Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 3433 – 3441
© TÜBİTAK
doi:10.3906/elk-1903-29

A heuristic algorithm to find rupture degree in graphs

Rafet DURGUT[1], Tufan TURACI[2], Hakan KUTUCU[1]*
[1]Department of Computer Engineering, Karabuk University, Karabuk, Turkey
[2]Department of Mathematics, Karabuk University, Karabuk, Turkey

Received: 05.03.2019 • **Accepted/Published Online:** 24.05.2019 • **Final Version:** 18.09.2019

Abstract: Since the problem of Konigsberg bridge was released in 1735, there have been many applications of graph theory in mathematics, physics, biology, computer science, and several fields of engineering. In particular, all communication networks can be modeled by graphs. The vulnerability is a concept that represents the reluctance of a network to disruptions in communication after a deterioration of some processors or communication links. Furthermore, the vulnerability values can be computed with many graph theoretical parameters. The rupture degree $r(G)$ of a graph $G = (V, E)$ is an important graph vulnerability parameter and defined as $r(G) = max\{\omega(G - S) - |S| - m(G - S) : \omega(G - S) \geq 2, S \subset V\}$, where $\omega(G - S)$ and $m(G - S)$ denote the number of connected components and the size of the largest connected component in the graph $G - S$, respectively. Recently, it has been proved that finding the rupture degree problem is NP-complete. In this paper, a heuristic algorithm to determine the rupture degree of a graph has been developed. Extensive computational experience on 88 randomly generated graphs ranging from 20% to 90% densities and from 100 to 200 vertices shows that the proposed algorithm is very effective.

Key words: Network design and communication, graph vulnerability, connectivity, rupture degree, heuristic algorithm

1. Introduction

Networks define a wide range of systems in various science including computer science, mathematics, chemistry, social sciences, informatics. Furthermore, designing a robust network is important for network designers because everyone wants a speedy, reliable, and nonstop communication. This situation increases the importance of the communication links that we have to construct. Graph theory has become one of the most robust mathematical tools in the analysis of the architecture of a network. The vulnerability is a concept that measures the reliability and stability of a network. Moreover, the vulnerability value of a communication network is the resistance of this communication network until certain stations or communication links between these stations are disrupted and thus communication interrupts occur [1].

A network is simply defined as an undirected connected graph without self-loops or multiple edges. In the literature, many graph parameters have been defined to measure the robustness of networks. [2, 3]. The first and the best-known parameter of the vulnerability of a graph is its connectivity number. The vertex (edge) connectivity is the minimum number of vertices (edges) whose deletion results in a disconnected or trivial graph [4]. Then toughness [5], integrity [6], scattering number [7], rupture degree [8], etc. have been defined for measuring the vulnerability of a network. The remainder of this section is devoted to more definitions and notation that will be used throughout the paper.

*Correspondence: hakankutucu@karabuk.edu.tr

Let $G = (V, E)$ be a simple undirected graph with n nodes and m edges. Let v be any vertex in G. The *open neighborhood* of v is defined to $N(v) = \{u \in V | uv \in E\}$ and *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. Furthermore, let $S \subseteq V$. The open neighborhood of S is the set $N(S) = \bigcup_{v \in S} N(v)$. Furthermore, the closed neighborhood of S is the set $N[S] = N(S) \cup S$. The *degree of vertex* v, denoted by $deg(v)$, in G is the number of the vertex v, that is the size of its open neighborhood. Let u and v be any two vertices of V. The *distance* $d(u, v)$ between the vertices u and v in G is the length of a shortest path between them. The largest distance between any pair of vertices in the graph G is the *diameter* in G, also denoted by $diam(G)$. Let v be any vertex in G. The value of $max\{deg(v) | v \in V(G)\}$ is the *maximum degree* of G and denoted by $\Delta(G)$. Furthermore, the value of $min\{deg(v) | v \in V\}$ is the *minimum degree* of G denoted by $\delta(G)$ [9].

The rupture degree $r(G)$ of a graph G is defined as:

$$r(G) = max\{\omega(G - S) - |S| - m(G - S) : \omega(G - S) \geq 2, S \subset V\},$$

where $\omega(G - S)$ denotes the number of connected components in the graph $G - S$ and $m(G - S)$ denotes the size of the largest connected component in the graph $G - S$. A set S which achieves the maximum value is called an r-set. The concept of rupture degree was introduced in [8], then this parameter is also considered as a reasonable parameter for measuring the stability or vulnerability of graphs, see [10–14]. For example, we consider the graph G with six edges and six vertices in Figure 1, where some alternative r-sets of G are illustrated by the set of darkened vertices. Clearly, $|S| = 3, \omega(G - S) = 3$ and $m(G - S) = 1$; therefore, $r(G) = -1$.

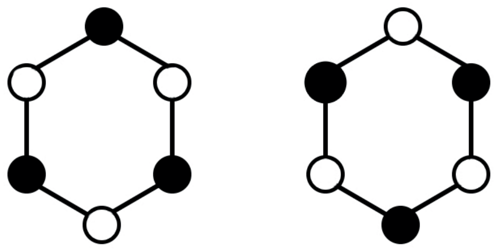


Figure 1. Some r-sets of G.

Moreover, Li et al. [15] showed that the finding rupture degree problem is NP-complete. However, it is possible to determine the rupture degree of large classes of graphs. For more results on rupture degree, we refer the readers to see [16–21]. Furthermore, Li gave an algorithm whose complexity is $O(n^2)$ for isolating rupture degree in Trees of order n [22]. In a similar study, Berberler and Berberler [23] gave a polynomial time heuristic algorithm for computing the integrity of any given graph G.

A variation of vulnerability measures, based on different heuristics, are proposed to quantify the hardness of networks in [23]. In the present study, a heuristic algorithm is developed for computing the rupture degree in graphs by incorporating the concept of the rupture degree and heuristic algorithm. The remainder of the paper is arranged as follows: We first give a few basic theorems related to rupture degree in Section 2. Then, a novel heuristic algorithm is proposed and discussed in an exhaustive schema to compute the rupture degree of graphs in Section 3 and computational experiments are implemented in Section 4. Finally, in Section 5, we present our conclusions.

2. Preliminary results for the rupture degree in graphs

In this section, we give some well-known results for the rupture degree in graphs:

Theorem 1 *[8] For any graph G, $3-n \leq r(G) \leq n-3$ and $r(G) \leq 2\delta(G)-1$.*

Theorem 2 *[8] For the path P_n of order n,* $r(P_n) = \begin{cases} -1 & , if\ n\ is\ even; \\ 0 & , if\ n\ is\ odd. \end{cases}$

Theorem 3 *[8] For the cycle C_n of order n,* $r(C_n) = \begin{cases} -1 & , if\ n\ is\ even; \\ -2 & , if\ n\ is\ odd. \end{cases}$

Theorem 4 *[8] For the complete graph K_n of order n, $r(K_n) = 1-n$.*

Theorem 5 *[8] For the star graph $K_{1,n-1}$ of order $n \geq 3$, $r(K_{1,n-1}) = n-3$.*

Theorem 6 *[8] For the complete bipartite graph $K_{m,n}$, where $1 \leq n \leq m$, $r(K_{m,n}) = m-n-1$.*

Theorem 7 *[8] Let G_1 and G_2 be two graphs of order n_1 and n_2, respectively. Then, $r(G_1+G_2) = max\{r(G_1)-n_2, r(G_2)-n_1\}$.*

3. A heuristic algorithm to find the rupture degree in graphs

In this section, we propose a heuristic algorithm whose pseudocode is presented in Algorithm 1 to find the rupture degree of an arbitrary graph G and runs in polynomial time. The algorithm works as follows:

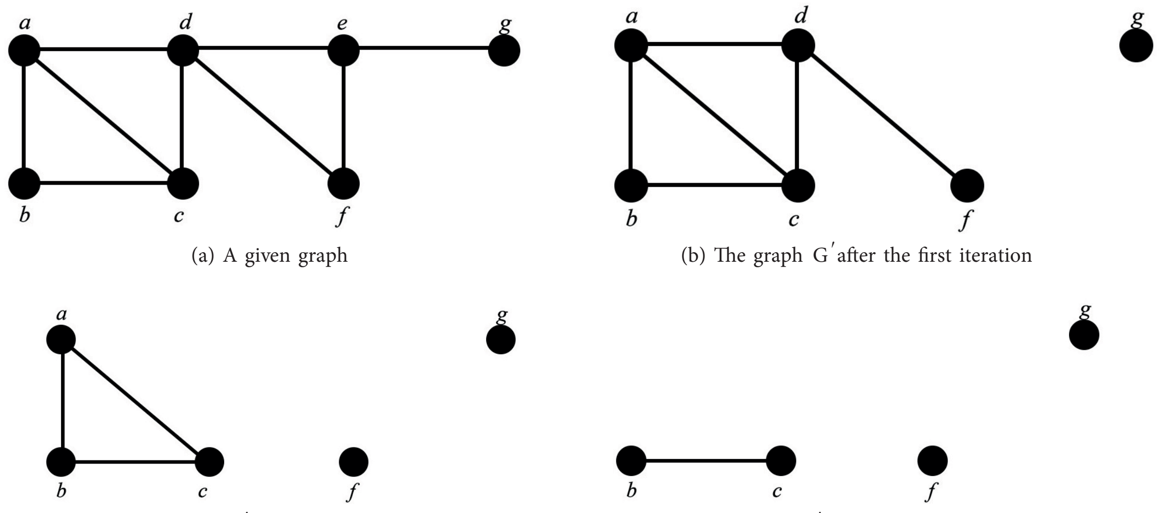
The set S is initially empty. In order to preserve the original graph, we make a copy of the graph as G'. We work on the underlying graph G' in the while loop of lines 5–15. The vertices whose degrees are greater than one in G' are most likely vertices to be included in S. The name of the set of such vertices is $Possible\_S$. The while loop terminates if $Possible\_S$ is empty and in each iteration of the while loop, $Possible\_S$ is updated according to the underlying graph G'. For each vertex in the set $Possible\_S$, a value is computed by the formula in line 12. The vertex with the maximum value is added into S and deleted in G' together with its all incident edges. We compute the rupture degree after the while loop terminates. In case we may add one more vertex into S, we delete each vertex in S one by one and compute the rupture degree as $rupture2$. If $rupture2$ is greater than the previous rupture degree, then we update the degree and delete the vertex in S. Finally, the algorithm return the estimated rupture degree of G. The key point of the algorithm is the formula in line 12. It determines which vertex (with the highest value in the formula) may be included in the $r-$set. While the degree of neighbors of a vertex is effective in the denominator, the degree of the vertex itself is important in the numerator.

We illustrate how the proposed algorithm works on the following graph shown in Figure 2a step by step: In the first iteration of the while loop, $Possible\_S = \{a, b, c, d, e, f\}$ and the values of the vertices in $Possible\_S$ according to the formula in line 12 are $[0.33, 0.22, 0.33, 0.53, 0.55, 0.16]$. The maximum value belongs to the vertex e. Thus, vertex e is added to S. Then we delete e and all its incident edges in the graph. The resulting graph is shown in Figure 2b. In the second iteration of the while loop, $Possible\_S = \{a, b, c, d\}$ and the values of the vertices in $Possible\_S$ according to the formula in line 13 are $[0.42, 0.22, 0.42, 0.55]$. The

Algorithm 1: Heuristic Algorithm to Find the Rupture Degree

```
    Input: An undirected connected simple graph $G = (V, E)$
    Output: Rupture degree of $G$
1   begin
2       $S = \emptyset$;
3       $V' = V$;
4       $G' = G$;
5       while $True$ do
6           for each vertex $i \in V'$ do
7               $degree[i]$=the degree of vertex $i \in G'$ ;
8           $Possible\_S =$The set of vertices with $degree[] > 1$;
9           if $Possible\_S$ is empty then
10              break;
11          for each vertex $i \in Possible\_S$ do
12              $value[i] = \frac{(degree[i])^3}{\left(\sum_{(i,j)\in G'} degree[j]\right)^2}$;
13          $v =$ a vertex with the maximum $value[v]$;
14          $S = S \cup \{v\}$;
15          $V' = V' \setminus \{v\}$; //delete $v$ and all its incident edges in $G'$
16      $rupture = w(G \setminus S) - |S| - m(G \setminus S)$;
17      for each vertex $i \in S$ do
18          $S = S \setminus \{i\}$;
19          $rupture2 = w(G \setminus S) - |S| - m(G \setminus S)$;
20          if $rupture2 > rupture$ then
21              $rupture = rupture2$;
22          else
23              $S = S \cup \{i\}$;
24      Return $rupture$;
```

maximum value belongs to the vertex d. Thus, $S = \{e, d\}$. Then we delete d and all its incident edges in the underlying graph. The resulting graph is shown in Figure 2c. In the third iteration of the while loop, $Possible\_S = \{a, b, c\}$ and the values of the vertices in $Possible\_S$ according to the formula in line 12 are $[0.5, 0.5, 0.5]$. The maximum value is the same for all vertices. We can choose one of them arbitrarily. Let us choose the vertex a. Thus, $S = \{e, d, a\}$. Then we delete a and all its incident edges in the underlying graph. The resulting graph is shown in Figure 2d. In the fourth iteration of the while loop, $Possible\_S = \emptyset$. This terminates the loop with $S = \{e, d, a\}$. In line 16, we evaluate the rupture degree of the given graph in Figure



(a) A given graph
(b) The graph G$'$after the first iteration
(c) The graph G$'$after the 2nd iteration
(d) The graph G$'$after the 3rd iteration

Figure 2. An example of how the proposed algorithm works.

2a for the set S. As can be seen in Figure 2d, the number of connected components after deleting S is three, that is, $w(G \setminus S) = 3$ and the size of the largest connected component is two, that is, $m(G \setminus S) = 2$. The rupture degree $r(G) = 3 - 3 - 2 = -2$. Lastly, we check whether some vertices in S are really necessary to be in S. Therefore, we delete each vertex one by one in S and evaluate the rupture degree. If the latter rupture degree is greater than the previous one, then we update the rupture degree as the latter one. Otherwise, we add again the deleted vertex into S. In this example, the set $S = \{e, d, a\}$ is the r-set.

3.1. Running time analysis of the proposed algorithm

The while loop in line 5 works until there is no vertex whose degree is greater than one in the underlying graph G'. At each iteration of the loop, we delete one vertex and all its incident edges in G'. In the worst case, if the graph is a complete graph, then the while loop works $|V| - 2$ times which runs in $O(|V|)$ time. The inner for loop in line 6 runs in $O(|V|)$ time. The other inner loop in line 11 runs in $O(|V|)$ time. After the while loop terminates, we calculate the rupture degree of the given graph for the set S using the formula in line 16. $w(G \setminus S)$ and $m(G \setminus S)$ are the number of connected components and the size of the largest connected component in $G \setminus S$, respectively. We use depth-first search (DFS) procedure to find the connected components. The DFS algorithm has the worst case running time $O(|V| + |E|)$ [24]. The last for loop in line 17 runs in $O(|V|)$ time. In conclusion, the running time of Algorithm 1 is $O(|V|^2) + |E||V|)$.

4. Computational Tests

Since there are no benchmark instances to compute the rupture degree of graphs, we use the problem instances generated by Berberler and Berberler to find the integrity of graphs [23]. The test instances are available at `http://kisi.deu.edu.tr/murat.berberler/integrity/`.

In Table 1, $|V|$ is the number of vertices of graph G, $R(G)$ is the rupture degree found by the heuristic algorithm in G, *OptR* is the actual rupture degree of G, *Error* is the absolute gap, which is the magnitude of the difference between the actual rupture degree and the rupture degree obtained by the algorithm, $t(s)$ is

Table 1. Computational experiments on small-size graphs.

	25%				50%				75%				95%			
$\|V\|$	R(G)	OptR	Error	t(s)	R(G)	OptR	Error	t	R(G)	OptR	Error	t	R(G)	OptR	Error	t
10	–1	–1	0	0.02	–3	–3	0	0.03	–3	–3	0	0.02	–7	–7	0	0.02
11	–1	–1	0	0.02	–4	–4	0	0.02	–6	–6	0	0.01	–8	–8	0	0.01
12	–1	–1	0	0.02	–5	–5	0	0.02	–8	–8	0	0.01	–9	–9	0	0.01
13	–2	–2	0	0.02	–4	–4	0	0.02	–6	–6	0	0.02	–10	–10	0	0.01
14	–3	–3	0	0.02	–6	–6	0	0.02	–9	–9	0	0.02	–11	–11	0	0.01
15	–2	–2	0	0.02	–6	–6	0	0.02	–8	–8	0	0.02	–12	–12	0	0.01
16	–3	–3	0	0.02	–7	–7	0	0.02	–11	–11	0	0.02	–13	–13	0	0.02
17	–2	–2	0	0.03	–8	–8	0	0.03	–12	–12	0	0.02	–14	–14	0	0.02
18	–3	–3	0	0.03	–8	–8	0	0.03	–12	–12	0	0.02	–15	–15	0	0.02
19	–4	–4	0	0.03	–8	–8	0	0.03	–12	–12	0	0.03	–14	–14	0	0.02
20	–5	–5	0	0.04	–9	–9	0	0.05	–13	–13	0	0.02	–17	–17	0	0.02
21	–7	–7	0	0.04	–12	–12	0	0.04	–16	–16	0	0.02	–16	–16	0	0.02
22	–6	–6	0	0.04	–12	–12	0	0.04	–15	–15	0	0.03	–19	–19	0	0.02
23	–7	–7	0	0.04	–12	–12	0	0.04	–16	–16	0	0.03	–20	–20	0	0.02
24	–9	–8	1	0.05	–14	–14	0	0.04	–19	–17	2	0.03	–21	–20	1	0.02
25	–9	–9	0	0.05	–14	–14	0	0.04	–18	–18	0	0.03	–22	–22	0	0.02
26	–10	–9	1	0.05	–17	–16	1	0.04	–19	–19	0	0.03	–23	–23	0	0.03
27	–10	–10	0	0.07	–16	–16	0	0.07	–20	–20	0	0.03	–24	–23	1	0.04
28	–9	–9	0	0.06	–15	–15	0	0.05	–21	–21	0	0.04	–25	–25	0	0.03
29	–9	–9	0	0.07	–19	–18	1	0.06	–22	–22	0	0.06	–24	–24	0	0.04
30	–10	–10	0	0.09	–19	–18	1	0.06	–23	–23	0	0.05	–27	–27	0	0.04
31	–11	–11	0	0.09	–18	–18	0	0.06	–25	–25	0	0.04	–26	–26	0	0.03
32	–14	–13	1	0.07	–21	–20	1	0.06	–25	–25	0	0.05	–28	–28	0	0.05

Table 2. Computational experiments on medium-size graphs.

	20%		30%		40%		50%	
$\|V\|$	R(G)	t(s)	R(G)	t(s)	R(G)	t(s)	R(G)	t(s)
100	–60	0.619	–75	0.461	–78	0.339	–84	0.332
110	–71	0.489	–83	0.376	–90	0.295	–93	0.286
120	–83	0.560	–92	0.440	–96	0.384	–104	0.308
130	–86	0.677	–99	0.499	–109	0.420	–113	0.318
140	–97	0.724	–109	0.618	–117	0.427	–124	0.399
150	–112	0.685	–118	0.597	–127	0.478	–133	0.386
160	–117	0.796	–129	0.638	–138	0.514	–143	0.419
170	–124	0.908	–140	0.676	–147	0.518	–151	0.509
180	–133	0.987	–149	0.736	–155	0.660	–162	0.520
190	–146	1.062	–158	0.798	–166	0.665	–172	0.534
200	–155	1.115	–168	0.847	–176	0.716	–182	0.577

Table 3. Computational experiments on medium-size graphs.

	60%		70%		80%		90%	
$\|V\|$	R(G)	t(s)	R(G)	t(s)	R(G)	t(s)	R(G)	t(s)
100	–86	0.258	–89	0.196	–92	0.181	–93	0.147
110	–98	0.217	–101	0.188	–101	0.179	–103	0.142
120	–107	0.231	–110	0.212	–111	0.206	–113	0.147
130	–118	0.265	–119	0.254	–122	0.328	–125	0.162
140	–127	0.341	–129	0.247	–131	0.216	–133	0.185
150	–137	0.352	–137	0.297	–141	0.259	–144	0.223
160	–145	0.389	–149	0.320	–151	0.290	–153	0.248
170	–155	0.413	–159	0.364	–162	0.306	–163	0.265
180	–164	0.473	–167	0.371	–171	0.331	–173	0.295
190	–175	0.500	–179	0.409	–180	0.392	–183	0.316
200	–185	0.477	–190	0.395	–191	0.380	–193	0.309

the CPU time in seconds and 25%, 50%, 75%, and 95% indicate the edge density of G. We found the actual rupture degree of small-size graphs of up to 32 vertices by brute force technique.

The proposed algorithm was implemented in MATLAB and tested on i7-5600U machine with a 2.60 GHz processor and 8GB RAM. As it can be seen in Table 1, the proposed algorithm almost found the optimal (actual) rupture degree of the graphs. We also tested the proposed algorithm on the medium-size graphs with more than 100 vertices. Since we do not know the actual rupture degree of the graphs with more than 100 vertices, we give the rupture degrees obtained by the algorithm and the CPU time in Tables 2 and 3. Furthermore, we tested the algorithm on several specific classes of graphs such as complete graphs, comets, cycles, and path graphs. We obtained the same rupture degree as given in the theorems.

5. Conclusion

In this paper, we considered the rupture degree in graphs that are usually thought to measure the vulnerability of graphs. We proposed a polynomial time heuristic algorithm to find the rupture degree of graphs. We then analyzed the running time of our algorithm and presented the computational experiments on graphs with up to 200 vertices. The results show that the proposed heuristic algorithm efficiently computes the rupture degree of a given graph. Developing several heuristics for computing the other graph parameters such as the edge rupture degree, the weak rupture degree, and the neighbor rupture degree of graphs are the subjects of future work.

References

[1] Mishkovski I, Biey M, Kocarev L. Vulnerability of complex networks. Communications in Nonlinear Science and Numerical Simulation 2011; 16 (1): 341-349. doi: 10.1016/j.cnsns.2010.03.018

[2] Durgut R, Turaci T, Kutucu H. Global distribution center number of some graphs and an algorithm. RAIRO-Operations Research 2018; *In press.* doi: 10.1051/ro/2018119

[3] Turaci T, Okten M. Vulnerability of mycielski graphs via residual closeness. Ars Combinatoria 2015; 118: 419-427.

[4] Frank H, Frisch IT. Analysis and design of survivable networks. IEEE Transactions on Communications Technology 1970; 18 (5): 501-519. doi: 10.1109/TCOM.1970.1090419

[5] Chvatal V. Tough graphs and hamiltonian circuits. Discrete Mathematics 1973; 5 (3): 215-228. doi: 10.1016/0012-365X(73)90138-6

[6] Bagga KS, Beineke LW, Goddard WD, Lipman MJ, Pippert RE. A survey of integrity. Discrete Applied Mathematics 1992; 37-38: 13-28. doi: 10.1016/0166-218X(92)90122-Q

[7] Jung HA. On maximal circuits infinite graphs. Annals of Discrete Mathematics 1978; 3: 129-144. doi: 10.1016/S0167-5060(08)70503-X

[8] Li Y, Zhang S, Li X. Rupture degree of graphs. International Journal of Computer Mathematics 2005; 82 (7): 793-803. doi: 10.1080/00207160412331336062

[9] West BD. Introduction to Graph Theory. 2nd ed. Urbana, NJ, USA: Prentice Hall, 2001.

[10] Aslan E. Weak-rupture degree of graphs. International Journal of Foundations of Computer Science 2016; 27 (6): 725-738. doi: 10.1142/S0129054116500258

[11] Aslan E. Edge-rupture degree of a graph. Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms 2015; 22 (2): 155-161.

[12] Bacak B, Oz E. Neighbor rupture degree of transformation graphs. International Journal of Foundations of Computer Science 2017; 28 (4): 335-355. doi: 10.1142/S0129054117500216

[13] Kurkçu OK, Aslan E. A comparison between edge neighbor rupture degree and edge scattering number in graphs. International Journal of Foundations of Computer Science 2018; 29 (7): 1119-1142. doi: 10.1142/S0129054118500247

[14] Aytac V, Turaci T. Relationships between vertex attack tolerance and other vulnerability parameters. RAIRO - Theoretical Informatics and Applications 2017; 51 (1): 17-27. doi: 10.1051/ita/2017005

[15] Li F, Li X. Computing the rupture degrees of graphs. In: 7th International Symposium on Parallel Architectures, Algorithms and Networks; Hong Kong, China; 2004, pp. 368-373.

[16] Aytac A, Aksu H. Some results for the rupture degree. International Journal of Foundations of Computer Science 2013; 24 (8): 1329-1338. doi: 10.1142/S0129054113500366

[17] Aytac A, Aksu H. The rupture degree of some graphs. Mathematica Balkanica 2010; 24 (1-2): 85-101.

[18] Aytac A, Odabas ZN. Computing the rupture degree in composite graphs. International Journal of Foundations of Computer Science 2010; 21 (3): 311-319. doi: 10.1142/S012905411000726X

[19] Kirlangic A, Bacak-Turan G. On the rupture degree of a graph. Neural Network World 2012; 22 (1): 39-51. doi: 10.14311/NNW.2012.22.003

[20] Li Y. The rupture degree of trees. International Journal of Computer Mathematics 2008; 85 (11): 1629-1635. doi: 10.1080/00207160701553367

[21] Odabas ZN, Aytac A. Rupture degree and middle graphs. Comptes Rendus de Lacademie Bulgare des Sciences 2010; 65 (3): 315-322.

[22] Li F. Isolated rupture degree of trees and gear graphs. Neural Network World 2015; 25 (3): 287-300. doi: 10.14311/NNW.2015.25.015

[23] Berberler ME, Berberler ZN. Measuring the vulnerability in networks: a heuristic approach. Ars Combinatoria 2017; 135: 3-15.

[24] Cormen HT, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed. Cambridge, London, UK: The MIT Press, 2009.