Konya Mühendislik Bilimleri Dergisi, c. 11, s. 1, 150-159, 2023
Konya Journal of Engineering Sciences, v. 11, n. 1, 150-159, 2023
ISSN: 2667-8055 (Elektronik)
DOI: 10.36306/konjes.1101410

KOMŞU İZOLE SAÇILMA SAYISININ ALGORİTMASI

[1]Mehmet Aykut TOSUN, [2]Ersin ASLAN, [3]Emin BORANDAĞ

[1]Niğde Ömer Halisdemir Üniversitesi, Uzaktan Eğitim Uygulama ve Araştırma Merkezi, Niğde, TÜRKİYE
[2,3]Manisa Celal Bayar Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, Manisa, TÜRKİYE
[1]aykuttosun@ohu.edu.tr, [2]ersin.aslan@cbu.edu.tr, [3]emin.borandag@cbu.edu.tr

Önemli Katkılar (Highlights)

- Ağların graflarla modellenebilirliğinden bahsedilerek, graflarda zedelenebilirlik yardımıyla da ağlarda güvenliğin sağlanabileceği konusu üzerinde durulmuştur
- Komşu ve izole kavramlarının önemi vurgulanarak komşu izole saçılma sayısının çalışmada kullanım amacı ortaya konulmuştur.
- Zedenelebilirlik ölçüm parametrelerinin yaptığı ölçümü hesaplayan algoritma çalışmalarının literatürdeki örneklerinden bahsedilmiş, algoritma ile hesaplama yapıldığında insan işgücünden zaman ve çaba konusunda tasarruf sağlanacağının önemi vurgulanmış ve komşu izole saçılma sayısını hesaplayan algoritma literatürde ilk kez verilmiştir.
- Verilen algoritma çeşitli yazılım metrikleri ile analiz edilmiş, algoritma karmaşıklığı hesaplanmış ve kullanılabilir bir algoritma olduğu gösterilmiştir.
- Verilmiş olan komşu izole saçılma sayısını hesaplayan algoritmanın ürettiği hesaplama sonuçları ile gerçek komşu izole saçılma sayısı hesaplama sonuçları tablolarla karşılaştırılarak ölçümlerin doğruluğu gösterilmiştir.

***Corresponding Author:** Mehmet Aykut TOSUN, aykuttosun@ohu.edu.tr

Konya Mühendislik Bilimleri Dergisi, c. 11, s. 1, 150-159, 2023
Konya Journal of Engineering Sciences, v. 11, n. 1, 150-159, 2023
ISSN: 2667-8055 (Elektronik)
DOI: 10.36306/konjes.1101410

KOMŞU İZOLE SAÇILMA SAYISININ ALGORİTMASI

[1]Mehmet Aykut TOSUN, [2]Ersin ASLAN, [3]Emin BORANDAĞ

[1]Niğde Ömer Halisdemir Üniversitesi, Uzaktan Eğitim Uygulama ve Araştırma Merkezi, Niğde, TÜRKİYE
[2,3]Manisa Celal Bayar Üniversitesi, Teknoloji Fakültesi, Yazılım Mühendisliği Bölümü, Manisa, TÜRKİYE
[1]aykuttosun@ohu.edu.tr, [2]ersin.aslan@cbu.edu.tr, [3]emin.borandag@cbu.edu.tr

(Geliş/Received: 12.04.2022; Kabul/Accepted in Revised Form: 11.12.2022)

ÖZ: Ağ güvenliği ve güvenilirliği, bilgisayar ağlarının önemli bir parçasıdır. Bilgisayar korsanları nedeniyle ağ güvenliği iyileştirilmek zorundadır. İş sürekliliği, iyileştirme için başka bir nedendir. Ağlar graflarla modellenebilir. Grafların ve dolayısıyla ağların zedelenebilirliğini ölçmek için çeşitli parametreler mevcuttur. Bu makalede, komşu izole saçılma sayısı ele alınmış ve önerilen zedelenebilirlik ölçüm parametresinin herhangi bir graf için ölçülmesini önerdiğimiz bir algoritma geliştirilmiş ve algoritma yazılım kod metrikleri ile analiz edilmiş ve faydalı olduğu gösterilmiştir. Böylelikle herhangi bir graf için zedelenebilirlik ölçümü yaparken algoritma kullanarak insan işgücünden tasarruf sağlanacağı sonucuna varılmıştır.

Anahtar Kelimeler: *Graf Teori, Algoritma, Ağ Tasarımı, Zedelenebilirlik, Komşu İzole Saçılma Sayısı (NIS)*

Algorithm of Neighbor Isolated Scattering Number

ABSTRACT: Network security and reliability is an essential part of computer networks. Network security has had to improve due to hackers. Business continuity is another reason for improvement. Networks can be modeled with graphs. Various parameters exist to measure the vulnerability of graphs, and hence of networks. In this paper, we consider neighbor isolated scattering number and an algorithm has been developed for the proposed vulnerability measurement parameter, which we recommend to measure for any graph and the algorithm is analyzed by software code metrics and have been shown to be useful. Thus, it has been concluded that humanpower will be saved by using an algorithm while measuring vulnerability for any graph.

Keywords: *Graph Theory, Algorithm, Network Design, Vulnerability, Neighbor Isolated Scattering Number (NIS)*

1. GİRİŞ (INTRODUCTION)

Herhangi bir ağ, merkezleri ve aralarında bağlantıları olan ilişkisel bir yapıdır. Bu yapıda merkezler tepelerle (düğümlerle) ve bağlantılar ayrıtlarla (kenarlarla) ifade edildiğinde bir graf modeli oluşturulur. Ağlar graflarla modellendiğinde tasarlanırken zedelenebilirlikleri ölçülebilir ve ayrıca maliyeti hesaplanabilir. Ağ, henüz tasarım aşamasındayken güvenlik açıklarını öngörmek ve önlemek, bunu yanında daha ekonomik ve güvenli ağlar tasarlamak için önerilmiş bağlanırlık (connectivity) [1], mukavemet (tenacity) [2] ve kopma derecesi (rupture degree) [3] gibi zedelenebilirlik ölçüm parametreleri bulunmaktadır.

Bir ağın merkezleri veya bağlantıları fiziksel veya siber yolla hasar aldığında iletişim kesintiye uğrar. Zedelenebilirlik ölçümü, bu iletişim kesintisine kadar ağın direncini ifade eder. Bu ölçümler, bir ağın ne kadar hassas veya dayanıklı olduğuna dair fikir verir. Saçılma sayısı [4], bu zedelenebilirlik ölçüm parametreleri için bir örnektir.

***Corresponding Author:** Mehmet Aykut TOSUN, aykuttosun@ohu.edu.tr

Birçok parametre türü, etkilenen ağ merkezlerinin komşularını yok sayar. Casus ağlarda bir merkez deşifre olduğunda, o merkezin komşuları da deşifre olur. Bu da iletişimin kesilmesine neden olur [5]–[7]. Bu nedenle komşuluk da dikkate alınmalıdır. Komşuluk kavramını kullanan parametrelerden biri komşu izole saçılma sayısıdır [8]. Komşu izole saçılma sayısının, [8] çalışmasında kullanılabilirliği ve ayırt edici sonuçlar verdiği gösterilmiştir. [8]'te komşu izole saçılma sayısı için matematiksel açıklamalar, teoremler ve bunların detayları verilmiştir.

Literatürde saçılma sayısı tabanlı, ancak izole ve komşuluk kavramlarını tek başına kullanan zedelenebilirlik parametrelerinin algoritmaları önerilmiştir. Bunlar, izole saçılma sayısını ölçen algoritmalar [9], [10] ve komşu saçılma sayısını ölçen algoritma [11] çalışmalarıdır. Biz, bu çalışmada komşuluk ve izole kavramlarını bir arada kullanan komşu izole saçılma sayısını ölçen algoritmayı ilk kez öneriyoruz. Bu çalışma, [8] çalışmasından farklı olarak komuşu izole saçılma sayısını ölçen bir algoritmayı ve bu algoritmanın analizini içermektedir.

Bu çalışmadaki amacımız, komşu izole saçılma sayısının algoritmasını oluşturmak ve insan gücünü azaltmaktır. Zedelenebilirlik ölçüm parametreleri çalışmalarında verilen tanımlar grafların zedelenebilirliğini ölçerken insan gücünü kullanmayı azaltıyor olsa da standart bir özel graf sınıfına girmeyen grafların algoritma kullanılarak hesaplanması insan gücünü daha çok azaltır. Bu algoritmanın kullanışlılığı yazılım kod metrik analiz sonuçlarıyla gösterilmiştir. Geliştirilen algoritma, Python programlama dili ile test edilmiştir. Bu bağlamda kod metrikleriyle analiz gerçekleştirmek için Radon [12] kütüphanesinden yararlanılmıştır.

Radon, Halstead metriklerini, ham metrikleri, sürdürülebilirlik endeksini ve döngüsel karmaşıklığı hesaplayabilir [12]. Döngüsel Karmaşıklık, bir kod bloğunun artı 1 içerdiği kararların sayısına karşılık gelir. Kod boyunca doğrusal olarak bağımsız ayrıtların sayısıyla bu sayı (ayrıca McCabe numarası olarak da adlandırılır) eşittir. Bu sayı, bloklarda koşullu mantığı (döngü, if, boolean gibi mantıksal ifadeler) test ederken kılavuz olarak kullanılabilir. Sürdürülebilirlik Endeksi, farklı metriklerin bazı değerlerinin kullanılmasıyla hesaplanır. Ham metrikler, diğer kod metriklerinin hesaplamak adına kullandığı kod içerindeki satır sayısını, yorum satırı sayısını, çok satırlı dizeleri, boş satır sayısını vb. ifade eder. Kodun ölçülebilir özelliklerini ve aralarındaki ilişkileri tanımak Halstead'in amacıdır [13]–[16]. Radon kütüphanesinin sonuçlarını verdiği bu yazılım metriklerinin detayları yazının sonraki bölümünde verilmiştir.

2. YAZILIM METRİKLERİ (SOFTWARE METRICS)

Yazılım metrikleri, yazılımda gerçekleştirilen ölçümlere dayalı olarak ölçülebilen veya hesaplanabilen değerlerdir. Beş ana yazılım metriği boyut, kalite, çaba, maliyet ve süredir [17]. Yazılım geliştirme sürecinde kaydedilen metrikler, hataların tespit edilmesine, yazılımın izlenmesine ve kalitesinin artırılmasına yardımcı olur. Yazılımda çok çeşitli bilgiler bulunduğundan farklı metrikler de geliştirilebilir. En iyi bilinen metriklerden biri, genellikle kaynak kodun basit satır sayısı olarak tanımlanan kodun boyutudur [18]. Chidamber Kemerer (CK) tarafından tanımlanan Nesne Yönelimli metrikler de yaygın olarak kullanılan metriklerdir. Ayrıca Halstead'in karmaşıklık metriği [19] ve McCabe'in Döngüsel Karmaşıklık metriği (CC) gibi bazı metrikler de geliştirilmiş ve yaygın olarak kullanılmaktadır. CC, (1) denkleminde verilen formülle hesaplanır [20].

$$CC = E - N + 2P \tag{1}$$

Burada E, yazılımın kontrol akış grafının ayrıt sayısını, N tepe sayısını ve P bu graftaki bağlı bileşenlerin sayısını temsil eder. CC için belirlenmiş olan ideal sınırlar şöyledir; 1-10 aralığı düşük risk ve basit karmaşık, 11-20 aralığı orta risk ve daha karmaşık, 21-50 aralığı yüksek risk ve karmaşık ve 50 üzeri çok yüksek risk ve test edilemez kod [20]–[22].

Yorum Oranı, (2) denkleminde görüldüğü gibi yorum ifadesinin sayısının toplam ifade sayısına bölünmesiyle hesaplanır.

$$Yorum\ Oranı\ =\ Yorum\ ifadesi/Toplam\ ifade \tag{2}$$

Ayrıca, literatürde Ortalama Derinlik ve Maksimum Derinlik gibi metrikler de tanımlanmıştır [22], [23]. Ortalama Derinlik, koddaki her bir ifadenin blok derinliklerinin toplanması ve ardından bu toplamın (2) denkleminde verilen toplam ifadeye bölünmesiyle hesaplanır. Maksimum Derinlik, en derin iç içe ifadenin blok derinliğidir.

Halstead Metrikleri: Halstead metrikleri, yazılımın ölçülebilir özelliklerini tanımlamak için kullanılır. Bu metrikler statik olarak kaynak koddan hesaplanır. (3) denkleminde Halstead program uzunluğu ($\widehat{N}$) hesaplaması gösterilmiştir.

$$\widehat{N} = \eta_1\, log_2\, \eta_1 + \eta_1\, log_2\, \eta_2 \tag{3}$$

η_1, farklı operatörlerin sayısını, η_2, farklı işlenenlerin sayısını temsil etmektedir. Bu sayılardan çeşitli yazılım ölçüleri hesaplanabilir. Programlama dillerinde sabit veya değişkenleri işleyen karakterlere operatör (operator), bu operatörlerin işlediği sabit veya değişkenlere ise işlenen (operand) denilmektedir.

Çizelge 1. Bazı yazılım metriklerinin hesaplanması
Table 1*. Calculations of Some Software Metrics*

Program sözlüğü	$\eta = \eta_1 + \eta_2$
Program uzunluğu	$N = N_1 + N_2$
Hesaplanan program uzunluğu	$\widehat{N} = \eta_1\, log_2\, \eta_1 + \eta_2\, log_2\, \eta_2$
Hacim	$V = N \log_2 \eta$
Zorluk	$D = (\eta_1/2) * (N_2/\eta_2)$
Çaba	$E = D * V$
Buglar	$B = V/3000$

N_1, toplam operatör sayısını, N_2, toplam işlenen sayısını ifade eder. Verilen bu metriklerin ideal sınırları Halstead tarafından [19] çalışmasında belirtilmiştir.

Sürdürülebilirlik endeksi, (4) denklemindeki gibi hesaplanır [12], [15], [16].

$$Sürdürülebilirlik\ Endeksi = \max\left[0{,}100 * \frac{171 - 5.2 \ln V - 0.23G - 16.2 \ln L - 50 \sin\sqrt{2.4C}}{171}\right] \tag{4}$$

Burada V Halstead hacmini, G döngüsel karmaşıklığı, L kodun kaynak satır sayısını (SLOC), C ise yorum satırlarının radyana dönüştürülmüş yüzdesini ifade eder. Çıkan sonuç kötüden iyiye şu şekilde yorumlanır; 0-9 aralığı C (kötü), 10-19 aralığı B (orta) ve 20-100 aralığı A (iyi) [12], [15], [16].

3. KOMŞU İZOLE SAÇILMA SAYISI (NEIGHBOR ISOLATED SCATTERING NUMBER)

Bu çalışmada komşuluk parametrelerinden biri olan komşu izole saçılma sayısı kullanılmıştır ve komşu izole saçılma sayısının tanımı şu şekilde verilmektedir: Bir G grafı için, komşu izole saçılma sayısı

$$NIS(G) = max\{\, i(G/X) - |X| : i(G/X) \geq 1\} \tag{5}$$

olarak tanımlanır. X, G grafının kesme stratejisidir, maksimum tüm kesme stratejileri için alınır ve $i(G/X)$, G/X grafında izole tepelerin sayısıdır. G grafının tüm tepelerini içeren küme $V(G)$ olarak ifade edilir. $NIS(G) = i(G/X) - |X|$ ise, bir $X \subset V(G)$ kümesine G grafının *NIS-kümesi* olduğu söylenir [8].

Komşu izole saçılma sayısı, bir ağ zarar görmeden önce karşılaşacağı hasarı ölçer, böylece ağa gelecek zarar önceden görülebilir. Kullanılması planlanan iki graftan birinin büyük bir komşu izole saçılma sayısı

varsa, graf daha savunmasızdır. Bu yüzden küçük değere sahip olan grafın kullanılması ağ için daha güvenilir olacaktır. Komşu izole saçılma sayısı ile ilgili detaylar Aslan tarafından [8]'te verilmiştir.

3.1. Bir Grafın Komşu İzole Saçılma Sayısını Hesaplayan Algoritma (Algorithm for Calculating the Neighboring Isolated Scattering Number of a Graph)

Bu kısımda bir grafın komşu izole saçılma sayısını hesaplayan algoritma verilmiştir. Şekil 1'deki algoritmada görüldüğü gibi temel birkaç işlemin de yürütülebilmesi için bazı metotlar kullanılmıştır. Bunlar Python için geliştirilmiş NetworkX [24] kütüphanesi içinde kullanılan graf yapısındaki metotlar ve arayüzlerdir. İzole tepe sayısını belirleyen IsolatedCount ve komşularıyla birlikte belirtilen tepeyi graftan kaldıran NeighborRemoveVertex olmak üzere iki yardımcı metot hesaplama için hazırlanmıştır.

Şekil 1'deki algoritma, graflar için komşu izole saçılma sayısını hesaplayabilir.

4. ALGORİTMA ANALİZİ VE SONUÇLARI (ALGORITHM ANALYSIS AND RESULTS)

Verilen algoritma ve yardımcı metotlar Python'da geliştirilmiş ve test edilmiştir. Test sonucunda yönsüz ve ağırlıksız graflar için, [8] çalışmasında bulunan "Bazı Özel Graf Sınıflarının Komşu İzole Saçılma Sayısı" bölümündeki teoremlere göre doğru sonuç verdiği görülmüştür.

Verilen algoritmanın karmaşıklığının hesaplanması (6), (7), (8), (9) ve (10) denklemlerinde gösterilmiş ve $O(2^n)$ olarak bulunmuştur. $T(0)$, karmaşıklıkta ilk ve en kısa adımdır ve $T(0) = 1$'dir.

$$T(n) = T(n-1) + T(n-2) + \cdots + T(0) \tag{6}$$

$$T(n-1) = T(n-2) + T(n-3) + \cdots + T(0) \tag{7}$$

$$T(n) = T(n-1) + T(n-1) = 2 * T(n-1) = 2 * 2 * T(n-2) \tag{8}$$

$$T(n) = 2^n * T(n-n) = 2^n * T(0) = 2^n * 1 \tag{9}$$

$$T(n) = O(2^n) \tag{10}$$

Radon kütüphanesi yardımıyla elde edilen yazılım metrikleri analiz sonuçları Çizelge 2-5 üzerinde görülmektedir. Algoritma tarafından hesaplanan bazı zedelenebilirlik ölçüm sonuçları da Çizelge 6-9 üzerinde görülmektedir.

Çizelge 2'ye göre NIS algoritması 44 satırdan (LOC) oluşur. Mantıksal satırların sayısı (LLOC) 34'tür. Yorum satırlarının LLOC'ye oranının %20'nin üzerinde olması, programın yorum satırlarıyla iyi anlatıldığı anlamına gelir. SLOC, sürdürülebilirlik endeksi tanımında da görüldüğü gibi sürdürülebilirlik endeksi hesaplamasında yardımcı olur. C%L, yorum satırı sayısı ile LOC arasındaki oranı, C%S, yorum satırı sayısı ile SLOC arasındaki oranı, C+M%L ise yorum ve çok satırlı dize satırlarının sayısı ile LOC arasındaki oranı ifade eder.

```
Girdi: G graf,
       L graftaki toplam tepe sayısı olan özyineleme sayısı,
       S etkilenen tepe sayısı.
Çıktı: G grafının komşu izole saçılma sayısı
if L = 0 then
    # Eğer özyineleme sayısı 0 ise G grafındaki izole tepe sayısını döndür
    return IsolatedCount(G)
esle if L = 1 then
    # Eğer özyineleme sayısı 1 ise G grafının NIS değerini döndür
    return IsolatedCount(G)−S
else
    # Maksimum NIS'i minimum integer olarak tanımla
    maxNIS ← minInt
    # G grafının tepelerinin listesi
    vertices ← G.nodes
    for all v in vertices do
        # Geçici graf olarak G grafının kopyasını al
        tempG ← copy.deepcopy(G)
        # v tepesini komşularıyla birlikte çıkar
        tempG. NeighborRemoveVertex(v)
        # Geçici graf için NIS'i hesapla
        tempNIS1 ← IsolatedCount(tempG)−S
        # Geçici graftan bir tepe daha silerek NIS'i hesapla
        tempNIS2 ← NIS(tempG, L − 1, S + 1)
        # Geçici NIS'leri maksimum NIS ile karşılaştır
        if tempNIS1>maxNIS then
            maxNIS ← tempNIS1
        end if
        if tempNIS2>maxNIS then
            maxNIS ← tempNIS2
        end if
        tempG.clear()
    end for
    # Hesaplanan NIS'i döndür
    return maxNIS
end if
```

Şekil 1. Bir G grafının Komşu izole saçılma sayısını veren sözde kod algoritması (NIS)
Figure 1. *Algorithm in pseudocode that gives Neighbor Isolated Scattering Number of a graph G (NIS)*

Çizelge 2. NIS'in ham metrikleri
Table 2*. Raw Metrics of NIS*

Ham Metrikler	***NIS***
LOC	44
LLOC	34
SLOC	34
Yorumlar	15
Tekil yorumlar	7
Çoklu yorumlar	0
Boş satır	3
Yorum istatistikleri	
(C%L)	34%
(C%S)	44%
(C+M%L)	34%

Çizelge 3. NIS'in Halstead metrikleri
Table 3*. Halstead Metrics of NIS*

Halstead Metrikler	NIS
η_1	5
η_2	14
N_1	11
N_2	22
Kelime Bilgisi	19
Uzunluk	33
Hesaplanan Uzunluk	64.91
Hacim	140.18
Zorluk	3.93
Çaba	550.71
Zaman	30.60
Buglar	0.05

Halstead metrikleri, fonksiyonel yazılım karmaşıklığını ölçmek için kullanılabilir. Bu doğrultuda Çizelge 3'e göre, Çizelge 1'de verilen program uzunluğu formülünden hareketle $N_1 + N_2$ 33 olarak hesaplanır. İdeal üst limit 300'ün altındadır. Program hacim değeri 140 olarak hesaplanmıştır. Hacim için ideal üst limit değeri 1000 olduğundan algoritma karmaşık bir yapıya sahip değildir. Çaba, bir kalite standardı olarak görülür ve algoritma, sınır değeri 5000'in altında sonuç vermiştir. Bu da programın metrikleri açısından bir başka artı olarak görünmektedir. Aynı şekilde zaman değerinin 90'ın altında olması da programın kalitesini gösteren bir diğer değerdir. Bug değeri 0'a çok yakındır, tahmini hata sayısını ifade eder, bu programda çok düşük bir hata olasılığına sahip olduğunu gösterir.

Çizelge 4. NIS'in sürdürülebilirlik endeksi
Table 4*. Maintainability Index Metrics of NIS*

Algoritma	Sürdürülebilirlik Endeksi
NIS	A

Çizelge 4'e göre Sürdürülebilirlik Endeksi, programın daha sonra güncellenebilirliğini gösterir. Geliştirilen algoritma A ile en yüksek değeri almıştır. Bu, algoritmanın gerekli koşullar için güncellenebilir olduğunu göstermiştir.

Çizelge 5. NIS'in döngüsel karmaşıklığı
Table 5*. Cyclomatic Complexity of NIS*

Algoritma	CC
NIS	16
IsolatedCount	4
NeighborRemoveVertex	11

Çizelge 5, geliştirilen algoritmada, IsolatedCount oldukça küçük bir değer almıştır ve düşük risk sınıfındadır. NIS ve NeighborRemoveVertex, 10'un üzerinde bir değere sahip oldukları için orta risk ve daha karmaşık sınıfta yer almaktadır. Döngüsel karmaşıklık için üst sınır 50 olduğu için programın döngüsel karmaşıklık oluşturmadığı görülmektedir.

Algoritma, yol, çember, tekerlek ve tam graflar gibi çeşitli graf sınıflarında test edilmiştir. Çizelge 6-9, bu graflar için komşu izole saçılma sayısı değerlerini içerir. Ayrıca Çizelge 6-9 üzerinde bulunan $s(V(G))$ ifadesi, bir G grafındaki toplam tepe sayısını gösterir.

Çizelge 6. P_n yol grafı için sonuçlar
Table 6. *Results for P_n path graph*

n	$s(V(P_n))$	$NIS(P_n)$	Süre (sn)
5	5	1	0.0018
6	6	0	0.0044
7	7	0	0.0091
8	8	0	0.022
9	9	1	0.0546
10	10	0	0.1315
11	11	0	0.4395
12	12	0	0.9855
13	13	1	2.9267

Farklı n tepe sayıları için her bir P_n yol grafının komşu izole saçılma sayısı, geliştirilen algoritma ile hesaplanmış ve hesaplanan komşu izole saçılma sayıları ve bu sayıların ne kadar sürede hesaplandığı Çizelge 6 içerisinde gösterilmiştir.

Çizelge 7. C_n çember grafı için sonuçlar
Table 7. *Results for C_n cycle graph*

n	$s(V(C_n))$	$NIS(C_n)$	Süre (sn)
4	4	0	0.001
5	5	-1	0.0018
6	6	-1	0.0039
7	7	-1	0.0068
8	8	0	0.0201
9	9	-1	0.0325
10	10	-1	0.0841
11	11	-1	0.2811
12	12	0	0.5891
13	13	-1	1.7682
14	14	-1	4.8263
15	15	-1	14.653
16	16	0	46.092

Farklı n tepe sayıları için her bir C_n çember grafının komşu izole saçılma sayısı, geliştirilen algoritma ile hesaplanmış ve hesaplanan komşu izole saçılma sayıları ve bu sayıların ne kadar sürede hesaplandığı Çizelge 7 içerisinde gösterilmiştir.

Çizelge 8. W_n tekerlek grafı için sonuçlar
Table 8. *Results for W_n wheel graph*

n	$s(V(W_n))$	$NIS(W_n)$	Süre (sn)
5	5	0	0.0012
6	6	-1	0.0018
7	7	-1	0.003
8	8	-1	0.0071
9	9	0	0.0158
10	10	-1	0.0348
11	11	-1	0.1759
12	12	-1	0.2111

Farklı n tepe sayıları için her bir W_n tekerlek grafının komşu izole saçılma sayısı, geliştirilen algoritma ile hesaplanmış ve hesaplanan komşu izole saçılma sayıları ve bu sayıların ne kadar sürede hesaplandığı Çizelge 8 içerisinde gösterilmiştir.

Çizelge 9. K_n tam graf için sonuçlar
Table 9. *Results for K_n complete graph*

n	$s(V(K_n))$	$NIS(K_n)$	Süre (sn)
5	5	-1	0.0009
6	6	-1	0.0015
7	7	-1	0.0023
8	8	-1	0.0024
9	9	-1	0.0027
10	10	-1	0.0035
11	11	-1	0.0051

Farklı n tepe sayıları için her bir K_n tam grafının komşu izole saçılma sayısı, geliştirilen algoritma ile hesaplanmış ve hesaplanan komşu izole saçılma sayıları ve bu sayıların ne kadar sürede hesaplandığı Çizelge 9 içerisinde gösterilmiştir.

5. SONUÇ (CONCLUSION)

Bu çalışmanın amacı, insan gücünden tasarruf etmek için komşu izole saçılma sayısı algoritmasını oluşturmaktır. Zedelenebilirlik parametreleri, tanımları ve teoremleri sayesinde ölçümde insan gücünden tasarruf sağlıyor olsa da standart dışı grafların zedelenebilirliğini ölçmek için bu tanım ve teoremler yetersiz kalacağından bunların algoritmalarının oluşturulması ve geliştirilmesi insan gücünden daha fazla tasarruf sağlayacaktır. Bir graf modeli hazırlanmak istendiğinde grafın herhangi bir saldırı girişimine karşı oldukça dayanıklı olması beklenir. Bu nedenle istenilen graf modelinin zedelenebilirlik analizi komşu izole saçılma sayısı gibi parametrelerle belirlenmelidir. Grafların zedelenebilirliğinin ölçümünde yaygın olarak kullanılan parametrelerden komşu izole saçılma sayısı ele alınmıştır. Parametreyi hesaplamak için bir algoritma önerilmiş ve Radon yazılım metrikleri ile analiz edilerek ve birkaç graf türünde sonuçlarla birlikte çalışma süreleri gösterilmiştir. Sonuçlar, geliştirilen algoritmanın komşu izole saçılma sayısını doğru bir şekilde hesapladığını göstermiştir. Algoritmanın Radon yazılım metriklerinden elde edilen sonuçları ideal sınırlar içinde yer almıştır. Sonuçlara göre NIS algoritmasının komşu izole saçılma sayısını ölçmek için faydalı olduğu görülmüştür.

Etik Standartlar Bildirimi (Declaration of Ethical Standards)

Bu çalışmanın yazarları olarak tüm etik standartlara uyulduğunu bildiririz.

Yazar Katkı Beyannamesi (Credit Authorship Contribution Statement)

Bu çalışmada yazar katkı oranları Mehmet Aykut TOSUN %34, Ersin ASLAN %33 ve Emin BORANDAĞ %33 şeklindedir.

Çıkar Çatışması Beyannamesi (Declaration of Competing Interest)

Bu çalışmanın yazarları olarak herhangi bir çatışma beyanımız bulunmadığını bildiririz.

Destek / Teşekkür (Funding / Acknowledgements)

Bu çalışma Manisa Celal Bayar Üniversitesi Bilimsel Araştırma Projeleri tarafından desteklenmektedir. Proje no: 2019-071.

Veri Kullanılabilirliği (Data Availability)

Bu çalışma kullanılabilir veri içermemektedir.

KAYNAKLAR (REFERENCES)

[1] C. A. Barefoot, R. C. Entringer, ve H. C. Swart, "Vulnerability in graphs—A comparative survey", Journal of Combinatorial Mathematics and Combinatorial Computing, c. 1, ss. 13-22, 1987.

[2] M. B. Cozzens, D. Moazzami, ve S. Stueckle, "The tenacity of the Harary Graphs", J. Combin. Math. Combin. Comput., c. 16, ss. 33-56, 1994.

[3] Y. Li, S. Zhang, ve X. Li, "Rupture degree of graphs", Int J Comput Math, c. 82(7), ss. 793-803, 2005, doi: 10.1080/00207160412331336062.

[4] H. A. Jung, "On a class of posets and the corresponding comparability graphs", Journal of Combinatorial Theory, Series B, c. 24(2), ss. 125-133, 1978, doi: 10.1016/0095-8956(78)90013-8.

[5] G. Gunther ve B. L. Hartnell, "Optimal K-secure graphs", Top Catal, c. 2(3), ss. 225-231, 1980, doi: 10.1016/0166-218X(80)90042-6.

[6] Z. Wei, A. Mai, ve M. Zhai, "Vertex-neighbor-scattering number of graphs", Ars Combinatoria, c. 102, ss. 417-426, 2011.

[7] S. S. Y. Wu ve M. B. Cozzens, "The minimum size of critically m-neighbour-connected graphs", Ars Combinatoria, c. 29, ss. 149-160, 1990.

[8] E. Aslan, "Neighbour isolated scattering number of graphs", ScienceAsia, c. 41, sy 6, ss. 423-431, 2015, doi: 10.2306/scienceasia1513-1874.2015.41.423.

[9] M. Jurkiewicz, "Bounds on isolated scattering number", ANZIAM J, c. 62, ss. 72-83, 2020, doi: 10.21914/anziamj.v62i0.15912.

[10] F. Li, Q. Ye, ve Y. Sun, "Isolated Scattering Number Can be Computed in Polynomial Time for Interval Graphs", ANZIAM Journal, c. 58, sy March, s. 81, 2017, doi: 10.21914/anziamj.v58i0.10993.

[11] F. Li ve X. Li, "The neighbour-scattering number can be computed in polynomial time for interval graphs", Computers and Mathematics with Applications, c. 54, sy 5, ss. 679-686, 2007, doi: 10.1016/j.camwa.2007.02.006.

[12] "Radon Documentation". https://radon.readthedocs.io/en/latest/ (erişim 7 Nisan 2022).

[13] D. Coleman, D. Ash, B. Lowther, ve P. Oman, "Using Metrics to Evaluate Software Svstem", IEEE Computer, c. 27, sy 8, ss. 44-49, 1994.

[14] A. van Deursen, "Think Twice Before Using the 'Maintainability Index'", https://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/, 2014.

[15] "Maintainability Index Range and Meaning", Code Analysis Team Blog, blogs.msdn, 2007. https://docs.microsoft.com/en-us/archive/blogs/codeanalysis/maintainability-index-range-and-meaning (erişim 7 Nisan 2022).

[16] P. Oman ve J. Hagemeister, "Metrics for assessing a software system's maintainability", Proceedings - Conference on Software Maintenance, ICSM 1992, ss. 337-344, 1992, doi: 10.1109/ICSM.1992.242525.

[17] S. R. Schach, Object-Oriented and Classical Software Engineering. New York: NY: McGraw-Hill, 2011.

[18] A. G. Koru, K. el Emam, D. Zhang, H. Liu, ve D. Mathew, "Theory of Relative Defect Proneness", Empir Softw Eng, c. 13, sy 5, ss. 473-498, 2008, doi: 10.1007/s10664-008-9080-x.

[19] M. H. Halstead, Elements of Software Science. New York: NY, USA: Elsevier Science Inc., 1977.

[20] T. J. McCabe, "A Complexity Measure", IEEE Transactions on Software Engineering, c. 2, sy 4, ss. 308-320, 1976, doi: 10.1109/TSE.1976.233837.

[21] M. Bray vd., "C4 Software Technology Reference Guide-A Prototype", Oca. 1997.

[22] M. Shepperd, "A Critique of Cyclomatic Complexity as a Software Metric", Software engineering journal, c. 3, sy 2, ss. 30-36, 1988, doi: 10.1049/sej.1988.0003.
[23] M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort", Journal of Systems and Software, c. 70, sy 1-2, ss. 37-60, 2004, doi: 10.1016/S0164-1212(02)00156-5.
[24] "NetworkX Offical Website". http://networkx.github.io (erişim 7 Nisan 2022).