AUT Journal of Mathematics and Computing

AUT J. Math. Com., 3(2) (2022) 165-172
DOI: 10.22060/AJMC.2022.21692.1101

Original Article

New heuristics for burning connected graphs

Maryam Tahmasbi*[a], Zahra Rezai Farokh[a], Yousof Buali[a], Zahra Haj Rajab Ali Tehrani[a]

[a]*Department of Computer and Data Sciences, Shahid Beheshti University, Tehran, Iran*

ABSTRACT: The concept of graph burning and burning number ($bn(G)$) of a graph G was introduced recently [4]. Graph burning models the spread of contagion (fire) in a graph in discrete time steps. $bn(G)$ is the minimum time needed to burn a graph G. The problem is NP-complete. In this paper, we develop first heuristics to solve the problem in general (connected) graphs. In order to test the performance of our algorithms, we applied them on some graph classes with known burning number such as θ-graphs. We tested our algorithms on DIMACS and BHOSLIB that are known benchmarks for NP-hard problems in graph theory. We also improved the upper bound for burning number on general graphs in terms of their distance to cluster. Then we generated a data set of 1000 random graphs with known distance to cluster and tested our heuristics on them.

Review History:
Received:20 August 2022
Accepted:27 August 2022
Available Online:01 September 2022

Keywords:
Burning number
Heuristic
Distance to cluster
θ-Graphs
DIMACS
BHOSLIB

AMS Subject Classification (2010):
05C85; 90B10; 90-08

(Dedicated to Professor S. Mehdi Tashakkori Hashemi)

1. Introduction

Burning number of a graph is a new concept that measures the speed of spreading a contagion (fire) in a graph [4]. Given an undirected unweighted network, the fire spread in the network synchronous rounds as follows: in round one, a fire starts at a vertex called an activator. In each following round two events happen:

1. The fire spreads to all neighbors of nodes that are on fire.
2. Fire starts at a new activator that is an unburned vertex.

The process continues until all the vertices of the graph are on fire. At this time we say that the burning process is complete [12]. A burning schedule specifies a burning sequence of vertices where the i^{th} vertex in the sequence is the activator in round i. The burning number $bn(G)$ is the minimum length of a burning sequence.

Problem: burning number

Input: a simple graph G of order n and an integer $k \geq 2$.

Question: is $bn(G) \leq k$? In other words, does G contain a burning sequence $(x_1, x_2, \ldots, x_k)$?

As the first result, some of the properties of this problem including characterizations and bounds was presented in [4, 16]. Bonato et al. [4] proved that the burning number of any connected graph with n vertices is at most

*Corresponding author.
E-mail addresses: m_tahmasbi@sbu.ac.ir, rezaifarokhz@gmail.com, yousof.buali@gmail.com, doorsatehrani@yahoo.com

$2\sqrt{n}-1$ and conjectured that it is always at most $\lceil\sqrt{n}\rceil$. It is proved that this problem is NP-complete even when restricted to trees with maximum degree three, spider graphs and path-forests [1]. They developed polynomial-time algorithms for finding the burning number of spider graphs and path-forests if the number of arms and components, respectively, are fixed. They also generated a polynomial-time approximation algorithm with approximation factor 3 for general graphs [1]. Bonato and Lidbetter [7] developed a 3/2-approximation algorithm for path forests (disjoint union of paths). There is another approximation algorithm with an approximation ratio of 2 for trees [6].

In a recent study Kamali et al. [12] considered connected n-vertex graphs with minimum degree δ. They developed an algorithm that burns any such graph in at most $\sqrt{\frac{24n}{\delta+1}}$ rounds. In particular, for graphs with $\delta \in \theta(n)$, they proved that all vertices are burned in a constant number of rounds. More interestingly, even when δ is a constant that is independent of n, their algorithm answers the graph-burning conjecture in the affirmative by burning the graph in at most $\sqrt{n}$ rounds. Some results for different classes of graphs are presented in Table 1.

Šimon et al. [19] developed some heuristics for graph burning based on some centrality measures. They tested their heuristics on limited number of networks.

Table 1: Results from previous works

Graph Classes	Results	Reference
trees(maximum degree 3) spider graphs,path-forests	NP-completeness	[1]
spider graphs and path-forests	polynomial time algorithms	[1]
trees	2-approximation algorithm	[6]
graph products	exact value	[15]
petersen graph	exact value	[18]
theta graph	exact value	[14]
dense and tree-like graph	exact value	[11]
grid graph	exact value	[3]
graph with constant δ	algorithm with almost $\sqrt{\frac{24n}{\delta+1}}$ rounds	[12]
graph with pathlength pl and diameter d	algorithm with almost $\sqrt{d-1}+pl$ rounds	[12]
graphs with $\delta \in \theta(n)$	proved all vertices are burned in a constant number of rounds	[12]

In this paper, we develop new heuristic algorithms for solving graph burning problem. As mentioned before, most of the studies on this problem concern limited classes of graphs. Since the problem is modeling the spread of contagion in a network, it is essential to develop algorithms for solving the problem. We developed 6 heuristics for burning a graph. These heuristics differ in selecting the first activator and also the order of selecting the following activators.

Except for approximation algorithms [7, 11, 19], and algorithm 1 in [5] there are no official algorithms for this problem, so to test the performance of our algorithm, we used some theoretical results: we generated a random class of theta graphs and a random class of graphs with known distance to cluster and report the result of applying our algorithms on these classes. We compared our results with exact values and bounds reported in former studies. We also applied our algorithms on various graphs in known data sets: DIMACS and Benchmarks with Hidden Optimum Solutions for Graph Problems (BHOSLIB). These data sets contain graphs with various sizes and structures and are benchmarks for testing several NP-hard graph algorithms including but not limited to the maximum clique problem, maximum independent set, minimum vertex cover and vertex coloring.

This paper is organized as follows: in section 2 we present some basic definitions and describe our heuristics. In section 3 we present the result of our experimental study on different data sets. In section 4 we state conclusions and future works.

2. Algorithms

In this section, we present 6 heuristics for solving graph burning problem. The output in each algorithm is a burning sequence for the input graph G. First, we review some basic definitions and then we present our heuristics.

2.1. Basic Definitions

We review some basic definitions from graph theory [8]. For a graph $G = (V, E)$, let $V(G)$ and $E(G)$ denote the vertex set and edge set of G respectively. We use n and m to denote the number of vertices and edges in a graph

respectively. For a vertex $v \in V(G)$, $N(v)$ denotes the set of vertices adjacent to v and $N[v] = N(v) \cup \{v\}$ is the closed neighborhood of v.

The distance $d(u, v)$ between two vertices u and v in a graph G is the number of edges in a shortest path from u to v. Given an integer k, $N_k[v]$ is the number of vertices with distance at most k of v. This set is called the kth neighborhood of v. For a vertex v in a graph G, the eccentricity of v is defined as $max\{d(u, v)|u \in V(G)\}$. The radius of G is minimum eccentricity over the set of all vertices in G. The diameter of G is the maximum eccentricity over the set of all vertices in G. In other words, it is the distance between the farthest pair of vertices in G. For a subset $X \subset V(G)$, the graph $G[X]$ denotes the subgraph of G induced by vertices of X.

The theta graph $\theta(l_1, \ldots, l_m)$ is a graph consisting of m pairwise internally disjoint paths with common endpoints and lengths $l_1 \leq \cdots \leq l_m$ [9].

2.2. Heuristics

In the proposed algorithms we have two steps: the first step is to select the first candidate for burning. It seems essential since this vertex will burn vertices in distance $bn(G)$ of the graph. So, we need to select a vertex with a large set of vertices in $N_{bn(G)}[v]$.

In the second step, we select the rest of the activators one by one.

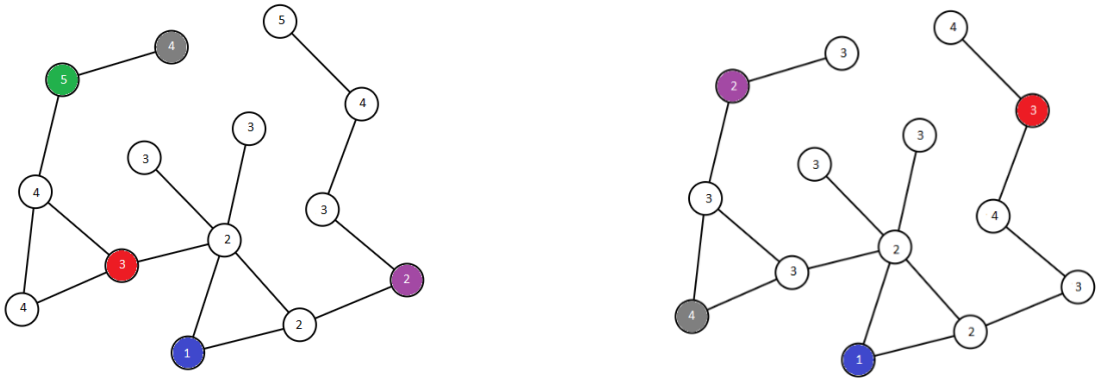
Given a burning sequence $S = (x_1, x_2, \ldots, x_{bn(G)})$ of a graph G, for each vertex v, there is a vertex x_i in S such that v is burned by a fire that is started in x_i, i.e. $d(v, x_i) < d(v, x_j)$ for all $j \neq i$. we call x_i the activator of v.

To reduce the length of a burning sequence, it is good to select activators such that each vertex has a unique activator.

We develop different heuristics based on different strategies for the first and second steps.

1. Based on arguments in former paragraphs, we can choose the first activator from the center of the graph. The farthest vertex to this vertex is in distance $rad(G)$ of it. So, it seems that this vertex has a big kth neighborhood. We used this in step one of heuristics *Ctr-Half dist.* and *Ctr-Far dist.*.
2. In each time k, for each unburned vertex v, we can calculate that in how many time steps this vertex will burn if we do not add any other activator. We call this "time-to-burn" of v and denote it by $t^k(v)$. Let $t^k = max\{t^k(v) : v \in V\}$. Hence, t is the maximum number of remaining time to burning the whole graph. We can select the next activator in two ways:
   (a) The next activator is a vertex v with $t^k(v) = t^k/2$. In this way the vertices with greater time-to-burn will burn in shorter time, using this new activator.
   (b) The next activator is a vertex with $t^k(v) = t^k - 1$.

   Figure 1 shows the two strategies. Heuristics *Ctr-Half dist.* and *Rnd-Half dist.* use the first and heuristics *Ctr-Far dist.* and *Rnd-Far dist.* use the second strategy for this step.
3. In heuristics *Rnd-Half dist.* and *Rnd-Far dist.* we select the first activator randomly to see the effect of selecting the first activator in our heuristics.



(a) Ctr-Half dist (b) Ctr-Far dist

Figure 1: Comparison of Ctr-Half dist and Ctr-Far dist. Activators are colored vertices and the number inside each vertex shows the time step it burns.

Table 2 summarizes the strategies in four heuristics.

We developed two other heuristics with a different idea: burning a path! The main idea is finding the diameter of the graph and the path with length $diam(G)$. Then burning the vertices of this path with the same order as

Table 2: Summary of first four heuristics

Heuristics	First Activator	Next Activator
Ctr-Half dist.	Center	$\frac{1}{2}$ time-to-burn
Ctr-Far dist.	Center	max time-to-burn
Rnd-Half dist.	Random	$\frac{1}{2}$ time-to-burn
Rnd-Far dist.	Random	max time-to-burn

burning a path in $\sqrt{diam(G)}$ steps. Since computing the diameter of a graph is of large complexity, we use two approaches to find a good approximation of it. In heuristic *DFS-path* we select a random vertex and perform a DFS algorithm to find a path. In heuristic *D-BFS-path* we use the algorithm by Birmele et al. [2] to approximate the diameter. This algorithm uses BFS twice, the first BFS starts from a random vertex and the second one starts from one of the leaves of previous BFS. It gives a 2-approximation of the diameter of the graph. There is no guarantee that all vertices of the graph burn using only vertices of these paths. So, after burning the vertices of the path, if there is still an unburned vertex, we select them randomly as activators.

3. Experimental Study

In this section, we describe the software platform, hardware, data and results considered in this matter. We implemented heuristics that were introduced and explained earlier in section 2 using Python 3. To model our graphs in a proper data structure and apply fundamental graph algorithms and measures, we used the well-known NetworkX package introduced by Hagberg et al. [10] in 2008.

These algorithms were evaluated on a hardware specification consisting of 4 units of Intel®Xeon®Processor E5-2680 v4 (56 cores) and 32GB of memory. We have not seen a critical need to parallelize the algorithms except for data. Instances of graphs were being evaluated in parallel using a simple data pool technique. We will explain our evaluation challenges later in this section.

3.1. Datasets

As mentioned before, there is no algorithm for burning general graphs. So, in order to evaluate our heuristics we use two types of datasets:

1. Classic datasets that are commonly used in some NP-hard problems in graph theory such as clique number, independence set, dominating number, etc.
2. Random graphs in some classes that the exact value or good bounds on their burning number is computed before.

In this section, we describe these datasets in more detail.

3.1.1. Classic datasets

These datasets are prepared for public use in the Network Repository website[1] [17]. We use graphs in DIMACS and BHOSLIB datasets. DIMACS dataset contains 78 large graphs with a maximum of 4000 vertices and over 5 million edges. BHOSLIB dataset contains 36 large graphs with a maximum of 4000 vertices and over 7 million edges.

3.1.2. Random θ-graphs

A $\theta-$graph consists of 3 (internally) disjoint paths between two vertices. In other words, a $\theta-$graph is a cycle with a disjoint path joining two vertices of it. We use the following simple technique to generate a random theta graph:

1. Generate a cycle C_m with a random number of vertices, m.
2. Choose two random vertices from the cycle.
3. Generate a path P_l of random size l to join these vertices.

[1] http://networkrepository.com/

3.1.3. Random graphs with fixed distance to cluster

Another class of graphs with known upper bound on burning number is the class of graphs with fixed distance to cluster. Distance to cluster is the minimum number of vertices that have to be deleted from G to get a disjoint union of complete graphs. This is an intermediate parameter between the vertex cover number and the clique-width/rank-width [13].

We generated 2000 random graphs in this class. Each of these graphs consists of k complete graphs with random sizes that are connected to a path of length d with a random number of edges. In fact, this is a special class of these graphs. We used the path structure of the d vertices, since paths have known burning number of $\sqrt{d}$ and this seems to be the maximum size in connected graphs. We generated these graphs using the following steps:

1. Generate a path P_d of length d.
2. Generate a set of k complete graphs with sizes $K = \{K_{n_i} \mid i \in [1, k]\}$.
3. For each complete graph, choose a random number a_i, $1 \leq a_i < n_i - 1$ and choose a_i random pairs of vertices (u, v) such that $u \in P_d$ and $v \in K_{n_i}$.
4. Connect P_d and complete graphs with pairs of vertices chosen above.

3.2. DIMACS, BHOSLIB

We applied our 6 heuristics on all graphs in DIMACS and BHOSLIB. From 78 graphs in DIMACS, all heuristics computed a burning sequence of length 3 for 71 graphs. Theorem 3.1 shows that this is optimal.

Theorem 3.1 ([16]). *Let G be a graph with n vertices. Then $bn(G) = 2$ if and only if $n \geq 2$ and G has maximum degree $n - 1$ or $n - 2$.*

Table 3 shows the results for those DIMACS instances with a burning sequence larger than 3. In these graphs, the average vertex degree and max degree are very near and the number of edges is small. This seems to be the reason of the growing burning number. Graphs c-fat200-* have 200 vertices and average degree grow from 15 to 84. As the average degree increases, the burning number decreases and different heuristics converge. The same is true for c-fat500-* graphs. On this benchmark, as the average degree grows, the heuristics find shorter burning sequences.

Table 3: DIMACS Results

Name	Vertices	Edges	Max deg.	Avg. deg.	Ctr-Half	Ctr-Far	Rnd-Half	Rnd-Far	DFS-path	D-BFS-path
c-fat200-1	200	1534	17	15	11	8	9	**7**	8	8
c-fat200-2	200	3235	34	32	6	6	6	**5**	**5**	6
c-fat200-5	200	8473	86	84	4	4	4	4	4	**3**
c-fat500-1	500	4459	20	17	12	11	12	**10**	15	17
c-fat500-10	500	46627	188	186	4	4	4	4	4	4
c-fat500-2	500	9139	38	36	9	**8**	9	**8**	11	**8**
c-fat500-5	500	23191	95	92	6	6	6	6	**5**	6

The results on BHOSLIB graphs are more interesting. All heuristics compute 3 for all graphs. We note that the average degree is very large compared with the number of vertices in these graphs. Since maximum vertex degree in these graphs is less than $n - 2$, by theorem 3.1, the burning sequence of length 3 is optimum for all of them.

3.3. $\theta-$Graphs

This set of graphs is generated by our algorithms described earlier in this section. Our evaluation observed results for 2000 $\theta-$graphs ranging from 400 to 900 vertices. There are tight bounds on the burning number of $\theta-$graphs that are proved by Liu and et al. [14]. They showed that the burning number of a theta graph with order $n = q^2 + r$ where $1 \leq r \leq 2q + r$ is either q or $q + 1$. In 1208 graphs (%60.4) the length of burning sequences in our heuristics meet bounds and in %81.7 of graphs, the difference is only one. The average difference between our best results and upper bounds is 0.6 and standard deviation 1.2309.

Table 4 shows the comparison of different heuristics. DFS-path finds the shortest burning sequence in 1476 graphs that is more than 73.8% of our graphs.

Table 5 shows some randomly chosen samples of our results. The expected burning number column is included in the table.

Table 4: Comparison of different heuristics on theta graphs

Heuristic	Success Rate
Ctr-Far	16.9%
Rnd-Far	7.7%
DFS-path	73.8%
D-BFS-path	1.6%

Table 5: Samples of $\theta-$graph Results

Name	Expected[14]	Ctr-Half	Ctr-Far	Rnd-Half	Rnd-Far	DFS-path	D-BFS-path
theta529-74-472-57	**24**	28	26	30	26	**24**	29
theta784-99-493-291	**29**	33	32	37	32	**29**	51
theta676-82-647-29	**27**	34	30	33	30	**27**	43
theta676-115-163-513	**27**	35	28	36	31	31	34
theta676-10-570-106	**27**	33	29	33	29	28	36

3.4. Graphs with fixed distance to cluster

This set of graphs is also generated randomly and has a name convention including characteristics of instances. Graph names start with "cluster" which reminds us of the graph type followed by the number of clusters (K_ns), minimum and maximum of random cluster size, number of vertices on P_d, number of the whole graph vertices and sample respectively. For this class of graphs, we evaluated our algorithms for 1000 instances varying from 50 to 100 clusters of size 4 to 20 and path length from 500 to 1000.

Kare et al. [13] computed an upper bound for graphs in terms of their distance to cluster, which is $3d+3$. We improve this bound in the following theorem.

Theorem 3.2. *Let G be a connected graph and A be a set of vertices such that $G[V(G)\backslash A]$ is a cluster graph. Then $bn(G) \leq bn(G[A]) + 2$.*

Proof. A burning sequence of A burns all vertices except possibly vertices of complete graphs that are adjacent to the last vertex of the burning sequence. These complete graphs burn in at most 2 rounds. So $bn(G) \leq bn(G[A])+2$. □

An immediate conclusion from theorem 3.2 is that $bn(G) \leq d+2$ for each connected graph with distance to cluster d, since the subgraph A in the proof of theorem 3.2 might be a set of d disjoint vertices.

Table 6: Samples of random graphs and their distance to cluster

Name	Distance to cluster	upper bound	Ctr-Half	Ctr-Far	Rnd-Half	Rnd-Far	DFS-path	D-BFS-path
cluster52-4-20-592-1170-0315	592	27	**11**	**11**	12	13	13	14
cluster78-4-20-571-1496-0210	571	26	11	**10**	11	11	13	12
cluster61-4-20-723-1498-0636	723	29	14	**11**	15	12	14	14
cluster10-4-20-759-886-0613	759	30	33	**30**	31	34	32	44
cluster13-4-20-861-1027-0447	861	32	32	29	33	**28**	32	33
cluster19-4-20-930-1190-0808	930	33	34	**31**	30	34	34	42
cluster73-4-20-512-1399-0496	512	25	**10**	**10**	12	12	13	13

As mentioned in section 3.1, we generated random graphs that each consists of a path of length d and some complete graphs that are connecting to this path with some edges. Since the number of edges between each complete

graph and the path is less than the number of vertices of the complete graph, the distance to cluster in this graph is d. Using theorem 3.2, the upper bound for burning number of these graphs is $\lceil\sqrt{d}\rceil + 2$. Our data set consists of 1000 graphs. We applied our heuristics on these graphs and compared the result with $\lceil\sqrt{d}\rceil + 2$. The results show that in %98 of graphs the results meet bounds. Table 6 shows some random graphs and the burning number computed by each heuristic.

We also compared all heuristics. Heuristics Ctr-Half dist, Rnd-Half dist and Rnd-Far dist find better solutions among our 6 heuristics. The winning heuristics that find the minimum solution in 1708 cases are the ones that select the first activator in different ways and the following ones according to far dist. strategy.

4. Conclusion and Future Work

In this paper, we developed the first heuristics for graph burning problem. To study the performance of our heuristics, we applied them on two types of datasets:

1. Known benchmarks for NP-hard problems in graph theory. We selected DIMACS and BHOSLIB. Our heuristics computed the optimal solution in 71 graphs out of 78 graphs in DIMACS, and all the 36 graphs in BHOSLIB.
2. Randomly generated graphs in classes with a known burning number. We generated 2000 $\theta-$graphs and applied our algorithms on them. Our heuristics succeeded to compute a burning sequence of length less than or equal to known upper bounds in 1208 graphs.
3. There is an upper bound on the burning number in terms of distance to cluster. We improved this bound and generated a special class of 2000 random graphs where each graph is a path of length d that is connected to a random number of disjoint complete graphs with some edges. In these graphs the distance to cluster is d and we proved that their burning number is at most $\lceil\sqrt{d}\rceil + 2$. In 1961 graphs, the burning number computed by our heuristics is less than or equal to this bound.

Since there are very few studies on algorithmic approaches to solve the burning number, of algorithmic approaches to solve the problem. We are interested in other algorithmic approaches to solve the problem such as local search algorithms. On the other hand, the problem has applications in social networks, that are usually disconnected graphs, we are going to develop heuristics for disconnected graphs.

Finally, there is a huge body of research on the spread of influence in social networks. There are measures called centrality measures to select seeds (activators) in a social network. It is interesting to develop algorithms for burning graph using these measures.

We presented the first results on burning number of different data sets of graphs. We hope that other algorithms will be developed for this problem and compare their results with ours.

References

[1] S. Bessy, A. Bonato, J. Janssen, D. Rautenbach, and E. Roshanbin, *Burning a graph is hard*, Discrete Applied Mathematics, 232 (2017), pp. 73–87.

[2] E. Birmelé, F. De Montgolfier, L. Planche, and L. Viennot, *Decomposing a graph into shortest paths with bounded eccentricity*, (2017).

[3] A. Bonato, K. Gunderson, and A. Shaw, *Burning the plane: densities of the infinite cartesian grid*, arXiv preprint arXiv:1806.05642, (2018).

[4] A. Bonato, J. Janssen, and E. Roshanbin, *Burning a graph as a model of social contagion*, in International Workshop on Algorithms and Models for the Web-Graph, Springer, 2014, pp. 13–22.

[5] ——, *How to burn a graph*, Internet Mathematics, 12 (2016), pp. 85–100.

[6] A. Bonato and S. Kamali, *Approximation algorithms for graph burning*, in International Conference on Theory and Applications of Models of Computation, Springer, 2019, pp. 74–92.

[7] A. Bonato and T. Lidbetter, *Bounds on the burning numbers of spiders and path-forests*, Theoretical Computer Science, 794 (2019), pp. 12–19.

[8] R. Diestel, *Graduate texts in mathematics*, Graph theory, 173 (2000).

[9] D. Eichhorn, D. Mubayi, K. O'Bryant, and D. B. West, *The edge-bandwidth of theta graphs*, Journal of Graph Theory, 35 (2000), pp. 89–98.

[10] A. Hagberg, P. Swart, and D. S Chult, *Exploring network structure, dynamics, and function using networkx*, tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[11] S. Kamali, A. Miller, and K. Zhang, *Burning two worlds: algorithms for burning dense and tree-like graphs*, arXiv preprint arXiv:1909.00530, (2019).

[12] ——, *Burning two worlds*, in International Conference on Current Trends in Theory and Practice of Informatics, Springer, 2020, pp. 113–124.

[13] A. S. Kare and I. V. Reddy, *Parameterized algorithms for graph burning problem*, in International Workshop on Combinatorial Algorithms, Springer, 2019, pp. 304–314.

[14] H. Liu, R. Zhang, and X. Hu, *Burning number of theta graphs*, Applied Mathematics and Computation, 361 (2019), pp. 246–257.

[15] D. Mitsche, P. Prałat, and E. Roshanbin, *Burning number of graph products*, Theoretical Computer Science, 746 (2018), pp. 124–135.

[16] E. Roshanbin, *Burning a graph as a model of social contagion*, PhD thesis, Dalhousie University, 2016.

[17] R. A. Rossi and N. K. Ahmed, *The network data repository with interactive graph analytics and visualization*, in AAAI, 2015.

[18] K. A. Sim, T. S. Tan, and K. B. Wong, *On the burning number of generalized petersen graphs*, Bulletin of the Malaysian Mathematical Sciences Society, 41 (2018), pp. 1657–1670.

[19] M. Šimon, L. Huraj, I. Dirgová Luptáková, and J. Pospíchal, *Heuristics for spreading alarm throughout a network*, Applied Sciences, 9 (2019), p. 3269.

Please cite this article using:

Maryam Tahmasbi, Zahra Rezai Farokh, Yousof Buali, Zahra Haj Rajab Ali Tehrani, New heuristics for burning connected graphs, AUT J. Math. Com., 3(2) (2022) 165-172
DOI: 10.22060/AJMC.2022.21692.1101