Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 29, Sayı 2, 434-439, 2025

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 29, Issue 2, 434-439, 2025

DOI: 10.19113/sdufenbed.1639530

Çizgelerde Ağırlıklı Bağlama Sayısı Üzerine

Şevket KESER[1], Ersin ASLAN*[2]

[1,2] Manisa Celal Bayar Üniversitesi, Hasan Ferdi Turgutlu Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, 45400, Manisa, Türkiye

(Alınış / Received: 08.07.2025, Kabul / Accepted: 07.08.2025, Online Yayınlanma / Published Online: 25.08.2025)

Anahtar Kelimeler
Çizge teori,
Ayrıt bütünlüğü,
Algoritmalar

Öz: Lojistik ağları, internet ağları, su ve kanalizasyon ağları, elektrik ağları gibi tüm ağlar çizge teori yardımıyla modellenebilmektedir. Bu ağların güvenilirliği, sağlamlığı, taşıyabilme kapasiteleri gibi benzer durumlar için incelenmesi gerekmektedir. Bu incelemeler, çizge teorideki zedelenebilirlik parametreleri yardımıyla sağlanabilmektedir. Ağırlıklı çizgeler yardımıyla, ağırlıklı zedelenebilirlik ölçümleri kullanılabilmektedir. Ağırlıklı olmayan çizgelerde tepe ağırlıkları 1 olarak kabul edilmektedir. Böylece ağırlıklı zedelenebilirlik ölçümleri hem ağırlıklı hem de ağırlıksız çizgeler için ayırt edici olmaktadırlar. Bu makalede ilk kez ağırlıklı bağlama sayısı tanımı verilecektir. Bu kavram detaylıca açıklanmıştır. Bu ağırlıklı bağlama sayısı için algoritma ve pseudo kod verilecektir. Ağırlıklı çizgelerin bazı durumlarda daha iyi bir ayırt edici olduğu verilmiştir. Bu parametrenin bazı çizgeler için önemi detaylıca açıklanmıştır.

On the Weighted Binding Number of Graphs

Keywords
Graph theory,
Edge integrity,
Algorithms

Abstract: All networks such as logistics networks, internet networks, water and sewer networks, and electricity networks can be modeled with the help of graph theory. These networks need to be examined for similar situations such as reliability, durability and carrying capacity. These investigations can be achieved with the help of vulnerability parameters in graph theory. With the help of weighted graphs, weighted vulnerability measurements can be used. In unweighted graphs, vertex weights are assumed to be 1. Thus, weighted vulnerability measurements are discriminative for both weighted and unweighted graphs. In this article, the definition of weighted binding number will be given for the first time. This concept is explained in detail. The algorithm and pseudo code for this weighted binding number will be given. It has been shown that weighted graphs are better discriminators in some cases. The importance of this parameter for some graphs is explained in detail.

1. Giriş

Ağların; güvenliği, iyileştirilmesi, korunması önemli çalışma konularındandır. Ağlarda oluşabilecek problemlere çözüm önerileri için bu problemlerin modellenmesi gerekmektedir. Bu problemleri ya da ağ yapılarını matematiksel olarak modellemek için çizge teori kullanılmaktadır [1]. Çizge teoride, bir modeldeki noktaya ya da merkeze tepe karşılık gelmektedir ve benzer şekilde bir modeldeki kenar, hat ya da bağlantılara ayrıt karşılık gelir. Çizge teoride tepeler kümesi $V(G)$, ayrıtlar kümesi $E(G)$ olarak gösterilir [2]. Bu makalede kullanılan tanımlar [2] kitabından alınmıştır.

Ağ yapıların elemanlarının zarara uğraması sonrası durumlar için zedelenme kavramı ortaya çıkmaktadır [1]. Bir ağın ya da yapının merkezlerinin ya da bağlantı hatlarının zarar gördükten sonra ağın ya da yapının işlevini yitirene kadar gösterdiği direnç ya da dayanma gücüne zedelenebilirlik denir [1]. Zedelenebilirlik kavramını incelemek için internet sağlayıcıları bir örnek olarak ele alınabilir. Çizgede; her bir müşteri merkezi, müşteriler arasındaki hatlar da ayrıtları temsil etsin. Burada, internet sağlayıcısının tüm müşterilerine mümkün olan en iyi hizmeti sunması hedeflenmiştir. İnternet sağlayıcısı, birtakım durumlardan dolayı sorunlar

* İlgili yazar: ersin.aslan@cbu.edu.tr

Bu çalışma birinci yazarın yüksek lisans tezinden üretilmiştir.

yaşayabilecektir. Örneğin sadece bir müşteride var olan bir sıkıntı bütün herkese etki etmeyebilir. Ağ uygun bir şekilde çalışmakta olup sadece müşterinin sorunu giderilerek makul bir sürede sorun aşılmış olacaktır. Benzer şekilde bir mahallede birtakım sorunlar olduğunda durum incelendiğinde, sorunların çözümü için bu mahalleye yeni bir hat döşenmesi çözüm olabilir veya mahallenin bakıma alınması durumları ele alınmalıdır. Burada önemli olan durum, bu sıkıntının çözümünün zaman ve faaliyet açısından irdelenmesidir. Daha genel olarak il bazında sorunlar oluşabilecektir. Çözüm için başka bölgelerden sıkıntılı ile internet akışı sağlanabilmesi gibi vb. başka çözümler önceden üretilecektir. Amacımız, sorunlar oluşmadan önce bu durumları göz önüne alıp yapılabilecek tüm çözümleri hazır hale getirmektir.

Çizgilerde zedelenebilirlik kavramı için çizgelerin tepeleri, ayrıtları, komşuluk ve ağırlıklandırılma durumları incelenebilmektedir. Bunlarla ilgili olarak birçok zedelenebilirlik ölçümleri ortaya çıkmıştır.

Zedelenebilirlikte tepe durumlarını incelemek için bir örnek verelim. İnternet ağı için merkez olarak gösterilen müşterileri ve internet sağlayıcıları düşünelim. Müşterilerde var olan sıkıntılar ya da internet sağlayıcıları için bazı sıkıntılar ortaya çıkabilir. Burada birkaç durum ortaya çıkar. Bunlar yapı zarar gördükten sonra kaç parçaya bölündü, geriye kalan yapıda hizmete devam eden en çok kaç müşteriye sahip ya da kaç müşteri zarar gördü vb. sorunlar olmaktadır. Bu sorunları incelemek için bağlantılılık sayısı [1], saçılım sayısı [3] gibi zedelenebilirlik ölçümleri tanımlanmıştır.

Zedelenebilirlikte ayrıtların durumlarını incelemek için bir örnek verelim. Lojistik taşıma ağını ele alalım. Karayollarında ortaya çıkabilecek durumları incelediğimizde yol yapım çalışmaları, kazalar ya da meteorolojik hava durumlarına göre yollar kapanabilmektedir. Bu tarz durumlarda merkezler arasında yollar hâlâ var mı, ya da lojistik ağı kaç parçaya bölündü, ya da birbiriyle etkileşimli kaç merkez vardır? Bu tarz soruları arttırabiliriz. Bu durumların yanıtları için ayrıt bütünlük değeri [1], ayrıt saçılım sayısı [4,5] gibi zedelenebilirlik ölçümleri tanımlanmıştır.

Zedelenebilirlikte komşuluk durumları için inceleme yaptığımız durum olarak bir örnek verelim. Bilgisayar ağları için bazı durumlarda virüs girerek ağa zarar verdiği durumu ele alalım. Bu durumlarda merkez ve ağdaki bu merkeze komşu olan tüm merkezler de zarar görür. Bu durumlarda ağda iletişim devam ediyor mu, ya da ağ kaç kısma bölündü, ya da kaç merkez zarar gördü şeklinde sorular üretilebilir. Bu sorunlar için bağlama sayısı [6], komşu saçılım sayısı [7] gibi zedelenebilirlik ölçümleri tanımlanmıştır.

Çizgelerde tepeler ağırlıklandırılmış ise bu çizgelere ağırlıklı çizgeler denir [2]. Zedelenebilirlikte ağırlıklı ağ yapılarını incelemeyi bir örnek ile açıklayalım. Su taşıma ağını ele alalım. Barajlar ya da su kaynaklarından tüm hane halkına ya da kullanıcılara su ulaşması gerekmektedir. Taşınan su miktarının hane halkına yetmesi gerekmektedir. Bunun için var olan ağların merkezlerine ağırlık verilerek ağ içinde taşıma durumuyla birlikte müşterilere suyun ulaşması sağlanmaktadır. Burada önemli olan barajların ya da kaynakların korunmasıdır. Su hatları zarara uğradığında su dağıtımının yine devam etmesi gereklidir. Ayrıca taşınan su miktarının hane halkına yetmesi gerekmektedir. Bu kaynaklar zarara uğradığında yine benzer sorunlar ortaya çıkmaktadır. Bu sorunlara çözüm bulmak için ağırlıklı saçılım sayısı [7], ağırlıklı bütünlük sayısı [8] kavramları tanımlanmıştır.

Yukarıda verilen örnekler dışında birçok zedelenebilirlik parametreleri tanımlanmış ve birçok çalışma yapılmıştır [9-14].

2. Ağırlıklı Bağlama Sayısı

Zedelenebilirlik parametreleri incelendiğinde bazı durumlarda var olan parametre ayırt edici özellikte olamamaktadır. Örneğin saçılım sayısı her durum için uygun sonuç üretilemediği için tepelerin ağırlıklı durumları için ağırlıklı saçılım sayısı üretilmiştir. Benzer durumdan bağlama sayısı için bu geliştirme yöntemiyle ağırlıklı bağlama sayısı ilk kez bizim tarafımızdan verilecektir.

2.1. Tanım

Ağırlıklı bağlama sayısının tanımı aşağıda verilecektir.

Tanım 1. G bağlantılı, v $\in V(G)$ ve $w: V \to R \geq 0$ ağırlık fonksiyonlu bir çizge olmak üzere **ağırlıklı bağlama sayısı;** $bind_{vw}(G)$ ile gösterilir ve tanımı

$$bind_{vw}(G) = \min_{S \in F(G)} \left\{ \frac{w(N(S))}{w(S)} \right\} \qquad (1)$$

Burada S, çizgede seçilen kümeyi, $w(S) = \sum_{v \in S} w(v)$ ise çizgede seçilen kümedeki tepe kümesindeki ağırlıklar toplamını, $N(S)$: çizgeden seçilen tepelere komşu olan tepeler kümesini, $w(N(S)) = \sum_{v \in N(S)} w(v)$ ise çizgede seçilen tepelere komşu tepelerin ağırlıkların toplamını ifade eder. Burada $F(G) = \{ S \subseteq V(G) \mid S \neq \emptyset, \mathrm{N(S)} \neq V(G) \}$ olmalıdır.

Ağırlıklı bağlama sayısında istenen çözümün minimum olması istenmektedir.

2.2 Motivasyon

Bağlama sayısı çizgeler için zedelenebilirlik parametresi olarak kullanılsa da, bazı özel çizgeler için ayırt edici parametre olarak kullanılamamaktadır. Ağırlıksız çizgelerde tepelerin ağırlığı 1 kabul edilmektedir. Ağırlıklı çizgeler, ağırlıksız çizgeleri de

içerdiği için çoğu çizge için ayırt edici bir parametre olarak kullanılabilmektedir.

Teorem 1 [6]. $n \geq 1$ çift tepeli yol çizgesi için

$$bind(P_n) = 1, \quad (2)$$

n tek tepeli yol çizgesi ise,

$$bind(P_n) = \frac{(n-1)}{n-2}. \quad (3)$$

Teorem 2 [6]. $n \geq 1$ yıldız çizgesi için

$$bind(K_{1,n}) = \frac{1}{n}. \quad (4)$$

Örnek 1. Ağırlıksız yol çizgenin bağlama sayı değeri tüm çift ve tek tüm durumlarda aynı değeri verecektir. Ancak ağırlıklı bağlama sayısı durumunda ağırlıklı yol çizgesi için farklı değerler olacağından çizgeler için ayırt edicidir.

Örnek 2. Ağırlıksız yıldız çizge için bağlama sayısı her durumda $\frac{1}{n}$ olacak olup ağırlıklı yıldız çizge için ağırlıklı bağlama sayısı her durumda farklı olacaktır.

Örnek 1 ve 2' den kolayca görülebileceği üzere ağırlıklı çizgeler ağırlıksız çizgeleri de içerdiği için birçok durumda daha iyi ayırt edici özelliğe sahiptir.

3. Ağırlıklı Bağlama Sayısı Algoritması ve Pseudo Kodu

Zedelenebilirlik parametrelerinin algoritmalarıyla ya da pseudo kodlarıyla ilgili olarak şimdiye kadar birçok çalışma yapılmıştır [15-18].

Bu bölümde yeni tanımladığımız ağırlıklı bağlama sayısı için bu parametrenin değerini bulan algoritma, pseudo kodu ve karmaşıklık değeri verilmiştir.

3.1 Algoritma.

- Ağırlıklı Bağlama Sayısı ($bind_{vw}$) Algoritması

1. Girdi: *G* Çizgesi
2. *G* çizgesi için S_i seçimlerinden tüm iterasyonları üret ve oluşan sonuçları listeye (*s_vertex_list*) ekle. (n= |V(G); | S_i | = |*s_vertex_list*| = 2^n -2)
3. **Tüm** S_i seçim iterasyonunda;
   - 3.1. S_i seçim iterasyonu için, *G* çizgesine uygulayıp geriye kalan tepeleri hesapla ve elde edilen sonuçları listeye ata (*cutted_vertex_list*)
   - 3.2. **Tüm** S_i seçim iterasyonunda;
     - 3.2.1. **Tüm** S_i seçim iterasyonu elemanı için;
       - 3.2.1.1. S_i seçim kümesindeki iterasyon için elemanların komşuluklarını bul ve bu tepeleri listeye (*n_vertex_list*) ekle. (G_{adj} matrisi kullanarak geriye kalan tepeler için listelerde komşulukları incele.)
       - 3.2.1.2. Komşuluk bitmiş ise bu tepeyi diğer tepeler (*other_vertex_list)* listesi elemanı olarak ekle.
   - 3.3. Komşu tepeler listesindeki (*n_vertex_list*) aynı elemanları temizle
   - 3.4. Diğer tepeler listesindeki (*other_vertex_list*) aynı elemanları temizle
   - 3.5. $F(G)$ ağırlıklı bağlama sayısı şartına uygun olan S_i seçim iterasyonlarında;
     - 3.5.1. S_i seçim iterasyonundaki ağırlıklar toplamını hesapla. ($w(S_i)$)
     - 3.5.2. S_i seçim iterasyonundaki komşuluktaki tepelerin toplam ağırlıklarını hesapla. ($w(N(S_i)_i)$)
     - 3.5.3. *Tanımdan* $w(N(S_i)_i) / w(S_i)$ oranını S_i için $bind_{vwi}$ değerini bulup hesapla ve değeri listeye ekle (*sample_bindw_list*)
     - 3.5.4. S_i, *n_vertex_list*, $bind_{vwi}$ değerlerini *sample_result_list* listesine ekle
4. Hesaplanan $bind_{vwi}$ değerleri (*sample_bindw_list*) içerisinden, minimum olanı *G* çizgesi için ağırlıklı bağlama sayısı ($bind_{vw}$) değeri olarak sakla.
5. Çıktı: *G* çizgesinin ağırlıklı bağlama sayısını ($bind_{vw}$) değeri olarak yaz.

3.2. Pseudo Kodu

Ağırlıklı Bağlama Sayısı ($bind_{vw}$) Pseudo Kodu

```
Girdi: G, ağırlıklı çizge
Çıktı: G çizgesi için ağırlıklı bağlama sayısını (bind_vw)
Başla
SET sample_bindw_list TO []
# S_i kümesi için bind_vwi hesaplamalarını saklar.
SET sample_result_list TO [] # S_i kümesi için S_i,
#n_vertex_list ve bind_vwi durumlarını saklar.
SET Gadj TO sample_list[sample_index][0].copy()
# G çizgesi için adjoint matris
SET Gw TO sample_list[sample_index][1].copy()
# G çizgesinin ağırlıkları
SET sample_vertex_count TO len(Gadj)
#  G çizgesi tepe sayısı
# S_i seçim iterasyonları bulunmakta
SET iteration_vertex_list TO list(powerset(range(1,
sample_vertex_count + 1)))
FOR s_vertex_list IN iteration_vertex_list:
# (2^n-2 kez çalışmaktadır)
 SET n_vertex_list TO []
# S_i kümesinin komşuluklarını tutar (N(S_i)_i)
 SET other_vertex_list TO []
# Other, N_i'nin komşularını tutar (O(N_i)_i )
 # S_i, kümesi G'ye uygulandığında çizgedeki geriye
#kalan tepeleri bul. (n^2  defa çalışmaktadır)
 SET cutted_vertex_list TO
core.find_cutted_vertex(s_vertex_list,
sample_vertex_count)
 FOR s_vertex_index IN range(0, len(s_vertex_list)):
# (n-1) defa uygulanır
  SET s_vertex TO s_vertex_list[s_vertex_index]
 # seçilen tepe numarasını sakla Gadj icinde a index'i
#ile cutted_vertex'nolu tepe için komşulukları aranıyor.
```

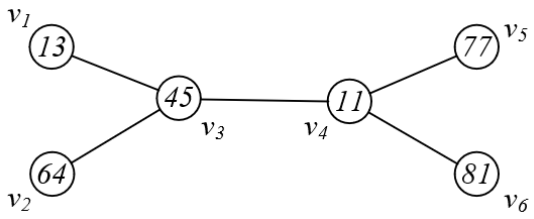
```
FOR g_index IN range(0, sample_vertex_count):
# n defa çalışır incelenen tepe, seçilen tepeye KOMŞU
# ise koşul şartları sağlanmalı.
IF Gadj[s_vertex - 1][g_index] EQUALS 1 and g_index + 1:
# koşul sağlandığında: cutted_vertex numaralı tepe,
# a + 1, ile komşu tepedir
n_vertex_list.append(g_index + 1)
 # n-1 komşu tepe listesine ekle
ELSE:
IF g_index + 1 not IN s_vertex_list and g_index + 1 IN
cutted_vertex_list:
# incelenenen tepe (g_index + 1); seçilen kümesine
#dahil değil ve seçilen kümede de komşu olan tepe
#değil ve geriye kalan tepeler dahil olma durumunda
other_vertex_list.append(g_index + 1)
 # FOR3: end
 # FOR2: end
 # n_vertex_list ile other_vertex_list listeleri içinden
# aynı elemanlar siliniyor.
 SET n_vertex_list TO list(dict.fromkeys(n_vertex_list))
 SET other_vertex_list TO
list(dict.fromkeys(other_vertex_list))
 IF len(n_vertex_list) <sample_vertex_count:
# F(G) genel şartımız
 SET s_count TO len(s_vertex_list)
# seçim kümesi için tepe sayısı
SET n_count TO len(n_vertex_list)
# komşuluk için tepe sayısı
SET bind TO (n_count / s_count)
# bind değeri bulundu
 SET Sw TO
core.calculate_s_vertex_weight(s_vertex_list, Gw)
# w(Si) (n kez)
 SET Nw TO
core.calculate_n_vertex_weight(n_vertex_list, Gw)
#w(Ni) (n kez)
  SET bindw TO Nw / Sw # bind_vwi değerini bul
 sample_bindw_list.append(bindw)
# sonuç listesine  bind_vwi değerini ekle
# FOR1: end
SET BINDw TO min(sample_bindw_list)
# G çizgesi için bulunan   bind_vw  değeri
# END of Program
```

Bu algoritmamızda tek bir seçim iterasyonundan oluşan karmaşıklık $O(n^2)'$ dir. Var olan her seçim stratejisi dahil edildiğinde elde edilen karmaşıklık değerimiz $O(2^n)'$ dir. Buradan algoritma için genel karmaşıklığımız $O(2^n)'$ olacaktır.

4. Ağırlıklı Bağlama Sayısı Algoritma Uygulaması

Ağırlıklı Bağlama Sayısı ($bind_{vw}$) hesaplaması aşağıda verilen örnek uygulama ile açıklanmıştır.

Örnek 3. Şekil 1'de örnek olarak 6 tepeli, ağırlıklı bir G çizgesi ele alınmıştır. Bu çizgelerin tepelerinin ağırlıkları sırasıyla v_1= 13, v_2= 64, v_3= 45, v_4= 11, v_5= 77, v_6 =81 dir.



Şekil 1. Ağırlıklı G çizgesi.

Şekil 1' deki 6 tepeli G çizgesi ele alındığında, S_i kümesi için F(G) şartını sağlayan iterasyonlarda incelenecek küme sayısı $S_i = 2^6 - 2 = 62$ olarak elde edilir. Tüm S_i kümeleri için $bind_{vw}$ değerleri bulunmaktadır.

Tüm durumlar Tablo 1' de verilmiştir. Burada S_i: seçim iterasyonlarını, *N(S):* seçim kümesinin komşuluklarını, $w(N(S))/w(S)$ *:* ağırlıklar cinsinden tanımdaki toplamlarının oranını, $bind_{vw}$ değeri olacak şekilde aşağıda verilen Tablo 1'de gösterilmiştir. Tabloda, *F(G)* şartını sağlamayan seçim iterasyonları için hesaplamaya tanımdan dolayı dahil edilmemiştir.

Tablo 1. G çizgesinde ağırlıklı bağlama sayısı ($bind_{vw}$) durumları

$S_i = S$	*N(S)*	$w(N(S))/w(S)$	$bind_{vw}$
$S_1 = \{v_1\}$	$\{v_3\}$	45 / 13	3.462
$S_2 = \{v_2\}$	$\{v_3\}$	45 / 64	0.703
$S_3 = \{v_3\}$	$\{v_1, v_2, v_4\}$	88 / 45	1.956
$S_4 = \{v_4\}$	$\{v_3, v_5, v_6\}$	203 / 11	18.455
$S_5 = \{v_5\}$	$\{v_4\}$	11 / 77	0.143
$S_6 = \{v_6\}$	$\{v_4\}$	11 / 81	0.136
$S_7 = \{v_1, v_2\}$	$\{v_3\}$	45 / 77	0.584
$S_8 = \{v_1, v_3\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 58	2.293
$S_9 = \{v_1, v_4\}$	$\{v_3, v_5, v_6\}$	203 / 24	8.458
$S_{10} = \{v_1, v_5\}$	$\{v_3, v_4\}$	56 / 90	0.622
$S_{11} = \{v_1, v_6\}$	$\{v_3, v_4\}$	56 / 94	0.596
$S_{12} = \{v_2, v_3\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 109	1.220
$S_{13} = \{v_2, v_4\}$	$\{v_3, v_5, v_6\}$	203 / 75	2.707
$S_{14} = \{v_2, v_5\}$	$\{v_3, v_4\}$	56 / 141	0.397
$S_{15} = \{v_2, v_6\}$	$\{v_3, v_4\}$	56 / 145	0.386
$S_{16} = \{v_3, v_4\}$	$\{v_1, v_2, v_4, v_3, v_5, v_6\}$	-	-
$S_{17} = \{v_3, v_5\}$	$\{v_1, v_2, v_4\}$	88 / 122	0.721
$S_{18} = \{v_3, v_6\}$	$\{v_1, v_2, v_4\}$	88 / 126	0.698
$S_{19} = \{v_4, v_5\}$	$\{v_3, v_5, 6, v_4\}$	214 / 88	2.432
$S_{20} = \{v_4, v_6\}$	$\{v_3, v_5, 6, v_4\}$	214 / 92	2.326
$S_{21} = \{v_5, v_6\}$	**$\{v_4\}$**	**11 / 158**	**0.070**
$S_{22} = \{v_1, v_2, v_3\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 122	1.090
$S_{23} = \{v_1, v_2, v_4\}$	$\{v_3, v_5, v_6\}$	203 / 88	2.307
$S_{24} = \{v_1, v_2, v_5\}$	$\{v_3, v_4\}$	56 / 154	0.364
$S_{25} = \{v_1, v_2, v_6\}$	$\{v_3, v_4\}$	56 / 158	0.354

$S_{26} = \{v_1, v_3, v_4\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{27} = \{v_1, v_3, v_5\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 135	0.985
$S_{28} = \{v_1, v_3, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 139	0.957
$S_{28} = \{v_1, v_4, v_5\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 101	2.119
$S_{30} = \{v_1, v_4, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 105	2.038
$S_{31} = \{v_1, v_5, v_6\}$	$\{v_3, v_4\}$	56 / 171	0.327
$S_{32} = \{v_2, v_3, v_4\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{33} = \{v_2, v_3, v_5\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 186	0.715
$S_{34} = \{v_2, v_3, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 190	0.700
$S_{35} = \{v_2, v_4, v_5\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 152	1.408
$S_{36} = \{v_2, v_4, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 156	1.372
$S_{37} = \{v_2, v_5, v_6\}$	$\{v_3, v_4\}$	56 / 222	0.252
$S_{38} = \{v_3, v_4, v_5\}$	$\{v_1, v_2, v_4, v_3, v_5, v_6\}$	-	-
$S_{39} = \{v_3, v_4, v_6\}$	$\{v_1, v_2, v_4, v_3, v_5, v_6\}$	-	-
$S_{40} = \{v_3, v_5, v_6\}$	$\{v_1, v_2, v_4\}$	88 / 203	0.433
$S_{41} = \{v_4, v_5, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 169	1.266
$S_{42} = \{v_1, v_2, v_3, v_4\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{43} = \{v_1, v_2, v_3, v_5\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 199	0.668
$S_{44} = \{v_1, v_2, v_3, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 203	0.655
$S_{45} = \{v_1, v_2, v_4, v_5\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 165	1.297
$S_{46} = \{v_1, v_2, v_4, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 169	1.266
$S_{47} = \{v_1, v_2, v_5, v_6\}$	$\{v_3, v_4\}$	56 / 235	0.238
$S_{48} = \{v_1, v_3, v_4, v_5\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{49} = \{v_1, v_3, v_4, v_6\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{50} = \{v_1, v_3, v_5, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 216	0.616
$S_{51} = \{v_1, v_4, v_5, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 182	1.176
$S_{52} = \{v_2, v_3, v_4, v_5\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{53} = \{v_2, v_3, v_4, v_6\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{54} = \{v_2, v_3, v_5, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 267	0.498
$S_{55} = \{v_2, v_4, v_5, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 233	0.918
$S_{56} = \{v_3, v_4, v_5, v_6\}$	$\{v_1, v_2, v_4, v_3, v_5, v_6\}$	-	-
$S_{57} = \{v_1, v_2, v_3, v_4, v_5\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{58} = \{v_1, v_2, v_3, v_4, v_6\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{59} = \{v_1, v_2, v_3, v_5, v_6\}$	$\{v_3, v_1, v_2, v_4\}$	133 / 280	0.475
$S_{60} = \{v_1, v_2, v_4, v_5, v_6\}$	$\{v_3, v_5, v_6, v_4\}$	214 / 246	0.870
$S_{61} = \{v_1, v_2, v_4, v_5, v_6\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-
$S_{62} = \{v_2, 3, v_4, v_5, v_6\}$	$\{v_3, v_1, v_2, v_4, v_5, v_6\}$	-	-

G çizgesi için tüm seçim kümeleri ele alındığında Tablo1'deki veriler elde edilir. $bind_{vw}$ değerlerinden minimum olan değer ortalama bağlama sayısı olarak seçilir.

$$bind_{vw}(G) = \min_{S \in F(G)} \left\{ \begin{array}{c} 3.462, 0.703, 1.956, 18.455, 0.143, 0.136, 0.584, \\ 2.293, 8.458, 0.622, 0.596, 1.220, 2.707, 0.397, \\ 0.386, \emptyset, 0.721, 0.698, 2.432, 2.326, \mathbf{0.070}, 1.090, \\ 2.307, 0.364, 0.354, \emptyset, 0.985, 0.957, 2.119, 2.038, \\ 0.327, \emptyset, 0.715, 0.700, 1.408, 1.372, 0.252, \emptyset, \emptyset, \\ 0.433, 1.266, \emptyset, 0.668, 0.655, 1.297, 1.266, 0.238, \\ \emptyset, \emptyset, 0.616, 1.176, \emptyset, \emptyset, 0.498, 0.918, \emptyset, \emptyset, \\ \emptyset, 0.475, 0.870, \emptyset, \emptyset \end{array} \right\}$$

$$= 0.070$$

Bulunan değerler içinden S_{21}'nolu iterasyonun en küçük değer 0,070 olduğu görülmektedir.

5. Tartışma ve Sonuç

Karşılaşılan problemler için bu problemleri çözebilmek ya da çözüm önerileri getirilmesi gerekmektedir. Günümüzde en önemli sorunlarından biri bu problemleri modelleyebilmektir. Çizge teori yardımıyla bu problemler modellenebilmektedir. Zedelenebilirlik parametreleri ile sorunlarla karşılaşılmadan önce modelin güvenilirliğiyle ilgili bilgi elde edilebilmektedir.

Zedelenebilirlik parametreleriyle ilgili olarak yaptığımız yeni parametre vasıtasıyla, çizgeler için ayırt edici bir parametre elde edilmiştir. Yeni tanımladığımız ağırlıklı bağlama sayısı, ağırlıklı ve ağırlıksız her çizge için değişken değerler göstermektedir.

Ağırlıklı bağlama sayısını hesaplayan algoritma ve pseudo kodu oluşturulmuştur. Ağırlıklı bağlama sayısı, Phyton dilinde detaylıca yazılmış ve birçok çizge için doğru çalıştığı görülmüştür. Ağırlıklı bağlama sayısının ayırt edilmesi için minimum olması gerekmektedir.

Algoritma bölümünde açıklandığı gibi, algoritmamızda tek bir seçim iterasyonundan oluşan karmaşıklık $O(n^2)'$ dir. Var olan her seçim durumu stratejisi dahil edildiğinde elde edilen karmaşıklık değerimiz $O(2^n)'$ dir. Buradan algoritma için genel karmaşıklığımız $O(2^n)$ olacaktır.

Avantaj: Elde edilen sonuçlardan; ağırlıklı bağlama sayısının her durumda aynı çizgeler için bile ayırt edici ve daha hassas bir parametre olduğu görülmektedir. Ağırlık kavramı ile tüm durumlar detaylıca incelenebilir. Böylece oluşturduğumuz parametrenin yapılan çalışmalar doğrultusunda iyi sonuçlar verdiği görülmektedir.

Dezavantantaj: Algoritmamızın, çok tepeli çizgeler için sonuçlar vermesi uzun zaman almaktadır.

Etik Beyanı

*Bu çalışma birinci yazarın yüksek lisans tezinden üretilmiştir.

Bu çalışmada, "Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesi" kapsamında uyulması gerekli tüm kurallara uyulduğunu, bahsi geçen yönergenin "Bilimsel Araştırma ve Yayın Etiğine Aykırı Eylemler" başlığı altında belirtilen eylemlerden hiçbirinin gerçekleştirilmediğini taahhüt ederiz.

Kaynakça

[1] Barefoot, C. A., Entringer, R., Swart, H.C. 1987. Vulnerability in graphs- a comparative survey. J. Combin. Math. Combin. Comput., 1: 13–22.

[2] Harary, F. Graph Theory. (1969). Addison-Wesley Publishing, 1-274 s.

[3] Jung, H.A. 1978. On a class of posets and the corresponding comparability graphs. J. Combin. Theory Ser. B 24, 125–133.

[4] Aslan, E. 2014. A Measure of Graphs Vulnerability: Edge Scattering Number. Bull.Soc.Math.Banja Luka, 4, 53 – 60.

[5] Aslan, E., Kürkçü, Ö. 2015. Edge Scattering Number of Gear Graphs. Bull.Soc.Math.Banja Luka, 5, 25-31.

[6] Woodall, D. R. 1973. The binding number of a graph and its Anderson number. J. Combinatorial Theory Ser. B 15 225–255.

[7] Li, F., Li., X. 2007. The neighbour-scattering number can be computed in polynomial time for interval graphs. Computers & Mathematics with Applications, 54 (5), 679-686.

[8] Ray, S., Kannan, R., Zhang, D., Jiang H. 2006. The weighted integrity problem is polynomial for interval graphs, Ars Combin., 79, 77-95.

[9] Arumugam, S., Velammal, S. 1998. Edge Domination in Graphs. Taiwanese Journal of Mathematics, 2(2) 173-179.

[10] Aytaç A., Turacı, T., Odabaş, Z.N. 2013. On the bondage number of middle graphs. Mathematical Notes, 93, 95–101.

[11] Aytaç A., Turacı, T., Odabaş, Z.N. 2016. Bondage and Strong-Weak Bondage Numbers of Transformation Graphs Gxyz, Inter. Jour.of Pure and App. Math., 106,(2), 689–698.

[12] Bagga, K.S., Beineke, L.W., Lipman M.J., Pippert, R.E. 1994. Edge-integrity: a survey. Discrete Mathematics, 124, 3-12.

[13] Kaval, B., Kırlangıç, A. 2024. Domination Scattering Number in Graphs. Journal of New Theory, 49, 53 – 61.

[14] Tokat, H., Kırlangıç, A. 2019. On the Domination Integrity. International Journal of Foundations of Computer Science, 30(5), 811-826.

[15] Aslan, E. Tosun, M.A., 2021. Computing the weighted neighbor isolated tenacity of interval graphs in polynomial time. Numerical Methods for Partial Differential Equations, 37(3), 2540-2549.

[16] Durgut, R., Turacı, T., Kutucu, H. 2019. A heuristic algorithm to find rupture degree in graphs. Turkish Journal of Electrical Engineering and Computer Sciences, 7: 3433 – 3441.

[17] Golumbic, M.C. 1980. Algorithmic Graph Theory and Perfect Graphs. Computer Science and Applied Mathematics Academic Press, New York.

[18] Li, F., Zhang, X., Broesmersma, H. 2019. A polynomial algorithm for weighted scattering number in interval graphs. Discrete Applied Mathematics, 264, 118-124.