaları

BFS'nin aşağıdakiler dahil çeşitli uygulamaları vardır:

En Kısa Yol Bulma: Tüm kenarlar aynı ağırlığa sahip olduğunda, BFS, bir başlangıç noktasından diğer tüm noktalara en kısa yolu bulmak için kullanılabilir.

Labirent Çözme: Bir labirentte girişten çıkışa kadar bir yol olup olmadığının belirlenmesi.

Bağlantılı Bileşenler: Yönsüz bir grafta birbirine bağlanan nokta gruplarının tanımlanması.

Ağ Yayını: Tüm noktalara ulaşmak için mesajların bir ağda yayınlanması.

Sosyal Ağ Analizi: Sosyal ağlardaki yapı ve bağlantıların incelenmesi.

Örnek

$G = (V, E)$ bir graf olmak üzere,

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (1,3), (2,4), (2,5), (3,6)\}$

Başlangıç noktası: 1

Başlangıç noktasını işaretle ve bir kuyruğa ekle: [1]

Kuyruğun başından noktayı çıkar (1) ve komşularını incelemeye başla.

Komşular: 2 ve 3

Henüz gidilmemiş olan 2 ve 3'e git ve kuyruğa ekle: [2, 3]

Kuyruğun başından noktayı çıkar (2) ve komşularını incelemeye başla.

Komşular: 4 ve 5

Henüz gidilmemiş olan 4 ve 5'e git ve kuyruğa ekle: [3, 4, 5]

Kuyruğun başından noktayı çıkar (3) ve komşularını incelemeye başla.

Komşu: 6

Henüz gidilmemiş olan 6'ya git ve kuyruğa ekle: [4, 5, 6]

Kuyruğun başından noktayı çıkar (4) ve komşularını incelemeye başla.

Hiç yeni komşusu yok.

Kuyruğun başından noktayı çıkar (5) ve komşularını incelemeye başla.

Hiç yeni komşusu yok.

Kuyruğun başından düğümü çıkar (6) ve komşuları incelemeye başla.

Hiç yeni komşusu yok.

Tüm noktalara gidildiği için BFS algoritması sona erdi.

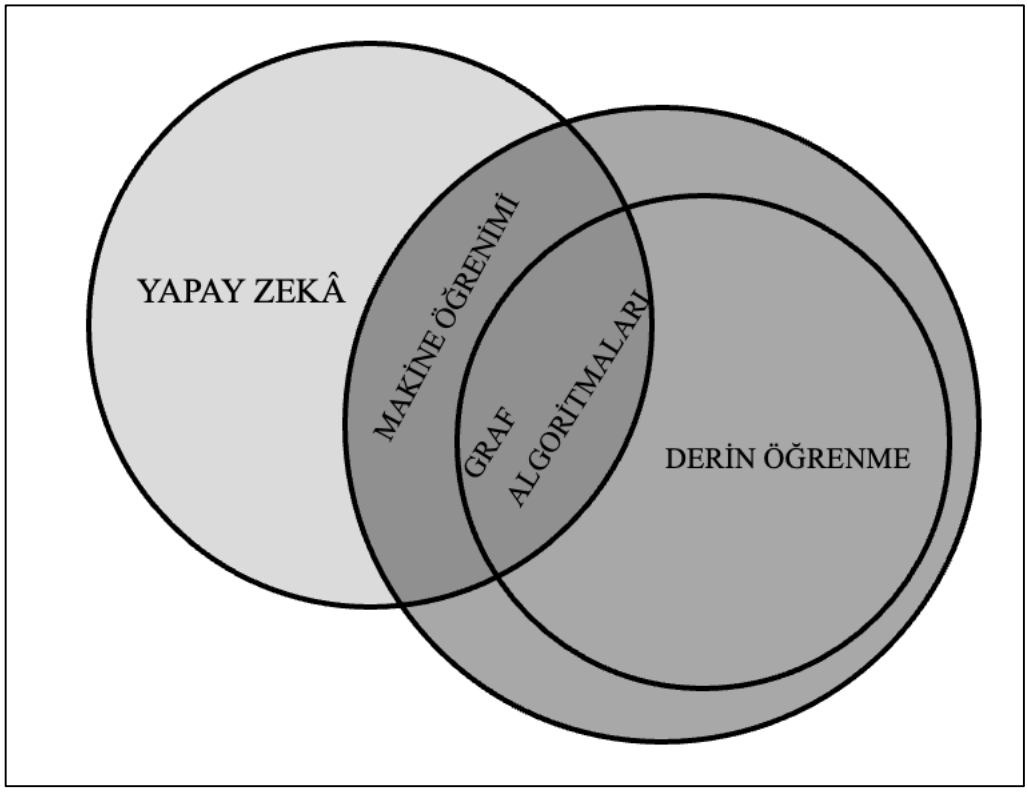
Noktaların BFS sırası: [1, 2, 3, 4, 5, 6]

4. GRAF MAKİNE ÖĞRENİMİ

Makine öğrenimi algoritmaları, görüntülerin tanınmasına, doğal dilin işlenmesine ve konumların tespit edilmesine yardımcı olarak modern uygulamalarda yaygın olarak kullanılmaktadır [5]. Destekleyici kütüphanelere ve araçların yanı sıra bu algoritmalar çeşitli projelerin başarıyla tamamlanmasını sağlar.

Graf makine öğrenimi (GML), veri yapılarını graf olarak temsil eden ve bu yapıları analiz etmek için makine öğrenimi tekniklerini kullanan bir alandır [5]. Geleneksel veri setlerinin yanı sıra, çeşitli ilişkisel veri kümeleri, sosyal ağlar, ağ bağlantıları, öneri sistemleri, biyoinformatik ve kimyasal bileşikler gibi alanlardaki graf yapılı verileri kapsar.

Graf makine öğrenimi, graf veri yapılarını modelleme, tahmin ve sınıflandırma gibi çeşitli görevlerde kullanmayı amaçlar. Graflar kullanılarak çözülebilecek çeşitli problemler vardır. Bu problemler ve çözümler problemden probleme ve veri setinden veri setine değişir.



Şekil 4.1. Yapay zekâ-makine öğrenimi Venn şeması.

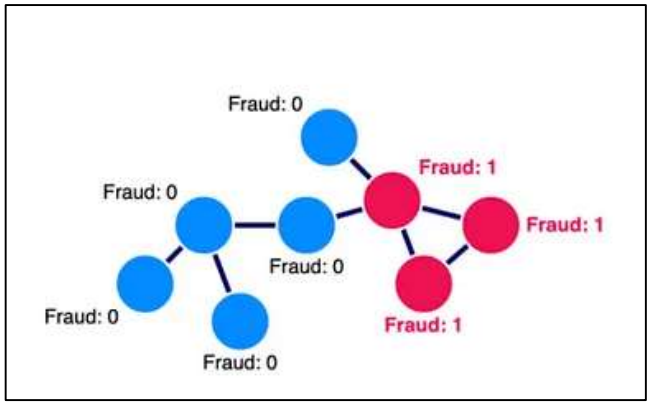
4.1. Denetimli Graf Makine Öğrenimi

Denetimli öğrenme algoritmaları, çıkış verileri bilindiğinde doğrusal regresyon,[1] lojistik regresyon ve destek vektör makineleri yaygın olarak kullanılır. Bu denetimli öğrenme tekniklerinin geçmiş verilere dayalı olarak konumların tahmin edilmesinde başarılı olduğunu kanıtlamaktadır. [5].

Denetimli GML, bir noktanın özelliğini tahmin ederek bir finans platformundaki bir hesabın dolandırıcılık olarak sınıflandırılması gerekip gerekmediğine veya bir çevrimiçi satış mağazasında ürün veya ürünlerin nasıl sınıflandırılacağına karar verir.

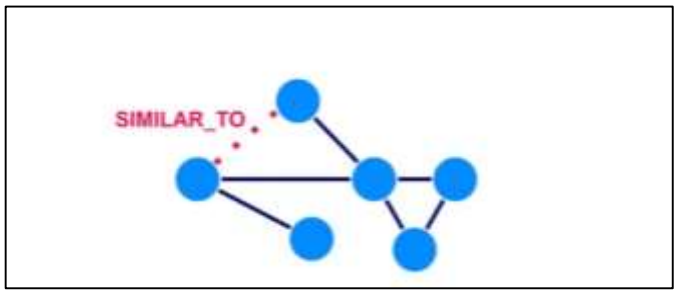
Bağlantı tahmini ile iki nokta arasında bir ilişkinin var olup olmayacağının ve ilişkiyle ilgili potansiyel olarak bazı özellikleri; bir kullanıcının bir sonraki adımda ne satın almak veya neyle etkileşimde bulunmak isteyeceğini tahmin etmek istediğimiz öneri sistemleri sunar.

Graf özelliği tahmini ile bir grafın veya alt grafın ayrık veya sürekli özelliğini tahmin eder. Graf özelliği tahmini, varlıkları tam bir veri kümesini temsil eden daha büyük bir graf içindeki noktaları modellemek yerine, tahmin için her bir varlığı ayrı bir graf olarak modellemek istediğiniz alanlarda kullanışlıdır. Kullanım örnekleri arasında malzeme bilimleri, biyoenformatik ve ilaç keşfi yer alır. [6].

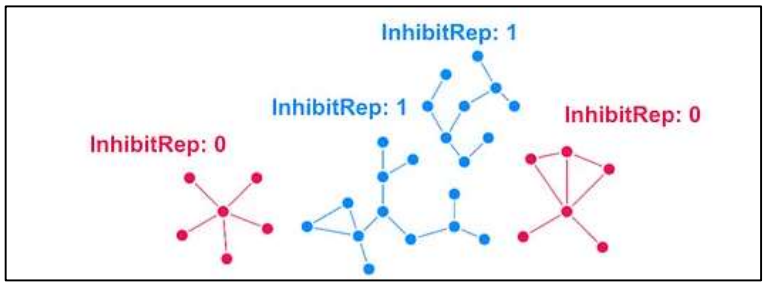


Şekil 4.2. Nokta özelliği tahmini

[1] Gerileme.



Şekil 4.3. Bağlantı tahmini



Şekil 4.4. Graf özelliği tahmini

4.2. Denetimsiz Graf Makine Öğrenimi

Denetimsiz öğrenme algoritmaları, k-ortalamalı kümeleme ve hiyerarşik kümeleme, çıktı verileri bilinmediğinde ya da etiketlenmediğinde kullanışlıdır. Bu algoritmalar kullanıcıların geçmiş verilerine dayalı olarak farklı konum kategorilerine ayrılmasına yardımcı olur. [6]

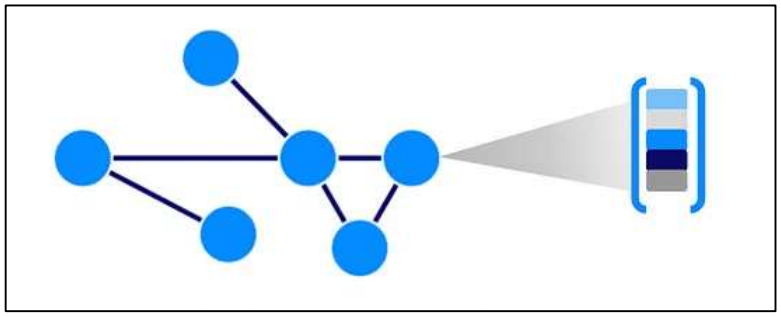
Denetimsiz GML; temsil öğrenimi, topluluk belirme, benzerlik ve yol bulma olmak üzere oldukça yaygın olan dört yönteme sahiptir.

Temsil öğrenimi, GML uygulamaları için ana temadır. Graf yapılarından düşük boyutlu özellikler üretir. Topluluk belirme, bir graf içinde birbirine bağlı nokta gruplarını tanımlamak için kullanılan bir terimdir.

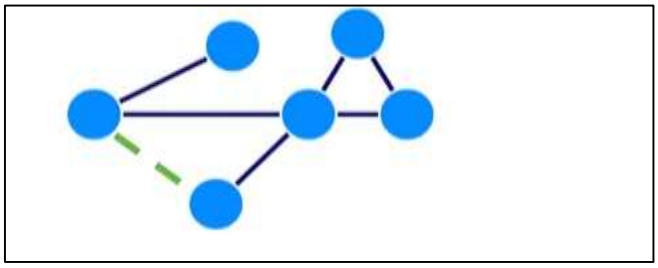
Topluluk belirleme yöntemi ile dolandırıcılık, anormal hareket tespiti, sosyal ağ analiz etmek gibi oldukça çeşitli ve pratik uygulamaları vardır.

Benzerlik; bir graftaki benzer nokta çiftlerini bulmayı ve ölçmeyi ifade eder. Bu yöntem, öneri, sahtekarlık ve anormallik gibi kullanım senaryolarına kolaylıkla uygulanır.

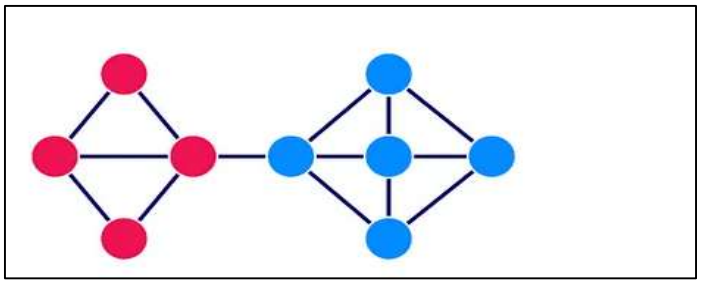
Yol bulmak yöntemi için öncelikle bir graftaki en önemli veya etkili merkez noktayı tespit etmek gereklidir. Merkez nokta tespiti ile bir graftaki en düşük maliyetli, en kullanışlı ya da en kısa yollar seçenek olarak elde edilir.



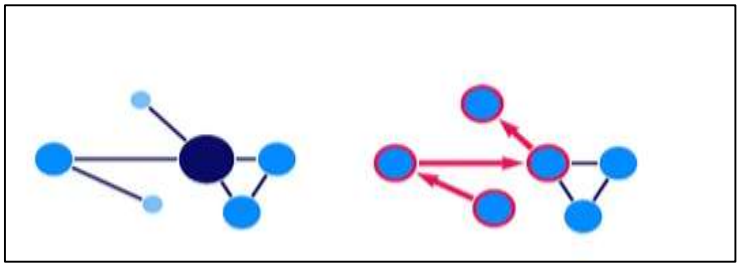
Şekil 4.5. Temsil öğrenme



Şekil 4.6. Benzerlik



Şekil 4.7. Topluluk belirleme



Şekil 4.8. Merkez nokta belirleme

4.3. Sıkıştırma ve Gömme

GML'nin en önemli hedeflerinden biri, tahmin ve çıkarım için önemli sinyalleri korurken, büyük ve seyrek veri yapılarını sıkıştırmaktır.[6] GML, genellikle karmaşık ağ yapılarını analiz eder. Bu ağ yapıları, büyük miktarda veri ve genellikle seyreklik içerir. GML algoritmaları, bu büyük ve karmaşık ağları daha iyi anlamak, özelliklerini çıkarmak ve daha az miktarda bilgi kaybıyla sıkıştırmak için kullanılır. Bu sıkıştırmayı gerçekleştirmek için üç GML yaklaşımı kullanır. Bunlardan ilki klasik graf algoritmalarıdır. Bu algoritmalar, denetimsiz topluluk tespiti, benzerlik analizi, merkeziyetçilik veya yol bulma gibi pek çok görevde kullanılabilirler. Klasik grafik algoritmaları genellikle basittir, başlangıcı kolaydır ve açıklanabilirdir. Ancak, diğer GML yaklaşımlarına göre daha fazla manuel çalışma ve konu uzmanlığı gerektirirler. Özellikle karmaşık yapı analizleri için bir başka yaklaşım tercih edilebilir. İkinci yaklaşım olan graf gömme, graf noktalarını sayısal vektörlerle temsil etmeyi sağlayan bir tekniktir. Bu teknik, bir noktanın komşuları, ilişkileri ve genel graf yapısını vektörlerle ifade eder. Noktaları benzer noktalarla ilişkilendirecek şekilde vektörler oluşturarak graf yapısındaki benzerlikleri korumayı amaçlar. Bu yaklaşım, küçük ölçekli graflardan büyük ölçekli gerçek dünya ağlarına kadar geniş bir yelpazedeki grafları temsil edebilir. Üçüncü yaklaşım olan GNN modeli, graf verilerini girdi olarak alır ve bu verileri ara katmanlara dönüştürür. Bu katmanlar daha sonra bir tahmin görevine hizalanmış son katmana beslenir. Bu tahmin görevi; denetlenmiş veya denetlenmemiş olabilir. Denetlenmiş görevler nokta özelliklerini tahmin etmek veya bağlantıları tahmin etmek gibi belirli tahminleme görevlerini içereir. Denetlenmemiş görevler daha genel amaçlı kümeleme veya temsili öğrenmeyi hedefleyebilir.

GNN'ler, geleneksel algoritmalardan ve GNN’siz gömme farklı olarak, tamamen uçtan uca graf odaklı çözümler sunarlar. Graf verilerinin analizini baştan sona yapabilen bir modeldir.

5. GRAF TABANLI DERİN ÖĞRENME

Graf tabanlı derin öğrenme, karmaşık veri yapılarını modellemek ve anlamak için derin öğrenme tekniklerinin graf teorisi ile birleştirilmesi sürecidir [7]. Graf yapıları; noktalar ve kenarlar aracılığıyla karmaşık ilişkisel verileri temsil eder. Bu yapılar, veri ağlarından temel bir rol oynar. Yönlü ve yönsüz, ağırlıklı ve ağırlıksız olabilen bu graf yapıları çeşitli veri türlerini ve ilişkisel bağlantıları yansıtır [7]. Özellikle ilişkisel ve yapısal verileri işlemek için son derece etkili ve yenilikçi bir yaklaşım sunar. İletişim ağlarında kullanılan tipik graf yapıları, noktalar ve kenarlar ile tanımlanır. Bu noktalar ve kenarlar, statik veya dinamik özelliklerle ilişkilendirilebilir.

Ek olarak, graf tabanlı derin öğrenme, modern veri analizi ve yapay zekâ araştırmalarında önemli bir role sahiptir. Bu alandaki gelişmeleri, karmaşık ve ilişkisel veri yapılarını anlamamızı ve işlememizi büyük ölçüde geliştirir ve yeni uygulamalar ile teknolojik gelişmelerin yolunu açar. Gelecekte, bu teknolojilerin daha da geliştirilmesi ve geniş bir yelpazede uygulanması mümkündür.

Graf tabanlı derin öğrenme modelleri, özellikle Graf Sinir Ağları (GNN), hızla gelişen bir araştırma alanıdır. Bu modeller, noktalar arası bilgi alışverişine dayalı bir mesaj geçiş mekanizması kullanarak çalışır. Çeşitli GNN varyantları, örneğin Graf Konvolüsyon Ağları (GCN) bu alandaki yenilikler arasında yer alır.

5.1. Graf Sinir Ağları (GNN)

Graf Sinir Ağları (GNN'ler), graf veri yapılarında öğrenme işlemlerini gerçekleştirmek için kullanılan derin öğrenme modelleridir [8]. GNN'lerin temelinde, noktaların birbirleriyle nasıl etkileşimde bulunduğunu modelleyen matematiksel formüller vardır. Bu açıdan GNN'ler matematiksel olarak graf teorisinin temel kavramlarını ve derin öğrenme tekniklerini birleştirir. GNN'lerin temel işleyişi mesaj aktarma ve durum güncelleme üzerine kuruludur.

5.1.1. Tanım

$G = (V, E)$ bir graf, V noktalar ve E kenarlar kümesi olmak üzere, her nokta v ve her kenar e için x_v bir noktanın, x_{uv} ise bir kenarın özellik vektörü olarak tanımlanır.

5.1.2. Tanım

Noktalar, komşu kenarlardan özellik vektörleri toplar ve toplanan bilgileri kendi bilgileriyle birleştirerek yeni bir vektör oluşturur.

$G = (V, E, X)$ bir graf ve H, yeni vektörlerin oluşturduğu bir gömme matris olmak üzere; A, tek bir mesaj geçiş katmanını temsil eden bitişiklik matrisi, X özellik matrisi ve W ağırlık matrisini temsil eder.

$$\mathrm{H} = \widetilde{\mathrm{A}}^{\mathrm{T}}\mathrm{X}\mathrm{W}^{\mathrm{T}}$$

şeklinde ifade edilir.

5.1.3. Tanım

$m_u^{(l)} = MSG^l(h_u^{(l-1)})$ fonksiyonu mesaj fonksiyonu olarak tanımlansın.

Bir mesaj hesaplaması, bir önceki katmandaki u noktasının temsilini alır ($h_u^{(l-1)}$) ve mesaj bilgisine ($m_u^{(l)}$) dönüştürür. Böylece her nokta için bir sonraki katmandaki diğer noktalara gönderecek bir mesaj oluşturur.

Örnek

$\mathrm{W}^{(\mathrm{l})}$, l katmanındaki bir ağırlık matrisi olmak üzere

$(m_u^{(l)}) = \mathrm{W}^{(\mathrm{l})} h_u^{(l-1)}$, W ile gelen dönüştürülmüş mesaj.

5.1.3. Tanım

$h_v^{(l)} = AGG^l(m_u^{(l-1)})$ mesaj toplama fonksiyonu olmak üzere,

$m_u^{(l-1)}$ mesajlarını tek bir mesajda toplayarak sıkıştırır.

5.1.4. Tanım

Gömme birden fazla katmana sahip olmak üzere,

Katman 0: x

Katman 1: $h^{(1)}$, birinci katmandaki komşulardan bilgi toplamı

Katman K: $h^{(l)}$, l katmandaki komşulardan bilgi toplamı

σ: aktivasyon fonksiyonu olmak üzere,

Bir k katmanındaki komşu gömmelerin ortalaması aşağıdaki şekilde hesaplanır.

$h_i{}^{(k+1)} = \sigma(h_i{}^{(k+1)} W^k + \sum_{j \epsilon N_i} h_j{}^{(k)} W^k)$

5.1.1. Graf konvolüsyonel ağları (GCN)

Graf Konvolüsyonel Ağlar (GCN), Graf Sinir Ağlarının (GNN) bir türüdür ve graf yapıları üzerinde derin öğrenme yapmak için kullanılır [9]. GCN'ler, veri yapılarını graf olarak modelleyerek, bu graf yapısındaki noktaların özelliklerini etkili bir şekilde öğrenmek ve işlemek için tasarlanmıştır. Bu, özellikle noktalar arası ilişkiler ve yapının bütünü için önemlidir.

GCN'nin temel işleyiş mantığı şu şekildedir [9]:

GCN, noktalar ve bu noktaları birbirine bağlayan kenarlar ile temsil edilen bir graf yapısını temel alır. Noktalar genellikle elemanlar, kenarlar ise bu elemanlar arasındaki ilişkileri temsil eder.

GCN, konvolüsyon işlemi gibi birçok katman içerir. Bu katmanlar her noktanın komşuları ile olan ilişkilerini öğrenmek ve noktaları güncellemek için kullanılır. Noktaların özellikleri, komşularının özelliklerini kullanarak güncellenir.

GCN, noktaları temsil etmek için vektörler veya özellik matrisleri kullanır. Her noktanın başlangıçta belirli bir özellik vektörü vardır. GCN, bu özellikleri güncellemek için konvolüsyon işlemlerini kullanır.

GCN, katmanları yığılabilir, daha net bir ifade ile birçok GCN katmanı birbirine bağlanabilir. Bu, daha derin ve karmaşık graf yapısı analizlerinin yapılabilmesine olanak tanır.

GCN, graf verilerini kullanarak öğrenme yapar. Bu öğrenme, noktaları sınıflandırmak, nokta gömme oluşturmak veya graf üzerinde çeşitli görevler gerçekleştirmek için kullanılabilir.

Kısaca ifade etmek gerekirse doğrudan bir graf üzerinde çalışır, çok katmanlı bir sinir ağıdır ve noktalarını komşuluk özelliklerine dayalı olarak gömme vektörleri oluşturur. Model, bir konvolüsyon katmanıyla yakın komşu bilgilerini yakalar ve birden fazla katman üst üste konduğunda daha geniş komşuluklar hakkında bilgi entegre eder.

5.1.1.1. Tanım

$G = (V, E)$ bir graf, V noktalar ve E kenarlar kümesi olmak üzere, A, grafın bitişiklik matrisidir, nokta i ve nokta j için A_{ij} i ve j arasında kenar olup olmadığı gösterir.

$A_{ij} = 1$ ise kenar var, $A_{ij} = 0$ ise iki nokta arasında kenar yoktur.

5.1.1.2. Tanım

$H^{(l)}$, l katmanındaki noktaların özelliklerini içeren bir matris,

$$\hat{A} = A + I$$

şeklinde ifade edilebilen bir simetrik matris, $\widehat{D}$, derece matrisi ve W^l, l katmanlı bir ağırlık matrisi olmak üzere GCN katmanları şu şekilde ifade edilir:

$$H^{(l)} = \sigma(\widehat{D}^{\frac{-1}{2}}\hat{A}\widehat{D}^{\frac{-1}{2}}H^{(l)}W^l)$$

5.1.1.3. Tanım

$h^{(l)}$, l katmandaki komşulardan bilgi toplamı, W, k katmanlı ağırlık matrisi ve σ aktivasyon fonksiyonu olmak üzere şu şekilde hesaplanır: [5]

$$h_v{}^{(l)} = \sigma(W^k \sum_{j \epsilon N_v} \frac{h_j{}^{(k-1)}}{|N(v)|})$$

5.1.1.4. Tanım

ReLU bir aktivasyon fonksiyon olmak üzere,

$$\mathrm{ReLU(x)} = \max(0, \mathrm{x}) = \begin{cases} 0, & \mathrm{x} < 0 \\ \mathrm{x}, & \mathrm{x} > 0 \end{cases}$$

şeklinde ifade edilir.

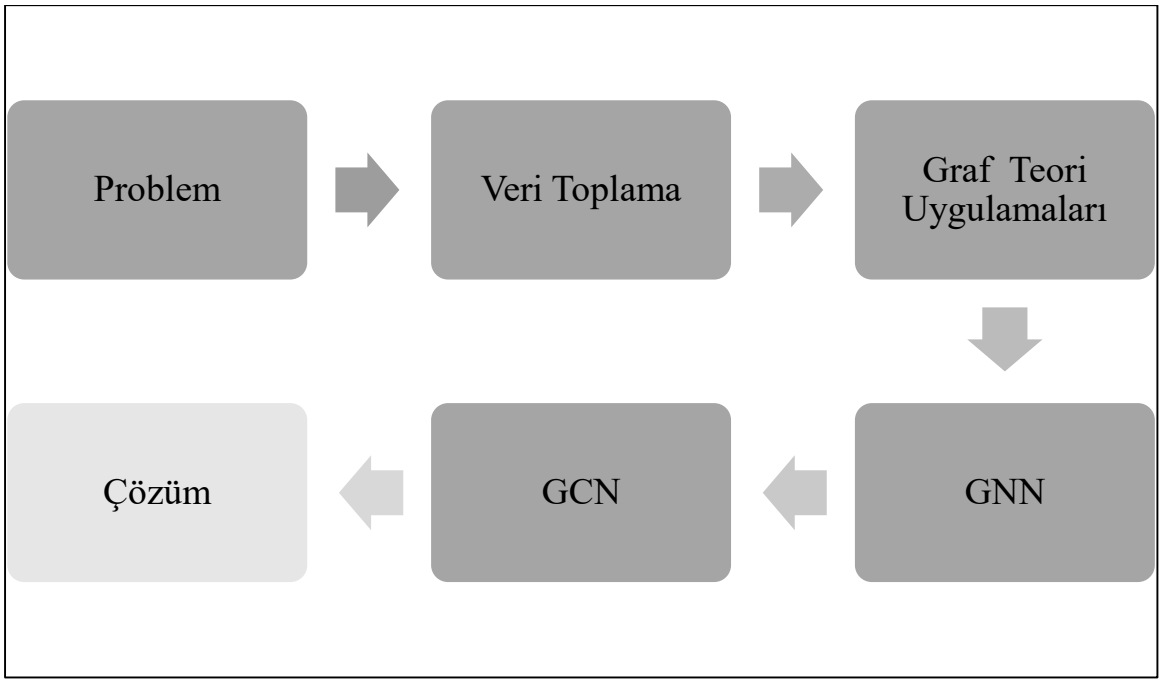
Derin öğrenme mimarilerinde sıklıkla tercih edilen bu fonksiyon, negatif değerleri 0'a dönüştürerek pozitif değerleri değiştirmeden bırakır.

5.1.2. GCN uygulamaları

1. Sosyal ağlardaki ilişkileri modellemek ve noktaları sınıflandırmak için kullanılır. Örneğin, kullanıcıların davranışlarını analiz etmek veya sosyal ağlarda dolandırıcılık algılama gibi görevleri.
2. E-ticaret sitelerinde veya içerik platformlarında, kullanıcılara özel öneriler yapmak için kullanılır.
3. Protein-protein etkileşim ağları, metabolik ağlar gibi biyolojik ağların analizi için kullanılır.
4. Kimyasal Molekül Yapıları: Kimyasal moleküllerin yapısını ve özelliklerini analiz etmek için kullanılır. Örneğin, ilaç tasarlama veya kimyasal bileşiklerin özelliklerinin tahmini gibi alanlar.
5. Metin verilerini temsil etmek için GCN kullanılır.
6. Link tahmini, nokta sınıflandırma, nokta gömme gibi graf tabanlı görevler için kullanılır.

6. ANALİZ

Bu bölüm, GCN modelinin olası bir İstanbul depremi senaryosunda kurtarma ekibi rotalarını oluşturmak için nasıl kullanıldığını, modelin performansını ve elde edilen sonuçları açıklar.



Şekil 6.1. Analiz Şeması

6.1. Veri Seti

Türkiye'de beklenen Marmara depremini temsil eden veriler toplandı ve Şişli bölgesindeki toplanma alanlarını içeren veri seti hazırlandı. Bu veri seti için gerekli olan bilgilere E-Devlet uygulamasının Afet ve Acil Durum Toplanma Alanı Sorgulama seçeneği üzerinden ulaşıldı. Veri setinde toplam 23 adet toplanma alanı kullanıldı.

6.2. Araçlar ve Kütüphaneler

Toplanan verilerin graf yapısına dönüştürülmesi, noktalar arasındaki bağlantıların oluşturulması ve GCN modeli için uygun formata getirilmesi için aşağıdaki araç ve kütüphaneler kullanıldı [10].

Python: Bir programlama dilidir. Basit ve anlaşılması kolay yapısı, geniş kütüphane desteği ve çok yönlülüğü nedeniyle pek çok alanda kullanılır. Veri analizi, yapay zekâ, bilimsel hesaplamalar, web geliştirme, sistem otomasyonu ve oyun geliştirme gibi birçok alanda tercih edilir.

PyTorch: Yapay zekâ ve derin öğrenme uygulamaları geliştirmek için kullanılan açık kaynaklı bir makine öğrenimi kütüphanesidir.

TensorFlow: Google tarafından geliştirilen açık kaynaklı bir makine öğrenimi ve derin öğrenme kütüphanesidir. Derin sinir ağı modelleri ve makine öğrenimi uygulamaları geliştirmek için kullanılır.

NetworkX: Python programlama dili için bir graf analizi kütüphanesidir. Ağları, grafları ve karmaşık ağ yapılarını oluşturmak, analiz etmek ve görselleştirmek için kullanılır.

NumPy:, Python'da bilimsel hesaplama için kullanılan temel bir kütüphanedir. Bu kütüphane, çok boyutlu diziler ve matrisler üzerinde yüksek performanslı matematiksel işlemler yapmak için kullanılır.

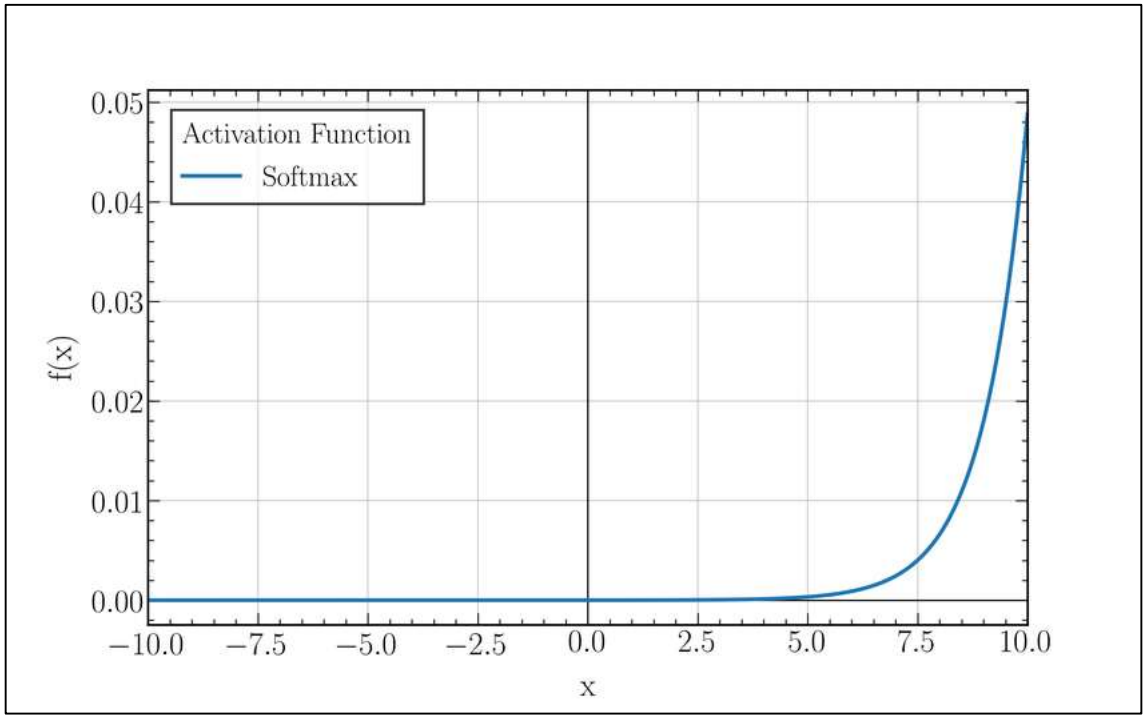
Matplotlib: Python'da graf oluşturma için kullanılan güçlü bir çizim kütüphanesidir. Bu kütüphane, graf görselleştirmeler oluşturmak için kullanılabilir ve birçok farklı graf türüyle uyumludur.

Scikit-learn: Python'da kullanılan açık kaynaklı bir makine öğrenimi kütüphanesidir. Bu kütüphane; sınıflandırma, regresyon, kümeleme, boyut azaltma, model seçimi ve ön işleme gibi birçok makine öğrenimi algoritmasını ve araçlarını içerir.

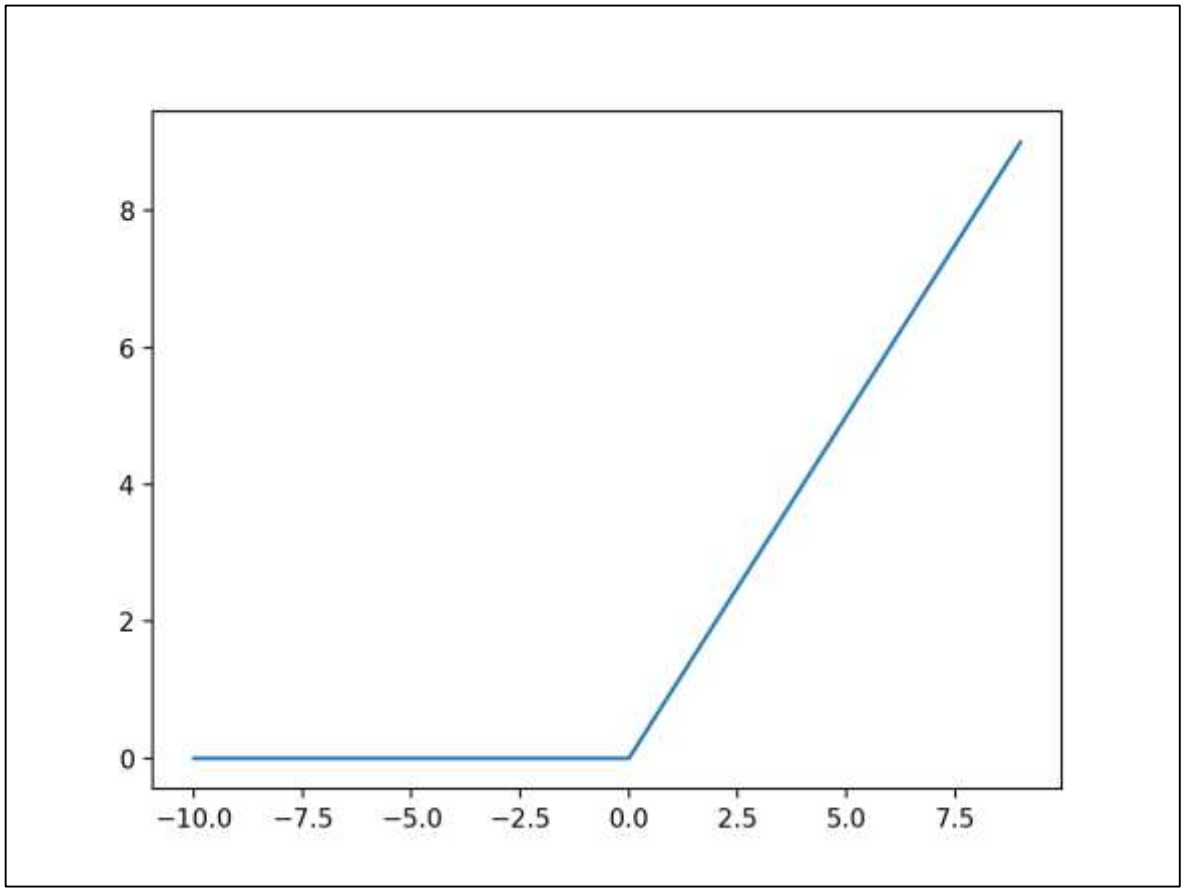
6.3. GCN Modeli

Modeli çalıştıracak kod tasarlandı. Kullanıcı modele, başlangıç noktası olarak arama kurtarma ekibinin anlık konum bilgisinin enlem ve boylam bilgisini girdi. Kod hazırlanan veri seti gömüldü. Veri setindeki konum bilgileri noktalara dönüştürüldü. Bu noktalar başlangıç noktasına uygun olarak tasarlanan kod tarafından hizalandı ve noktalar arası mesafeler hesaplandı. NetworkX, Python’a eklendi. NetworkX, kod tarafından hesaplanan

noktalar arası mesafeyi graf haline dönüştürdü. Python'a, Pytorch eklendi. PyTorch kullanarak iki katmanlı GCN modeli tanımlandı. Bu iki katmanını birbirine bağlamak için Aktivasyon fonksiyonu ReLu kullanıldı. Bitişiklik matrisi ve özellik matrisi de modele eklendi ve modelin parametreleri tanımlandı. Tanımlanan bilgilerle model oluşturuldu.

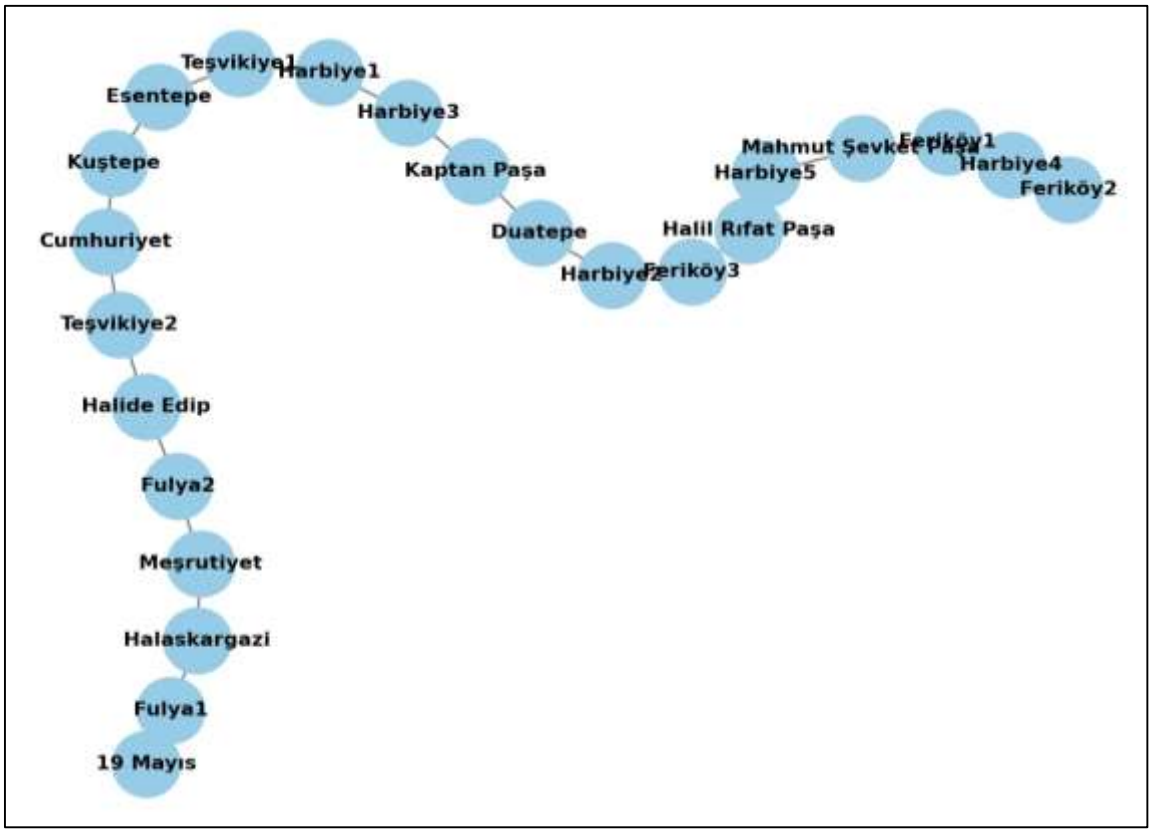


Şekil 6.2. Softmaks fonksiyonu.



Şekil 6.3. ReLu fonksiyonu.

Python'a TensorFlow eklendi. Noktalar, modelde tanımlanan bitişiklik matrisi ve özellik matrisinde kullanılacak tensor formatına dönüştürüldü. Bitişiklik matrisi iki boyutlu tensora çevrildi. GCN modelinin eğitilme aşamasına geçildi. Model en iyi 1000 tur çalıştırıldı. Her tur sonunda güncellenen ağırlıklar, optimizasyon ve kayıp fonksiyonu kullanılarak en uygun ağırlık değerine [0,1] ulaşıldı. Elde edilen ağırlık değerleri aktivasyon fonksiyonu Softmax tarafından iyileştirildi. Son aşamada, toplanma alanları ilk olarak matris şeklinde hizalanarak teriminale basıldı. NetworkX ve matlibplot birlikte kullanılarak model görselleştirildi.



Şekil 6.4. Optimal graf

7. SONUÇ VE ÖNERİLER

Genel olarak uygulama, belirli bir konumdan başlayarak çeşitli buluşma noktalarına giden bir kurtarma ekibi için en uygun yolları tahmin etmek üzere bir GCN kullanmaktadır. Graf yapısından örüntüleri öğrenmek için makine öğrenimi uygulanmış ve çok katmanlı GCN modeli aracılığıyla derin öğrenme kullanılmıştır. Yapay zekâ yönü, modelin öğrenilen örüntülere dayalı tahminler yapma becerisinde belirgindir. Bitişiklik matrisi ve özellik matrisi, sırasıyla graf yapısını ve düğüm özelliklerini temsil etmede önemli rol oynar.

Modelin gerçek hayata uygulanması gelecek zamanlar için düşünülebilir. Model, toplanma noktalarının zaman içerisinde değişiklerine uyum sağlaması açısından uygun veriyi sağlayacak bir sisteme entegre edilebilir. Canlı konum takibi, dinamik grafikler, trafik koşulları gibi hiper parametrelerin eklenmesi GCN modelinin gelişmesine katkı sağlar. Geliştirilen bu model bir mobil uygulama aracılığı ile uygun birimlerin kullanıma sunulabilir.

KAYNAKLAR

1. İnternet: Carlson, S. C. (2024). Graph Theory. Encyclopedia Britannica URL: https://www.britannica.com/topic/graph-theory Son Erişim Tarihi: 26.01.2024

2. Büyükköse, Ş., Kaya Gök, G., Özkan K, G. ve Eren, S. (2021). *Graf Teori* Ankara: Nobel, 32-65.

3. Khuller, S., and Raghavachari, B. (2010). *In Algorithms and theory of computation handbook: general concepts and techniques.* (Second Edition). Florida: CRC Press.7-1.

4. İnternet: Apostolico, A., and Drovandi, G. (2009). Graph compression by BFS Algorithms. *Algorithms*, 2(3), 1031-1044. URL: https://www.researchgate.net/publication/26850077_Graph_Compression_by_BFS Son Erişim Tarihi: 12.12.2023

5. İnternet: Stanford Online. (2021). Stanford CS224W: Machine Learning with Graphs 2021 Lecture 3.1 - Node Embeddings Video. Youtube. Son Erişim Tarihi: 10.12.2023

6. İnternet: Blumenfeld, Z. (2023). Graph Machine Learning: An Overview. URL: https://towardsdatascience.com/graph-machine-learning-an-overview-c996e53fab90 Son Erişim Tarihi: 13.12.2023

7. Heidari, N., Hedegaard, L. and Iosifidis, A. (2022). *Deep Learning for Robot Perception and Cognition,* London: Elsevier, 71-99.

8. İnternet: Hong, S. (2020). An Introduction to Graph Neural Network (GNN) For Analysing Structured Data. URL: https://towardsdatascience.com/an-introduction-to-graph-neural-network-gnn-for-analysing-structured-data-afce79f4cfdc Son Erişim Tarihi: 12.12.2023

9. İnternet: Kipf, T. (2016). Graph Convolutional Networks. URL: https://tkipf.github.io/graph-convolutional-networks/ Son Erişim Tarihi: 12.12.2023

10. Van Rossum, G., and Drake, F. (2011). *An İntroduction to Python.* Bristol: Network Theory Limited, 115.

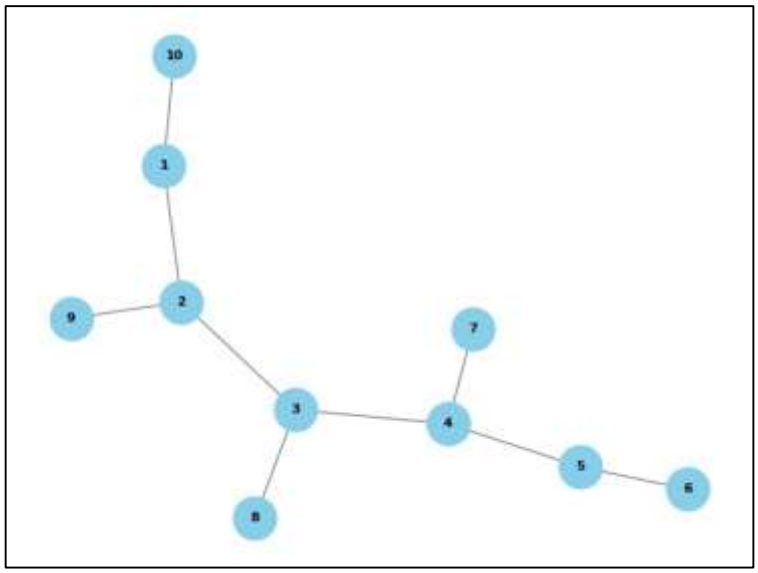
.

EKLER

EK-1. DFS uygulaması



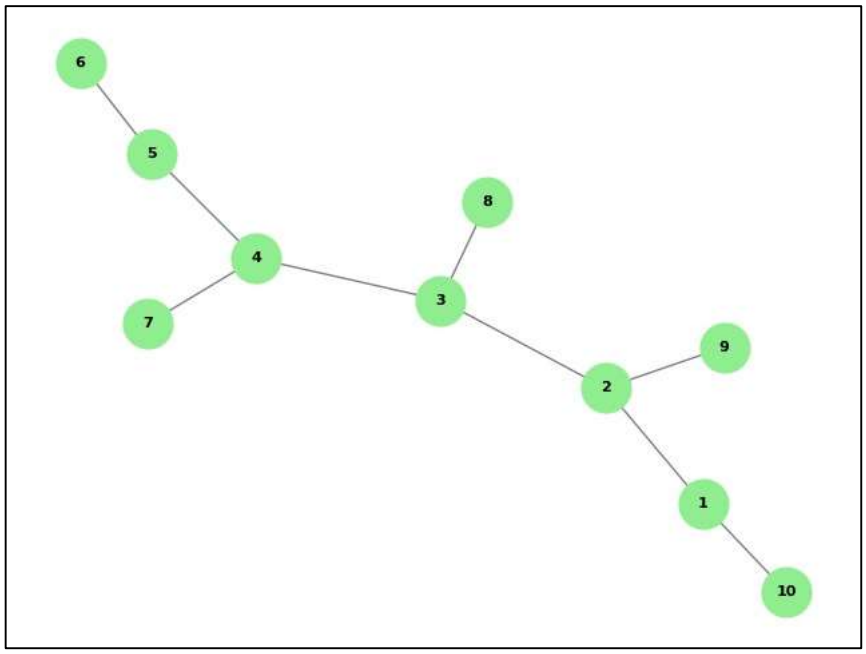
EK-2 DFS uygulaması graf sonucu



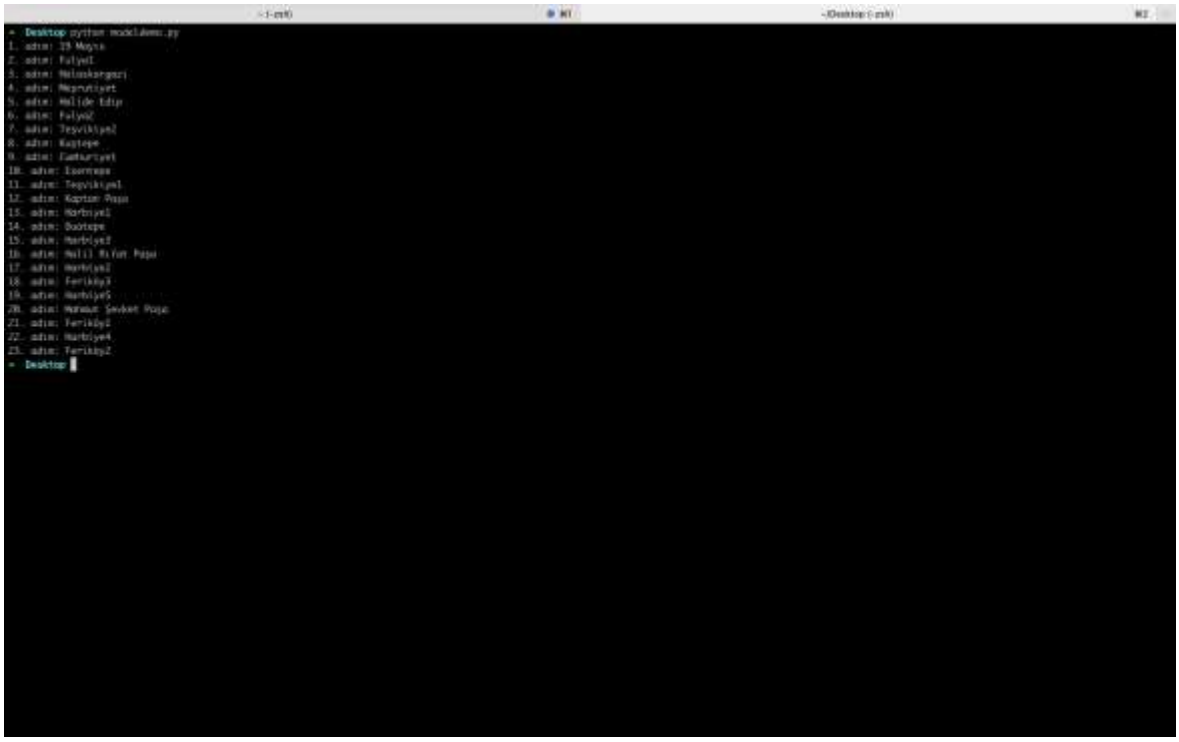
EK-3. BFS uygulaması



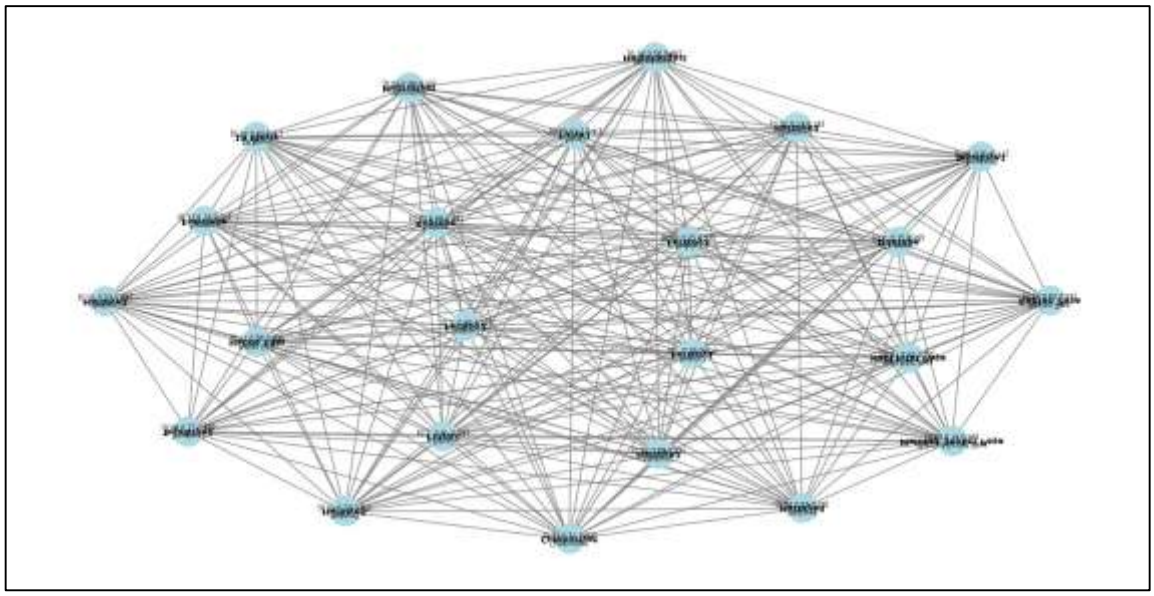
EK-4. BFS uygulaması graf sonucu



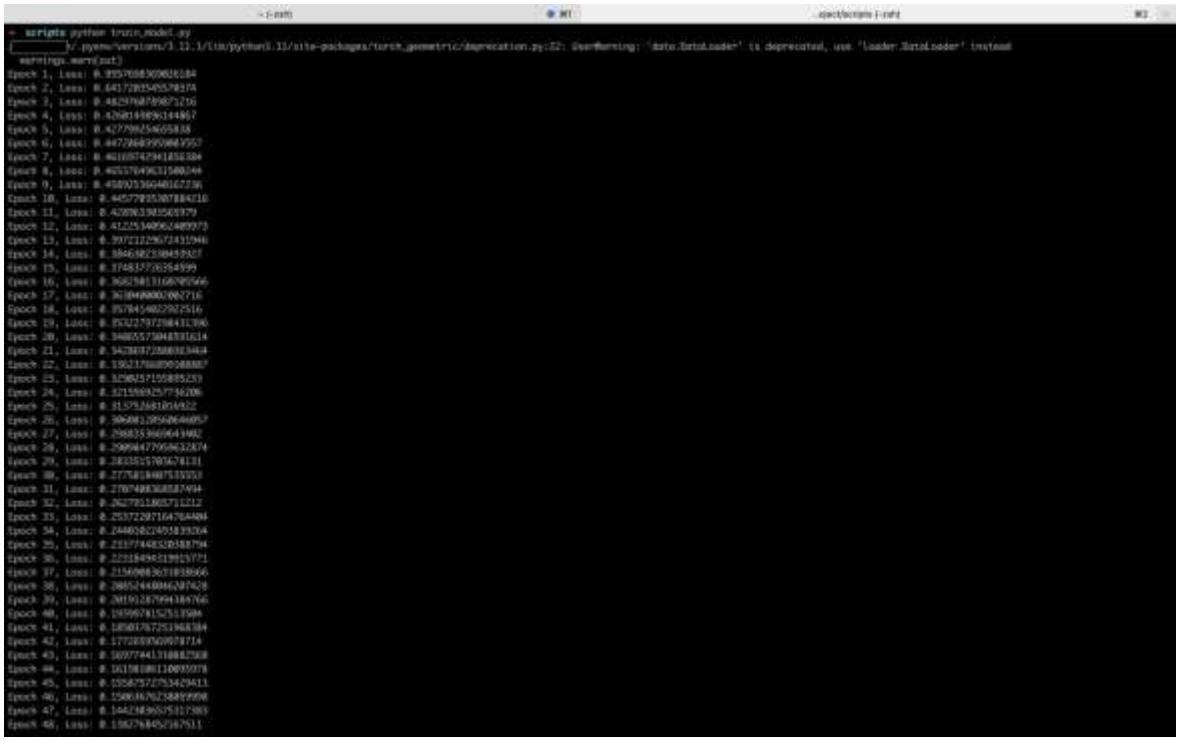
EK-5. Model veri seti



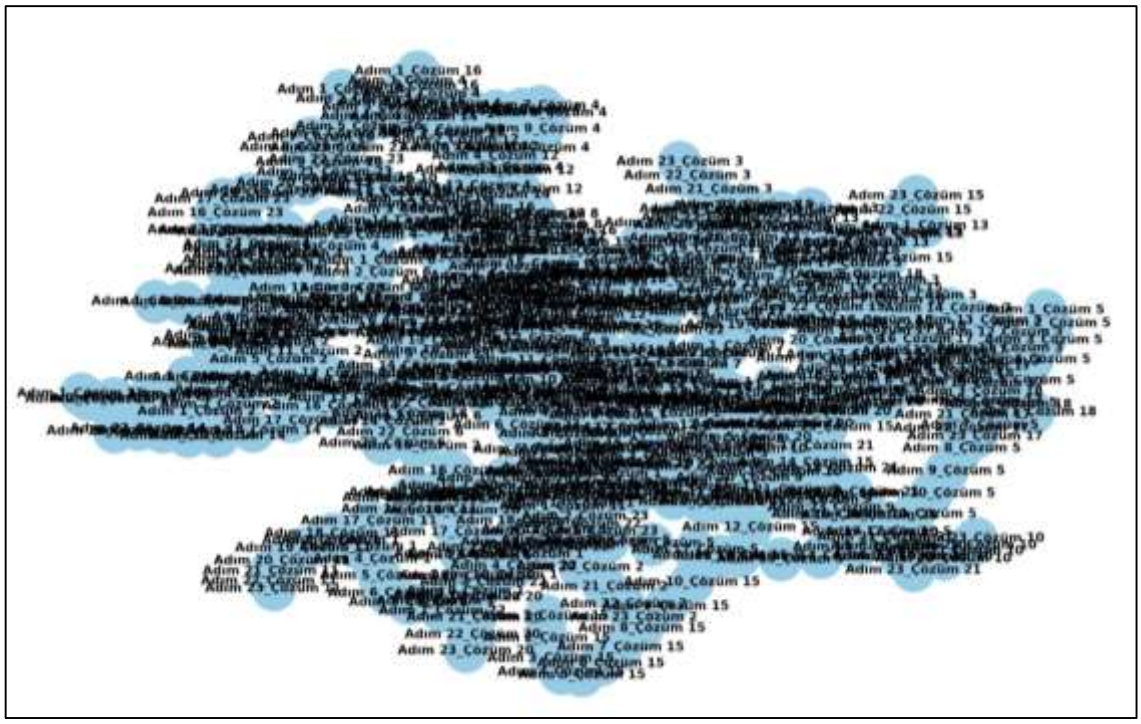
EK-6. Modelin GNN uygulaması graf sonucu.



EK-7. GCN modelinin eğitimi



EK-8. GCN modeli tüm bağlantılara sahip graf.



EK-9. GCN model kodu

```
import networkx as nx
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import matplotlib.pyplot as plt

# Yardım ekibinin başlangıç konumunu al
def get_starting_location():
    while True:
        try:
            latitude = float(input("Yardım ekibinin başlangıç enlem (latitude) değerini girin: "))
            longitude = float(input("Yardım ekibinin başlangıç boylam (longitude) değerini girin: "))
            return latitude, longitude
        except ValueError:
            print("Geçersiz giriş. Lütfen sayısal bir değer girin.")

# Yardım ekibinin başlangıç konumunu al
starting_location = get_starting_location()
print("Başlangıç konumu:", starting_location)

# Toplanma alanları bilgileri
toplantı_alanları = {
    "19 Mayıs": (41.06115716919379, 28.994107481132655),
    "Fulya1": (41.06206776718406, 28.996324998428154),
    "Fulya2": (41.066290770960634, 28.996749253975665),
    "Halaskargazi": (41.05496457018199, 28.993894959167783),
    "Teşvikiye1": (41.05215362137345, 28.993047988353577),
    "Teşvikiye2": (41.051486433463744, 29.001317657670736),
    "Harbiye1": (41.04541091234769, 28.992589015341423),
    "Meşrutiyet": (41.056881980052275, 28.99072955767072),
```

EK-9. (devam) GCN model kodu

```
    "Harbiye2": (41.04292050332294, 28.991622726987828),
    "Halide Edip": (41.063637, 28.982456),
    "Feriköy1": (41.046552391622996, 28.97843353964335),
    "Feriköy2": (41.04533740496555, 28.97794577033266),
    "Harbiye3": (41.04697828945946, 28.988935339643785),
    "Halil Rıfat Paşa": (41.06223134685876, 28.96839161638185),
    "Harbiye4": (41.04206277682959, 28.990712785674734),
    "Esentepe": (41.071640705183036, 29.009513055009453),
    "Kaptan Paşa": (41.06203303190189, 28.972396497328916),
    "Kuştepe": (41.07370919220789, 28.991366624320953),
    "Duatepe": (41.053329703147256, 28.978801416366714),
    "Mahmut Şevket Paşa": (41.05392683518495, 28.967906808755263),
    "Cumhuriyet": (41.05620503623092, 28.984981423791712),
    "Feriköy3": (41.051676390547335, 28.979287839647526),
    "Harbiye5": (41.04662206081584, 28.98716964705904)
}

# Yardım ekibinin başlangıç konumunu ekleyen bir giriş özelliği oluştur
for alan, koordinat in toplantı_alanları.items():
    toplantı_alanları[alan] = koordinat[0] + starting_location[0], koordinat[1] +
starting_location[1]

# Toplanma alanları arasındaki mesafeleri hesapla
graf = nx.Graph()
for alan1, koordinat1 in toplantı_alanları.items():
    for alan2, koordinat2 in toplantı_alanları.items():
        if alan1 != alan2:
            mesafe = ((koordinat1[0] - koordinat2[0]) ** 2 + (koordinat1[1] - koordinat2[1]) **
2) ** 0.5
            graf.add_edge(alan1, alan2, weight=mesafe)

# GCN modeli
```

EK-9. (devam) GCN model kodu

```
class GCN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(GCN, self).__init__()
        self.gc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.gc2 = nn.Linear(hidden_size, output_size)

    def forward(self, adjacency_matrix, feature_matrix):
        x = torch.matmul(adjacency_matrix, feature_matrix)
        x = self.gc1(x)
        x = self.relu(x)
        x = self.gc2(x)
        return x

# GCN parametreleri
input_size = len(toplantı_alanları)
hidden_size = 16
output_size = len(toplantı_alanları)

# GCN modelini oluştur
model = GCN(input_size, hidden_size, output_size)

# Veriyi tensor formatına çevir
adjacency_matrix = torch.tensor(nx.adjacency_matrix(graf).toarray(), dtype=torch.float32)

# adjacency_matrix'i iki boyutlu bir tensöre dönüştür
adjacency_matrix = torch.unsqueeze(adjacency_matrix, 0)

# feature_matrix'i tanımla
feature_matrix = torch.eye(input_size)

# Optimizasyon ve loss fonksiyonu
```

EK-9. (devam) GCN model kodu

```
optimizer = optim.Adam(model.parameters(), lr=0.01)
criterion = nn.CrossEntropyLoss()

# Eğitim
for epoch in range(1000):
    optimizer.zero_grad()
    outputs = model(adjacency_matrix, feature_matrix)

    # Softmax aktivasyonunu uygula
    outputs = F.softmax(outputs, dim=1)

    # Etiketleri oluştur (tamsayı yerine one-hot encoding)
    labels = torch.eye(output_size)[torch.arange(output_size)].unsqueeze(0)

    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()

# Tahminleri al ve sırala
_, predicted_indices = torch.sort(outputs, 1, descending=True)

# Toplanma alanlarına gidilecek sıralama ve matrisleri göster
for j in range(len(predicted_indices[0])):
    gidilecek_alanlar_indices = predicted_indices[0, j].tolist()  # Tensor'ı listeye dönüştür
    gidilecek_alanlar = list(toplantı_alanları.keys())
    gidilecek_alanlar_indices = [gidilecek_alanlar_indices]

    # Matrisi göster
    selected_adjacency_matrix = adjacency_matrix[0, gidilecek_alanlar_indices, :][:, :,
gidilecek_alanlar_indices]
    print(f"\nMatris {j + 1}:\n{selected_adjacency_matrix}")
```

EK-9. (devam) GCN model kodu

```
    # Grafiği görselleştir
    g = nx.Graph()
    for alan in gidilecek_alanlar[0]:
        g.add_node(alan)

    for i in range(len(gidilecek_alanlar[0]) - 1):
        alan1 = gidilecek_alanlar[0][i]
        alan2 = gidilecek_alanlar[0][i + 1]
        g.add_edge(alan1, alan2)

    plt.figure()
    pos = nx.spring_layout(g)
    nx.draw(g, pos, with_labels=True, font_weight='bold', node_size=700,
node_color='skyblue', font_color='black',
            font_size=8, edge_color='gray')
    plt.title(f"Çözüm {j + 1}")
    plt.show()
```



Gazili olmak ayrıcalıktır