# Measuring the vulnerability in networks: a heuristic approach

Murat Erşen BERBERLER
*Faculty of Science, Department of Computer Science, Dokuz Eylul University, 35160, Izmir/TURKEY*
*murat.berberler@deu.edu.tr*

Zeynep Nihan BERBERLER
*Faculty of Science, Department of Computer Science, Dokuz Eylul University, 35160, Izmir/TURKEY*
*zeynep.berberler@deu.edu.tr*

**Abstract.** The integrity of a graph $G=(V,E)$ is defined as $I(G)=\min\{|S|+m(G-S): S\subseteq V(G)\}$, where $m(G-X)$ denotes the order of the largest component in the graph $G-X$. This is a better parameter to measure the stability of a network, as it takes into account both the amount of work done to damage the network and how badly the network is damaged. Computationally it belongs to the class of intractable problems known as $NP$-hard. In this paper we develop a heuristic algorithm to determine the integrity of a graph. Extensive computational experience on 88 randomly generated graphs ranging from 20% to 90% densities and from 100 to 200 vertices has shown that the proposed algorithm is very effective.
*Keywords:* Network vulnerability, Integrity, Heuristics

## 1. Introduction

Networks with complex topology describe a wide range of systems in nature and society including examples social networks, the World Wide Web, the Internet, neural networks, chemical and biological networks, electric power grids, supply chains, urban road networks, and communication networks. The study on networks is an important area of multidisciplinary research involving social sciences, mathematics, physics, chemistry, biology, informatics, and other theoretical and applied sciences [13-14,17-18]. The stability of a communication network, composed of processing nodes and communication links, is of prime importance to network designers. As the network begins losing links or nodes, eventually there is a loss in its effectiveness. In an analysis of the vulnerability of a communication network to disruption, two quantities that come to mind are the number of elements that are not functioning and the size of the largest remaining sub-network within which mutual communication can still occur. Therefore, it would be desirable for a network to be such that the two qualities can be made to be simultaneously small. Thus, communication networks must be constructed to be as stable as possible, not only with respect to the initial disruption, but also with respect to the possible reconstruction of the network.

A variety of measures have been proposed to quantify the robustness of networks. The vertex (edge) connectivity of a graph is an important measure for robustness of a network [8]. However, the connectivity only partly reflects the ability of graphs to ensure connectedness after vertex (or edge) deletion. Other improved measures are toughness [19], scattering number [7], integrity [1], tenacity [12], etc. In contrast to vertex (edge) connectivity, these measures consider both the cost to damage a network and how badly the network is damaged. Unfortunately, from an algorithmic point of view, the problem of calculating these measures for general graphs is nondeterministic polynomial-time hard problem [4].

The integrity of a graph was introduced in 1987 by Barefoot, Entringer and Swart [1]. They defined the *integrity* $I(G)$ of a graph $G$ as

$$\min_{S \subset V(G)} \{|S| + m(G-S)\},$$

where $m(G-S)$ denotes the maximum order of a component of $G-S$, and the minimum is taken over all noncomplete subsets $S$ of the vertex set $V$ of $G$. A set $S$ which achieves the minimum is called an $I$-set. A subset $S$ of $V$ is called a *cut set* of a graph $G$, if $w(G-S) > w(G)$ or $G-S$ is an isolated vertex, where $w(G)$ is the number of components of $G$. For a survey of integrity, see Bagga et al. [10].

In [1], Barefoot, Entringer and Swart compared integrity, connectivity, binding number and toughness for several classes of graphs. Their results suggested that integrity is well suited to measuring vulnerability in that it is best able to distinguish between graphs that intuitively should have different measures of vulnerability.

Note that the definition of integrity does not require that a graph be connected. Moreover, the original definition permitted the removal of all vertices. As immediate consequences of the definition, we have that if $G$ is a graph of order $n$, then $1 \le I(G) \le n$, and for any subgraph $H$ of $G$, $I(H) \le I(G)$. Clark, Entringer, and Fellows [11] showed that the vertex integrity problem is $NP$-complete, even when restricted for planar graphs but that for each fixed value of $k$ it is decidable in time $O(n^2)$ whether an arbitrary graph $G$ of order $n$ satisfies $I(G) \le k$. This last result is obtained by a simple application of the powerful results of Robertson and Seymour on graph minors. Moreover, Moazzami [5] showed that the vertex integrity problem is $NP$-hard. However, it is possible to determine the integrity of large classes of graphs. For more results on integrity see [10].

It is well known that a network can be represented by a graph. Vertices and edges of the graph correspond to nodes and links between the nodes, respectively. In this paper, we consider simple finite undirected graphs without

loops and multiple edges. Let $G=(V,E)$ be a graph with a vertex set $V=V(G)$ and an edge set $E=E(G)$. The *order* of $G$ is the number of vertices in $G$. The *degree* $\deg_G(v)$ of a vertex $v\in V(G)$ is the number of edges incident to $v$. The *maximum degree* of $G$ is $\Delta(G)=\max\{\deg_G(v)|v\in V(G)\}$. A vertex of degree zero is an *isolated vertex* or an *isolate*. For vertices $u$ and $v$ of a graph $G$, the *open neighborhood* of $u$ is $N(u)=\{v\in V(G)|(u,v)\in E(G)\}$. We define analogously for any $S\subseteq V(G)$ the open neighborhood $N(S)=\bigcup_{u\in S}N(u)$. [3,6,9].

The paper proceeds as follows. In Section 2, a novel algorithm based on heuristic is proposed and discussed in an exhaustive schema to compute the integrity of networks. Computational experiments are implemented in Section 3.

## 2. The computation of integrity based on heuristic

A graph labeling is an assignment of integers to the vertices or edges, or both, subject to certain conditions. When searching for heuristic/algorithmic solutions for the graph problems related to *NP* complexity class, it is reported that the labeling of the vertices in graph is very important for determination the solution quality [2, 15-16]. Since the number of labeling is $n!$ for a given graph $G$ with $n$ vertices, it can be clearly seen that finding the most appropriate labeling for a given specific problem is intractable. However, none of the heuristics can try to find the optimal labeling because being optimal is specific for the problem and is very costly in the sense of combinatorics so this conflicts the philosophy of the heuristic algorithms that it is expected to obtain results in short times. Instead, in order to minimize the disadvantages of the labeling, algorithm runs by manipulating the labels of vertices in a graph with different criteria.

The graph shown in Figure 1 can be handled to observe the effect of the labeling on the integrity problem. The graph $G$ has 7 vertices: 4 vertices of degree 3 and 2 vertices of degree 2. To create a handicap for a heuristic algorithm based on the degrees of vertices of a graph, the labeling is chosen deliberately as follows in Figure 1. If the heuristic algorithm sorts the degrees of the vertices of the graph in non-increasing order based on the original labeling in Figure 1, then the vector is obtained as [1,2,5,7,3,4,6].

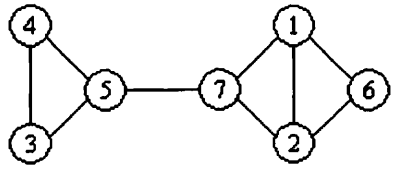



Figure 1. The graph $G$

The results obtained by running the heuristic step by step that uses this vector statically are given in Figure 2. Here "statically" means that when a vertex is removed from the graph, the degrees of the remaining vertices are not updated and remain the same as in the original graph. It can be observed that using the labeling vector statically does not affect the optimal solution significantly for this example but when a path is taken into consideration and the labeling occurs sequentially –for example for $P_5$ $V_1, V_2, V_3, V_4, V_5$–, it can be easily seen that the optimal solution *lopt* that should be achieved is obtained as $2 * lopt - 2$. Absolutely, the difference is not in acceptable range. The shaded vertices in Figure 2 mean that these vertices will be deleted in the next step.

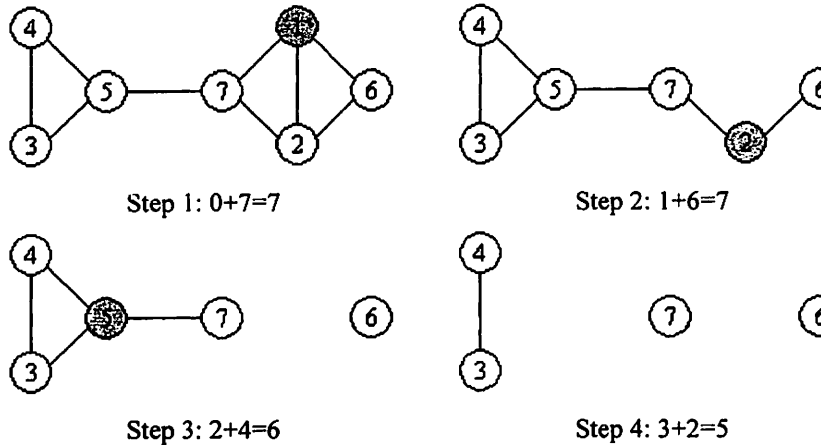
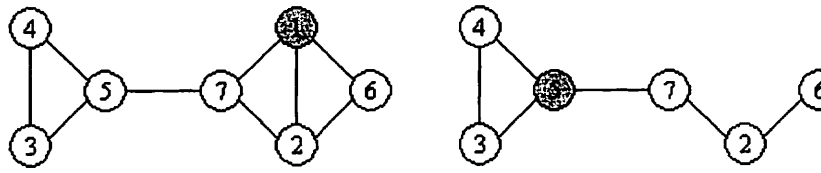
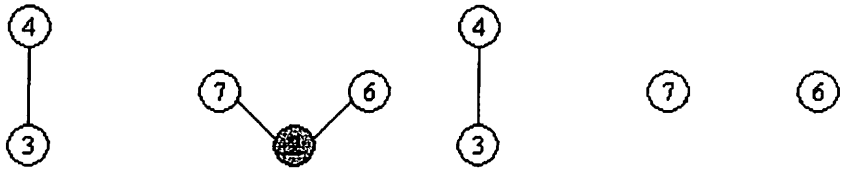



Figure 2.

Hence, if the labeling vector that is manipulated based on the degrees of vertices is used dynamically instead of the static use that is only advantageous by means of execution time but does not improve the solution quality anymore, then the algorithm generates the steps in Figure 3 and it is seen that again the optimal solution cannot be found.





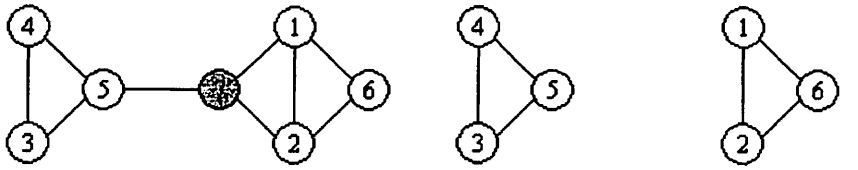

Step 3: 2+3=5 Step 4: 3+2=5

Figure 3.

As seen in Figure 3, for the labeling manipulation that the heuristic will perform, we have noticed that other criteria are needed and this is the key aspect for the motivation on this paper. It is observed that the criterion that makes the vertex prior is the sum of the degrees of the vertices in open neighborhood (set). When sorting based on the degrees of vertices, in the case of the equality of the degrees, if the labeling vector is created by giving priority to the vertex that has the greater sum of the degrees of neighboring vertices, then the vector [7,1,2,5,6,3,4] is obtained, and hence this leads the optimal solution at first step (see Table 1 and Figure 4). In Table 1, $N = \sum \deg_G(v)$, where $v \in N(u)$.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | $\deg_G(v)$ | $N$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | | | | 1 | 1 | 3 | 8 |
| 2 | 1 | | | | | 1 | 1 | 3 | 8 |
| 3 | | | | 1 | 1 | | | 2 | 5 |
| 4 | | | 1 | | 1 | | | 2 | 5 |
| 5 | | | 1 | 1 | | | 1 | 3 | 7 |
| 6 | 1 | 1 | | | | | | 2 | 6 |
| 7 | 1 | 1 | | | 1 | | | 3 | 9 |

Table 1.




Step 1: 0+7=7 Step 2: 1+3=4

Figure 4.

It is clear that having obtained the optimal solution at first step in Figure 4 is a special case and other techniques are needed for the heuristic to obtain the optimal or near-optimal solution for the more complex scenarios. In this study, two main approaches, cycle technique and frequency of use, are proposed.

### *2.1 Cycle technique*

Cycle technique includes converting the labeling vector to a circular queue and detecting the solution space wisely starting from a different initial point at each step. For instance, when cycle technique is applied to the labeling vector [1,2,3,4,5], the following sequences are obtained.

[1,2,3,4,5]
[2,3,4,5,1]
[3,4,5,1,2]
[4,5,1,2,3]
[5,1,2,3,4]

Thus, the number of the adverse cases is decreased related to the faulty selection of the sorting criterion used for labeling or the choice of pre-post in case of equality encountered while sorting. Here, by means of "equality", it is emphasized that when the elements handled are equal to each other, the problem is dealt with the case in which one will have the priority based on the labeling. This case can occur also in any simple sorting algorithm.

Although the integrity problem belongs to $NP$ class because of being a subset selection problem in its origin, a heuristic algorithm that tries to generate a solution for this problem should handle the vertices that will be removed from the graph in a proper order –if it does not have a tricky point of view–. Since the ordering is taken into account, the algorithm must attempt to evaluate a considerable part of $n!$ possible cases although no assessment for all cases is required. That, the following parts will explain how to perform this task.

The five main branches seen above are obtained using cycle technique by accepting that the labeling vector that is manipulated by using the degrees of vertices is [1,2,3,4,5]. The method that is based on taking the increment amount as a variable is used in order to create the sub-branches of those five main branches for efficient scan of solution space. Here when the increment amount is 1, the vector gives the order as 1,2,3,4,5; when it is 2, the vector gives the order as 1,3,5,2,4, and when it is 3, the vector will give the order as 1,4,3,5,2. Note that the increment amount is chosen from the interval $[1, n]$. Thus, in the nested loops used both in two main parts of the heuristic that determine the complexity of the algorithm, the outermost loop is for the starting vertex and the innermost loop is for the amount of increment.

### *2.2 Frequency of use*

While searching for solution by cycle technique, at that step $+1$ point is given to the vertices that update the value of integrity and $+n$ points are given to the vertices that update the value of integrity with less value. Proceeding by this way, a new manipulation criterion is obtained different than the degrees of the vertices. When the cycle technique terminates, a new labeling is obtained by sorting the frequency vector in non-increasing order and by using this vector, cycle technique is run once more.

When the algorithm terminates, the least integrity value stored and the vector including the vertices that gives this value are printed in a file with the running time.

### *2.3 Formal description of the heuristic algorithm*

*Step 1.* The vertex adjacency matrix of the graph is entered.

*Step 2.* The vertex labeling vector that is sorted in non-increasing order by considering the degree of each vertex and the sum of the degrees of each neighbor of the related vertex constitutes the initial vector of the cycle technique.

*Step 3.* While the solution space is scanning by cycle technique, the best solution is saved as a record and the frequency vector that saves the frequency of use of the vertices placed in the best solution is updated at every step.

*Step 4.* The vertex labeling vector belonging the frequency that will be used in the second stage of the algorithm is generated by sorting the frequency vector in non-increasing order.

*Step 5.* The solution space is scanned by the cycle technique that uses the vertex labeling vector depending on frequency and the record solution is updated if required.

*Step 6.* Algorithm terminates and the record is printed.

### *2.4 The time and space complexities of the heuristic algorithm*

The most important data structure that determines the space complexity is the two-dimensional $n \times n$ vector that represents the vertex adjacency matrix of the graph, so the cost is $O(n^2)$. Since other data structures used are the one-dimensional vectors with $n$ elements, they don't affect the space complexity of big $O$ notation.

The most important factor that affects the time complexity of a heuristic algorithm is the cycle technique. The outer loop of $n$ steps that constitutes the cycle technique determines the initial value of the tour while the inner loop of $n$ steps determines the amount of increment. The integrity value is evaluated in the innermost loop that uses each vector obtained by cycle technique. In conclusion, the heuristic algorithm has time complexity of $O(n^3)$, the consistency of the complexity is checked by computational experiments as seen in Table 5.

## 3. Computational experiments

In order to test the solution quality of the proposed heuristic algorithm, primarily computational experiments are performed for small-sized samples on which the optimal solution can be found by enumeration technique. Related optimal solutions and belonging results can be seen in Table 2. The problems in Table 2 are randomly generated in which $n$ is the number of vertices of graph

$G$, $I(G)$ is the integrity value of $G$, $t(\text{sec})$ is the $CPU$ time and 25%, 50%, 75%, and 95% indicate the edge density of $G$.

| | 25% | | 50% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t(\text{sec})$ | $I(G)$ | $t(\text{sec})$ |
| 10 | 5 | 0,000 | 7 | 0,000 |
| 11 | 5 | 0,000 | 8 | 0,000 |
| 12 | 7 | 0,000 | 9 | 0,000 |
| 13 | 8 | 0,000 | 9 | 0,000 |
| 14 | 8 | 0,003 | 10 | 0,000 |
| 15 | 8 | 0,017 | 11 | 0,001 |
| 16 | 9 | 0,018 | 12 | 0,030 |
| 17 | 10 | 0,032 | 13 | 0,060 |
| 18 | 10 | 0,061 | 13 | 0,122 |
| 19 | 11 | 0,137 | 14 | 0,371 |
| 20 | 12 | 0,260 | 15 | 0,603 |
| 21 | 12 | 0,588 | 16 | 0,726 |
| 22 | 13 | 0,961 | 17 | 3,488 |
| 23 | 14 | 3,346 | 18 | 7,761 |
| 24 | 15 | 7,604 | 19 | 15,825 |
| 25 | 15 | 13,904 | 19 | 36,260 |
| 26 | 17 | 25,760 | 21 | 76,882 |
| 27 | 18 | 66,774 | 22 | 162,711 |
| 28 | 18 | 150,776 | 22 | 318,336 |
| 29 | 18 | 301,518 | 23 | 670,070 |
| 30 | 19 | 667,744 | 24 | 1.506,323 |
| 31 | 20 | 1.331,120 | 24 | 3.102,376 |
| 32 | 20 | 2.598,510 | 26 | 6.390,127 |
| 33 | 22 | 3.333,925 | 26 | 7.015,158 |
| 34 | 21 | 11.287,910 | 27 | 21.583,944 |
| 35 | 22 | 25.309,292 | 29 | 52.011,686 |

Table 2.1.

| | 75% | | 95% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t(\text{sec})$ | $I(G)$ | $t(\text{sec})$ |
| 10 | 7 | 0,000 | 9 | 0,000 |
| 11 | 9 | 0,000 | 10 | 0,000 |

| | | | | |
|---|---|---|---|---|
| 12 | 10 | 0,000 | 11 | 0,000 |
| 13 | 10 | 0,000 | 12 | 0,000 |
| 14 | 12 | 0,000 | 13 | 0,000 |
| 15 | 12 | 0,001 | 14 | 0,000 |
| 16 | 14 | 0,029 | 15 | 0,029 |
| 17 | 15 | 0,091 | 16 | 0,107 |
| 18 | 15 | 0,201 | 17 | 0,246 |
| 19 | 16 | 0,480 | 17 | 0,541 |
| 20 | 17 | 1,071 | 19 | 0,211 |
| 21 | 18 | 2,209 | 19 | 2,614 |
| 22 | 19 | 4,922 | 21 | 5,624 |
| 23 | 20 | 10,428 | 22 | 12,300 |
| 24 | 21 | 22,111 | 22 | 26,183 |
| 25 | 22 | 46,789 | 24 | 55,868 |
| 26 | 23 | 101,201 | 25 | 115,773 |
| 27 | 24 | 220,089 | 25 | 255,609 |
| 28 | 25 | 465,834 | 27 | 556,875 |
| 29 | 26 | 972,755 | 27 | 1.196,367 |
| 30 | 27 | 2.083,617 | 29 | 2.546,673 |
| 31 | 28 | 4.292,939 | 29 | 5.514,203 |
| 32 | 29 | 8.987,844 | 30 | 11.403,165 |
| 33 | 29 | 9.917,838 | 32 | 13.312,233 |
| 34 | 30 | 31.799,092 | 32 | 41.722,300 |
| 35 | 31 | 77.678,434 | 33 | 105.140,196 |

Table 2.2.

The proposed heuristic algorithm finds the optimal solutions for all problems in Table 2 in *CPU* times given in Table 3.

| | $t(\text{sec})$ | | | |
|---|---|---|---|---|
| *n* | 25% | 50% | 75% | 95% |
| 10 | 0,000 | 0,000 | 0,000 | 0,000 |
| 11 | 0,000 | 0,000 | 0,000 | 0,000 |
| 12 | 0,000 | 0,000 | 0,000 | 0,000 |
| 13 | 0,000 | 0,000 | 0,000 | 0,000 |
| 14 | 0,000 | 0,000 | 0,000 | 0,000 |
| 15 | 0,000 | 0,000 | 0,000 | 0,000 |
| 16 | 0,000 | 0,000 | 0,000 | 0,000 |
| 17 | 0,000 | 0,000 | 0,000 | 0,000 |

| | | | | |
|---|---|---|---|---|
| 18 | 0,000 | 0,000 | 0,000 | 0,000 |
| 19 | 0,000 | 0,000 | 0,000 | 0,010 |
| 20 | 0,000 | 0,000 | 0,016 | 0,012 |
| 21 | 0,000 | 0,000 | 0,016 | 0,015 |
| 22 | 0,015 | 0,016 | 0,016 | 0,019 |
| 23 | 0,015 | 0,016 | 0,031 | 0,023 |
| 24 | 0,031 | 0,031 | 0,031 | 0,028 |
| 25 | 0,031 | 0,031 | 0,031 | 0,034 |
| 26 | 0,031 | 0,031 | 0,047 | 0,037 |
| 27 | 0,031 | 0,047 | 0,047 | 0,048 |
| 28 | 0,031 | 0,047 | 0,047 | 0,050 |
| 29 | 0,031 | 0,047 | 0,047 | 0,063 |
| 30 | 0,031 | 0,047 | 0,078 | 0,078 |
| 31 | 0,031 | 0,063 | 0,078 | 0,093 |
| 32 | 0,062 | 0,078 | 0,094 | 0,094 |
| 33 | 0,062 | 0,078 | 0,109 | 0,110 |
| 34 | 0,062 | 0,089 | 0,125 | 0,141 |
| 35 | 0,078 | 0,109 | 0,141 | 0,156 |

Table 3.

After the effectiveness of heuristic is observed on the sample problems, a problem library is generated randomly being $n = 100,110,\ ,200$ and edge density $20,30,\ ,90$. The optimal values belonging to library problems and the running time of heuristic are seen in the following tables.

| | 20% | | 30% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t(\text{sec})$ | $I(G)$ | $t(\text{sec})$ |
| 100 | 77 | 8,582 | 83 | 11,853 |
| 110 | 87 | 13,315 | 94 | 18,874 |
| 120 | 95 | 19,966 | 103 | 28,372 |
| 130 | 104 | 29,624 | 113 | 41,864 |
| 140 | 113 | 41,870 | 121 | 61,446 |
| 150 | 125 | 58,392 | 131 | 86,193 |
| 160 | 132 | 79,747 | 141 | 118,299 |
| 170 | 143 | 107,121 | 151 | 158,922 |
| 180 | 151 | 141,756 | 160 | 209,962 |
| 190 | 161 | 184,885 | 170 | 269,732 |
| 200 | 171 | 237,822 | 180 | 345,673 |

Table 4.1.1.

| | 40% | | 50% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t$(sec) | $I(G)$ | $t$(sec) |
| 100 | 87 | 15,482 | 91 | 18,570 |
| 110 | 97 | 24,474 | 100 | 29,814 |
| 120 | 106 | 36,759 | 110 | 46,096 |
| 130 | 117 | 54,518 | 120 | 68,257 |
| 140 | 126 | 79,515 | 130 | 98,393 |
| 150 | 136 | 110,712 | 139 | 136,650 |
| 160 | 146 | 152,693 | 150 | 185,193 |
| 170 | 156 | 203,658 | 158 | 250,835 |
| 180 | 166 | 268,157 | 169 | 329,845 |
| 190 | 176 | 351,868 | 179 | 436,878 |
| 200 | 185 | 452,352 | 189 | 562,917 |

Table 4.1.2.

| | 60% | | 70% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t$(sec) | $I(G)$ | $t$(sec) |
| 100 | 92 | 22,233 | 94 | 26,050 |
| 110 | 103 | 35,354 | 104 | 40,529 |
| 120 | 112 | 54,583 | 114 | 62,912 |
| 130 | 122 | 80,928 | 124 | 93,996 |
| 140 | 132 | 113,839 | 134 | 132,986 |
| 150 | 142 | 162,661 | 143 | 188,046 |
| 160 | 152 | 221,135 | 154 | 259,519 |
| 170 | 162 | 296,380 | 163 | 349,559 |
| 180 | 171 | 391,588 | 174 | 472,567 |
| 190 | 181 | 517,651 | 183 | 635,449 |
| 200 | 191 | 668,786 | 193 | 819,037 |

Table 4.2.1.

| | 80% | | 90% | |
|---|---|---|---|---|
| $n$ | $I(G)$ | $t$(sec) | $I(G)$ | $t$(sec) |
| 100 | 96 | 28,053 | 97 | 28,260 |
| 110 | 106 | 44,004 | 107 | 45,678 |
| 120 | 115 | 67,500 | 117 | 69,667 |
| 130 | 125 | 103,413 | 127 | 103,546 |
| 140 | 135 | 145,026 | 137 | 146,522 |

| | | | | |
|---|---|---|---|---|
| 150 | 145 | 204,111 | 147 | 208,921 |
| 160 | 155 | 280,944 | 157 | 291,701 |
| 170 | 165 | 380,063 | 166 | 398,589 |
| 180 | 175 | 502,858 | 176 | 524,410 |
| 190 | 185 | 656,706 | 187 | 680,816 |
| 200 | 195 | 854,995 | 197 | 897,400 |

Table 4.2.2.

Computational experiments are performed on a computer with Intel Core2 2.8 *GHz CPU* and 3 *GB* Ram and all problems and programs are available in the following link: http://kisi.deu.edu.tr/murat.berberler/integrity/.

| | Cubic Polynomial Regression | | | | |
|---|---|---|---|---|---|
| Density | $X^3$ | $X^2$ | $X^1$ | $X^0$ | $R^2$ |
| 20% | 0,0002 | -0,0489 | 5,3189 | -201,25 | 1 |
| 30% | 0,0002 | -0,0527 | 5,1995 | -180,30 | 1 |
| 40% | 0,0003 | -0,0903 | 9,7519 | -367,76 | 0,999 |
| 50% | 0,0005 | -0,1331 | 15,044 | -589,77 | 0,999 |
| 60% | 0,0005 | -0,1562 | 17,551 | -683,80 | 0,999 |
| 70% | 0,0007 | -0,2164 | 24,491 | -952,38 | 0,999 |
| 80% | 0,0006 | -0,1930 | 21,445 | -827,63 | 0,999 |
| 90% | 0,0007 | -0,2005 | 22,008 | -837,84 | 0,999 |

Table 5

## 4. References


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

[2] D.B. Johnson, Parallel algorithms for minimum cuts and maximum flows in planar networks, Journal of the ACM, 34(4) (1987) 950-967.

[3] D.B. West, Introduction to Graph Theory, Prentice Hall, NJ (2001).

[4] D. Kratsch, T. Kloks, H. Müller, Measuring the vulnerability for classes of intersection graphs, Discrete Applied Mathematics 77 (1997) 259-270.

[5] D. Moazzami, An Algorithm for the Computation of Edge Integrity, International Journal of Contemporary Mathematical Sciences 6(11) (2011) 507-516.

[6] G. Chartrand, L. Lesniak, Graphs and Digraphs, Second Edition, Wadsworth. Monterey (1986).

[7] H.A. Jung, On a class of posets and the corresponding comparability graphs, Journal of Combinatorial Theory, Series B 24 (2) (1978) 125–133.

[8] H. Frank, I.T. Frisch, Analysis and design of survivable networks, IEEE Transactions on Communications Technology COM-18 567 (1970).
[9] J.A. Bondy, U.S.R. Murty, Graph theory with applications, American Elsevier Publishing Co., Inc., New York (1976).
[10] K.S. Bagga, L.W. Beineke, W.D. Goddard, M.J. Lipman, R.E. Pippert, A survey of integrity, *Discrete Appl. Math.* 37/38 (1992) 13-28.
[11] L.C. Clark, R.C. Entringer, M.R. Fellows, Computational complexity of integrity, *J. Combin. Math. Combin. Comput.* 2 (1987) 179-191.
[12] M. Cozzens, D. Moazzami, S. Stueckle, Seventh International Conference on the Theory and Applications of Graphs, Wiley, New York (1995) 1111–1122.
[13] M.E.J. Newman, The structure and function of complex networks, SIAM Review 45 (2003) 167-256.
[14] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of Modern Physics 74 (2002) 47-97.
[15] R.G. Downey, P.A. Evans, M.R. Fellows, Parameterized learning complexity, Proceedings of the sixth annual conference on Computational learning theory, Santa Cruz, California, USA (1993) 51-57.
[16] S. Arnborg, B. Courcelle, A. Proskurowski, D. Seese, An algebraic theory of graph reduction, Journal of the ACM, 40(5) (1993) 1134-1164.
[17] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Hwang, Complex Networks: Structure and Dynamics, Physics Reports 424 (2006) 175-308.
[18] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of networks, Adv. Phys. 51 (2002) 1079-1187.
[19] V. Chvàtal, Tough graphs and Hamiltonian circuits, Discrete Math. 5 (1973) 215-228.