Makale Graf Analiz Uygulaması: 3. Proje Raporu

Zehra Gülmüş
zehragulmus25@gmail.com

Ahsen İkbal Türk
akturk1176@gmail.com

***Özet*—Bu çalışmada, bilimsel makaleler arasındaki karmaşık atıf ilişkilerini analiz etmek, metrikleri hesaplamak ve büyük veri setlerini görselleştirmek amacıyla masaüstü tabanlı bir yazılım geliştirilmiştir. Proje, JSON formatındaki ham veriyi işleyerek yönlü graf (directed graph) modeline dönüştürmekte ve harici bir grafik kütüphanesi kullanmadan GDI+ teknolojisi ile görselleştirmektedir. Geliştirilen sistemde; düğümlerin önem derecesini belirleyen *Betweenness Centrality* (Arasılık Merkeziliği), ağın dayanıklılığını ölçen *K-Core Decomposition* ve yazarların üretkenliğini analiz eden *H-Index* algoritmaları sıfırdan implemente edilmiştir. Ayrıca, geliştirilen arayüz üzerinden kullanıcıların graf üzerinde yakınlaştırma (zoom), kaydırma (pan) ve detay görüntüleme işlemleri yapabilmesi sağlanmıştır. Deneysel sonuçlar, uygulamanın binlerce düğüm içeren ağlarda bile yüksek performansla çalıştığını ve hesaplanan metriklerin görsel öğelerle (renk, boyut) başarıyla eşleştirildiğini göstermektedir.**

***Anahtar Kelimeler*—Graf Teorisi, Sosyal Ağ Analizi, Betweenness Centrality, K-Core, H-Index, C#, Veri Görselleştirme.**

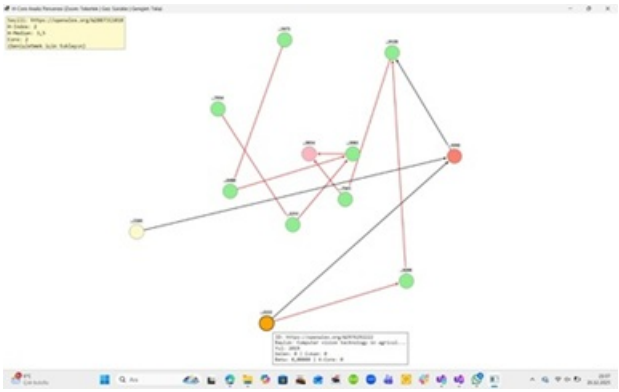
I. Giriş

Bilimsel literatürün hızla büyümesi, araştırmacıların ilgili çalışmaları takip etmesini ve makaleler arasındaki etki ilişkisini anlamasını zorlaştırmaktadır. Atıf ağları, bilginin yayılımını modellemek için kullanılan en etkili yöntemlerden biridir. Bu ağlarda makaleler düğüm (node), atıflar ise kenar (edge) olarak temsil edilir.

Bu projenin temel amacı, akademik atıf ağlarını analiz edebilen, modüler ve yüksek performanslı bir yazılım aracı geliştirmektir. Literatürdeki mevcut araçların aksine, bu çalışmada temel graf algoritmalarının "black-box" (hazır kütüphane) olarak kullanılması yerine, algoritmaların temel mantığının kavranması amacıyla C# dilinde özgün implementasyonları gerçekleştirilmiştir.

Proje kapsamında aşağıdaki temel problemler ele alınmıştır:

- JSON tabanlı yapısal olmayan verinin nesne yönelimli graf modeline dönüştürülmesi.
- Brandes algoritması kullanılarak en kısa yol tabanlı merkezilik hesaplamaları.
- Ağın çekirdek yapısını ortaya çıkaran K-Core analizi.
- Kullanıcı etkileşimli, çift tamponlama (double buffering) destekli bir görselleştirme motorunun tasarımı.



Şekil 1. K-Core değerlerine göre renklendirilmiş ana graf görünümü.

II. Yöntem

Geliştirilen yazılım, Nesne Yönelimli Programlama (OOP) prensiplerine uygun olarak tasarlanmıştır. Sistem üç ana katmandan oluşmaktadır: Veri Katmanı, Analiz Katmanı ve Sunum Katmanı.

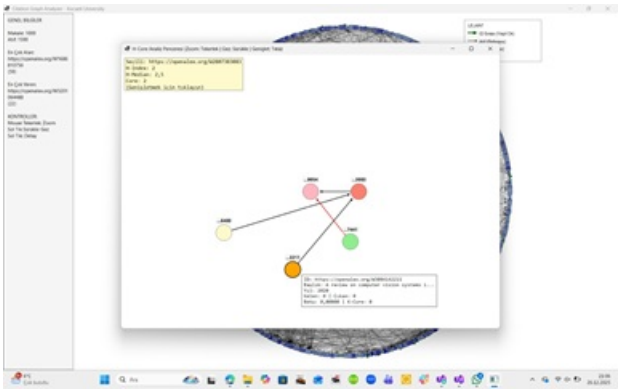
A. Veri Modelleme

Projede kullanılan veri seti JSON formatındadır. Verinin belleğe alınması sürecinde, dış bağımlılığı en aza indirmek ve algoritma hakimiyetini artırmak amacıyla hazır bir serileştirme kütüphanesi (System.Text.Json vb.) kullanılmamıştır.

Bunun yerine, ham JSON verisini karakter düzeyinde işleyen özgün bir `JsonDocumentParser` sınıfı geliştirilmiştir. Bu sınıf, metin üzerinde `IndexOf`, `Substring` ve döngüsel karakter kontrolleri yaparak nesne bloklarını ({}) ve dizileri ([]) tespit etmekte, ardından `ArticleDocument` nesnelerine dönüştürmektedir.

Graf yapısı, `Node` (Düğüm) ve `Edge` (Kenar) sınıfları üzerinden kurgulanmıştır.

- **Node:** Makale ID'si, başlığı ve yılı bilgilerini tutar. `IEquatable` arayüzünü implemente ederek hash tabanlı koleksiyonlarda performanslı çalışması sağlanmıştır.
- **Edge:** Kaynak ve hedef düğümleri tutar. Kenarlar "Citation" (Atıf) ve "Sequence" (Sıralama) olmak üzere iki tiptedir.



Şekil 2. Kaynak ve hedef düğümler.

B. Analiz Algoritmaları

Projenin analiz katmanında, `GraphAnalyzer` soyut sınıfından türetilen üç ana analizör bulunmaktadır.

1) K-Core Decomposition: K-Core analizi, ağdaki önemsiz veya kenarda kalmış düğümleri eleyerek ağın "çekirdek" yapısını bulmayı hedefler. Bu işlem için literatürdeki temel yöntemler takip edilmiştir [3].

2) Betweenness Centrality (Brandes Algoritması): Bir düğümün ağdaki bilgi akışını ne kadar kontrol ettiğini ölçen bu metrik için Brandes'in algoritması kullanılmıştır. `BetweennessAnalyzer` sınıfındaki implementasyon şu adımları izler:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

Burada σ_{st}, s ve t arasındaki en kısa yol sayısını; $\sigma_{st}(v)$ ise bu yollardan v düğümünden geçenlerin sayısını ifade eder. Algoritma, her düğüm için BFS (Breadth-First Search) çalıştırarak en kısa yolları bulur ve skoru geriye doğru yayarak hesaplar [2].

3) H-Index ve H-Core Hesaplaması: Belirli bir makalenin etki alanını hesaplamak için `HIndexCalculator` sınıfı geliştirilmiştir. Algoritma, hedef makaleye atıf yapan makaleleri tespit eder ve bu komşuların kendi atıf sayılarına göre sıralama yapar [1].

$$H = \max\{h \mid \text{En az } h \text{ makalenin atıf sayısı } \geq h\}$$

Bu hesaplama, dinamik olarak açılan `HCoreGraphForm` penceresinde görselleştirilir.

C. Sunum ve Görselleştirme Katmanı

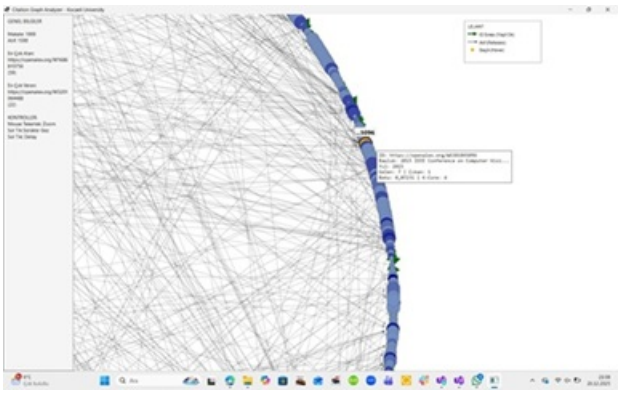
Projenin kullanıcı arayüzü ve görselleştirme modülü, harici bir grafik kütüphanesi kullanılmadan, .NET Framework içerisindeki **GDI+ (System.Drawing)** kütüphanesi ile özgün olarak geliştirilmiştir. Bu katman, analiz sonuçlarının son kullanıcıya etkileşimli ve anlaşılır bir şekilde sunulmasını sağlar.

1) Özgün Çizim Motoru: Grafın çizimi için `GraphImager` sınıfı tasarlanmıştır. Bu sınıf, graf verisini alır ve her bir düğüm ile kenarı vektörel olarak çizer. Çizim performansı, özellikle binlerce düğüm içeren büyük ağlarda kritik öneme sahiptir. Bu nedenle, ekrandaki titremeyi (flickering) önlemek ve akıcı bir görüntü sağlamak amacıyla **Çift Tamponlama (Double Buffering)** tekniği kullanılmıştır. Bu teknikte, çizim işlemleri önce bellekteki sanal bir yüzeye (buffer) yapılır ve işlem bittiğinde tek seferde ekrana transfer edilir.

2) Koordinat Dönüşüm Sistemi: Kullanıcıların büyük graflar üzerinde detaylı inceleme yapabilmesi için "Yakınlaştırma" (Zoom) ve "Kaydırma" (Pan) özellikleri geliştirilmiştir. Ekran koordinatları (Mouse) ile dünya koordinatları (Graf) arasındaki eşleşme, afin dönüşüm matrisleri kullanılarak sağlanmıştır:

$$P_{world} = \frac{P_{screen} - Offset}{Scale} \tag{2}$$

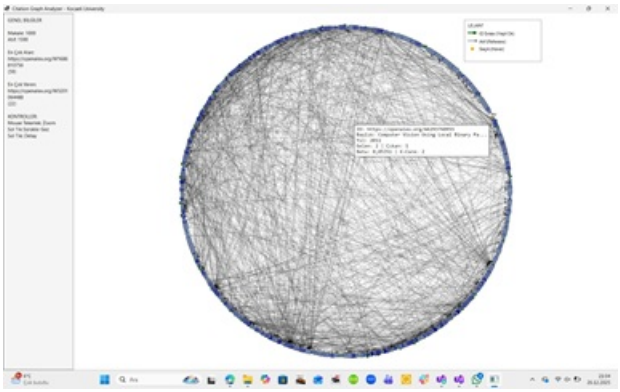
Bu dönüşüm sayesinde kullanıcı fare tekerleği ile grafa $0.1x$ ile $10x$ arasında yakınlaşabilmekte ve grafı sürükleyebilmektedir.



Şekil 3. Zoomlanmış görünüm.

3) Görsel Kodlama (Visual Encoding): Analiz sonuçlarının görsel olarak ayırt edilebilmesi için düğümlerin renk ve boyut özellikleri dinamik olarak hesaplanmaktadır:

- **Renk (K-Core):** Ana graf üzerinde düğümler, hesaplanan *K-Core* değerlerine göre renklendirilir. Düşük dereceli (kenarda kalan) düğümler **açık mavi** tonlarıyla, ağın merkezindeki (yüksek K-Core) düğümler ise **koyu mavi** tonlarıyla gösterilir.
- **Boyut (Betweenness):** Düğümlerin yarıçapı, *Betweenness Centrality* skoruna göre ölçeklenir. Böylece ağdaki merkezi düğümler daha büyük ve belirgin hale gelir.
- **Etkileşim:** Kullanıcı bir düğüme tıkladığında, ilgili makalenin H-Core kümesini gösteren, Şekil 5'teki gibi sarı ve yeşil renklerin kullanıldığı detay penceresi açılır.



Şekil 4. Genel görünüm.

III. Deneysel Sonuçlar

Geliştirilen uygulama, sağlanan test veri setleri üzerinde çalıştırılmış ve analiz metrikleri doğrulanmıştır.

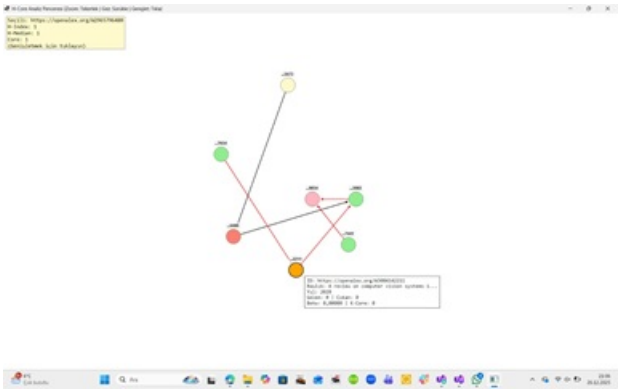
A. Analiz Çıktıları

Uygulama yüklendiğinde sol panelde (`pnlStats`) genel istatistikler sunulmaktadır:

- Toplam Makale (Düğüm) ve Atıf (Kenar) sayısı.
- En çok atıf alan makalenin ID'si ve atıf sayısı (In-Degree).
- En çok atıf veren makalenin ID'si ve sayısı (Out-Degree).

B. Görselleştirme Performansı

Yapılan testlerde, 'GraphImager' sınıfının çizim performansı gözlemlenmiştir. GDI+ kütüphanesi kullanılarak oluşturulan çizimlerde [5], düğümlerin üzerine gelindiğinde (Hover) detaylı bilgi kartları gösterilmektedir.



Şekil 5. Detaylı bilgi kartları.

Düğümlerin üzerine gelindiğinde, düğümün ID, Başlık, Yıl ve hesaplanan Betweenness değeri kullanıcıya anlık olarak sunulmaktadır. Ayrıca, bir düğüme tıklandığında açılan 'HCoreGraphForm' penceresi, sadece ilgili makalenin etki alanını izole ederek göstermekte, bu da karmaşık ağlarda analizi kolaylaştırmaktadır.

IV. Sonuç

Bu proje kapsamında, bilgisayar mühendisliği ve veri bilimi için kritik öneme sahip olan graf analizi teknikleri pratik bir uygulamaya dönüştürülmüştür. C# programlama dili ve .NET Framework kullanılarak geliştirilen yazılım, dış bağımlılık olmadan karmaşık hesaplamaları (Brandes, K-Core) gerçekleştirebilmektedir.

Çalışmanın sonucunda elde edilen kazanımlar şunlardır:

1) Yönlü ve ağırlıksız graf yapılarının verimli bir şekilde bellekte tutulması sağlanmıştır.
2) Matematiksel graf teorisi algoritmaları kod düzeyinde implemente edilmiştir.
3) Kullanıcı dostu, interaktif bir görselleştirme aracı ortaya konulmuştur.

Gelecek çalışmalarda, graf çizimi için "Force-Directed" (Kuvvete dayalı) yerleşim algoritmaları eklenerek düğümlerin kümelenmesinin daha net görülmesi sağlanabilir.

V. Algoritmaların Kaba Kodu

Aşağıda, projede kullanılan temel algoritmaların işleyiş mantığı kaba kod (pseudocode) formatında sunulmuştur.

A. Genel Sistem Akışı

```
FONKSİYON SistemBaslat(jsonDosyasi):

    EĞER jsonDosyasi BOŞ İSE:
        DÖN "Hata: Dosya seçilmedi."

    Graf = Yeni Graf()
    HamVeri = DosyaMetniniOku(jsonDosyasi)
    VeriListesi = BosListe()

    DÖNGÜ (HamVeri bitene kadar):
        Blok = SüslüParantezBloguBul(HamVeri)

        EĞER Blok Bulunduysa:
            id = MetinAyikla(Blok, "\"id\":")
            baslik = MetinAyikla(Blok, "\"
            title\":")
            yil = SayiAyikla(Blok,"\"year\
            ":")
            refs = DiziAyikla(Blok, "\"
            referenced_works\":")

            Makale = Yeni Makale(id, baslik,
            yil, refs)
            VeriListesi.Ekle(Makale)

    HER makale İÇİN VeriListesi İÇİNDE:
        Düğüm = Yeni Düğüm(makale.id, makale.
        baslik, makale.yil)
        Graf.DugumEkle(Düğüm)

    HER makale İÇİN VeriListesi İÇİNDE:
        Kaynak = Graf.DugumBul(makale.id)

        HER ref_id İÇİN makale.refs İÇİNDE:
            Hedef = Graf.DugumBul(ref_id)

            EĞER Hedef VARSA:
                Graf.KenarEkle(Kaynak, Hedef,
```

```
                TİP="Atıf")

    AnalizleriCalistir(Graf)
    DÖN Graf
```

Listing 1. Sistem Başlatma ve Manuel JSON Ayrıştırma

B. K-Core Ayrıştırma Algoritması

```
FONKSİYON KCoreAnalizi(Graf):

    Dereceler = Graf.DereceleriHesapla()
    AktifDugumler = Graf.TumDugumler()
    k = 0

    DÖNGÜ (AktifDugumler BOŞ OLANA KADAR):

        TEKRARLA:
            silinenVarMi = YANLIŞ

            HER dugum İÇİN AktifDugumler
            İÇİNDE:

                EĞER Dereceler[dugum] <= k
                İSE:
                    AktifDugumler.Cikar(dugum)
                    SonucHaritasi[dugum] = k

                    HER komsu İÇİN dugum.
                    Komsular İÇİNDE:
                        Dereceler[komsu] =
                        Dereceler[komsu] - 1

                    silinenVarMi = DOĞRU

        VE (silinenVarMi == DOĞRU) İSE DEVAM
        ET

        k = k + 1
```

Listing 2. K-Core Analizi

C. Betweenness Centrality (Brandes)

```
FONKSİYON BrandesAlgoritmasi(Graf):

    HER d İÇİN Graf.Dugumler: Skor[d] = 0

    HER kaynak İÇİN Graf.Dugumler:
        Yigin = Yeni Yigin()
        Kuyruk = Yeni Kuyruk()
        Kuyruk.Ekle(kaynak)

        DÖNGÜ Kuyruk BOŞ DEĞİLSE:
            v = Kuyruk.Cikar()
            Yigin.Ekle(v)

            HER komsu İÇİN v.Komsular:
                EĞER komsu ZiyaretEdilmedi
                İSE:
                    Kuyruk.Ekle(komsu)
                    Mesafe[komsu] =
                    Mesafe[v] + 1

                EĞER EnKisaYol Üzerindeyse:
                    Sigma[komsu] =Sigma[komsu]
                    + Sigma[v]

        DÖNGÜ Yigin BOŞ DEĞİLSE:
            w = Yigin.Cikar()

            HER v İÇİN w.Onculleri:
                c = (Sigma[v] / Sigma[w]) *
                (1 + Delta[w])
                Delta[v] = Delta[v] + c

            EĞER w != kaynak İSE:
                Skor[w] = Skor[w] + Delta[w]

    NormalizeEt(Skor)
    DÖN Skor
```

Listing 3. Betweenness Centrality Hesaplama

D. H-Index Hesaplama

```
FONKSİYON HIndexHesapla(HedefDugum, Graf):

    ReferansVerenler = Graf.GetirenleriBul
    (HedefDugum)
    AtifListesi = []

    HER r İÇİN ReferansVerenler İÇİNDE:
        Sayi = GlobalAtifSayisi[r]
        AtifListesi.Ekle(Sayi)

    AtifListesi.Sirala(BUYUKTEN_KUCUGE)
    h_degeri = 0

    HER i İÇİN (0'dan ListeUzunlugu'na):
        EĞER AtifListesi[i] >= (i + 1) İSE:
            h_degeri = i + 1
        YOKSA:
            DUR (Döngüden Çık)

    CoreListesi = AtifListesi.Ilk(h_degeri)
    Median = OrtancaHesapla(CoreListesi)

    DÖN h_degeri, Median, CoreListesi
```

Listing 4. H-Index ve H-Core Bulma

KAYNAKLAR

[1] J. E. Hirsch, "An index to quantify an individual's scientific research output," Proceedings of the National Academy of Sciences, vol. 102, no. 46, pp. 16569–16572, Nov. 2005.

[2] U. Brandes, "A faster algorithm for betweenness centrality," Journal of Mathematical Sociology, vol. 25, no. 2, pp. 163–177, 2001.

[3] V. Batagelj ve M. Zaversnik, "An O(m) algorithm for cores decomposition of networks," Advances in Data Analysis and Classification, vol. 5, no. 2, pp. 129–145, 2011.

[4] M. E. J. Newman, "The structure and function of complex networks," SIAM Review, vol. 45, no. 2, pp. 167–256, 2003.

[5] Microsoft, "Graphics and Drawing in Windows Forms," .NET Documentation. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/advanced/graphics-and-drawing-in-windows-forms. [Erişim Tarihi: 26-Ara-2025].