Anatolian Journal of Computer Sciences

Volume:5 No:2 2020 © Anatolian Science

pp:144-149 ISSN:2548-1304

Efficient Algorithms for Determining the Maximum Independent Sets in Graphs

Ali KARCI
İnönü University, Department of Computer Engineering, ali.karci@inonu.edu.tr

Received Date : May 30, 2020.
Acceptance Date : :Jun. 23, 2020.
Published Date : Dec. 1, 2020

Abstract. It is known that there are many NP-hard and NP-complete problems in graph theory. The aim of this paper to solve the maximum independent set which is an NP-Hard and NP-Complete problem. In order to solve maximum independent set for given graph, a special spanning tree which is defined in this paper for the first time, is solved to obtain the fundamental cut-sets. The fundamental cut-sets are used to determine the effects of nodes. These effects of nodes are used to determine the elements of maximum independent set.

Keywords: *Spanning Tree, Fundamental Cut-Sets, Efficient Algorithms, Effects of Nodes.*

1.INTRODUCTION

The graphs are mathematical models to solve the modelled problems such as computer networks, mathematical equations, object-oriented design, social networks, etc. A graph can be defined as in Definition 1.

Definition 1*: A graph G = (V,E) consists of a set V of vertices and a set E of edges. A graph, which does not consist of parallel and loop edges, called simple graph.*

The main of this study is to solve maximum independent set problem in graphs with efficient algorithms. There are many studies in literature for solving maximum independent set problems in graphs.

Assume that G=(V,E) is a simple graph where V is a set of nodes (vertices) and E is a set of edges ($E \subseteq VxV$). A node v_i is said to be neighbour of v_j if $(v_i,v_j) \in E$. $I \subseteq V$, $\forall v_i,v_j \in I$, $v_i \notin N(v_j)$ where $N(v_j)$ is the set of nodes which are neighbours of v_j, I is called as independent set for graph G. Assume that $I_2 \subseteq V$, $\forall v_i,v_j \in I_2$, $v_i \notin N(v_j)$, if there is no such $I \subset I_2$, I is called maximal independent set. In another word, independent set (stable set, coclique, anticlique) is a set of nodes in the corresponding graph (so called G), no two of which are adjacent.

It is known that Independent Set problem is an NP-Hard problem, and there are many studies on Independent Set problem. Brandstadt and Mosca (Brandstadt and Mosca, 2018) used dynamic programming approach and shown that the maximum weight independent set can be solved in polynomial time for $\mathcal{A}$claw-free graphs (fixed $\mathcal{A}$). Laflamme and his friends (Laflamme and et al, 2019) tried to show that Kn-free graph and minimal r=r(G,m) where $m \in N$, independent set meets at least m colour classes in a set of size |V| for any balanced r-colouring of the vertices of graph G. Lin (Lin, 2018a) tried to obtain the number of independent sets and maximal independent sets in path-tree bipartite graphs, which are a subclass of bipartite graphs between tree convex bipartite graphs and convex bipartite graphs. Oh (Oh, 2017) studied on the number of maximal independent sets on a complete rectangular grid graph. Beside on this, Oh studied on recursive matrix-relation producing the partition function and asymptotic behaviour of the maximal hard square entropy. Wan and his friends (Wan et al, 2018) tried to solve problem of independent sets and matchings of given sizes in graphs of order n and tree-width at most p by proposing two dynamic programming approaches. Lin and Chen (Lin and Chen, 2017) developed some linear-time algorithms for independent sets counting, determination of maximal independent sets, independent perfect dominating sets in bipartite permutation graphs. Lin (Lin, 2018b) developed linear-time algorithms for counting independent sets and their two variants, independent dominating sets, independent perfect dominating sets for

distance-hereditary graphs. Jarden and his friends (Jarden et al, 2018) presented many various relations between unions and intersections of maximum and critical independent sets of a graph concluding in new characterization of König-Egevary graphs. Sah and his friends (Sah et al, 2019) proved some limitations for the cardinality of independent sets in graphs which do not consist of isolated nodes, and they illustrated these limitations for some example of graphs. Peramau and Perkins (Peramau and Perkins, 2018) proved a tight upper bound on the number of independent sets of cubic graphs of girth at least 5. Wan and his friends (Wan et al, 2020) took the number of independent sets and matchings in consideration for graph entropy measures.

The aim of this paper is to determine the effective and ineffective nodes by using spanning tree and fundamental cut-sets of given graph as done in (Karci and Karci, 2020; Karci, 2020). The spanning trees used in this study were defined for the first time.

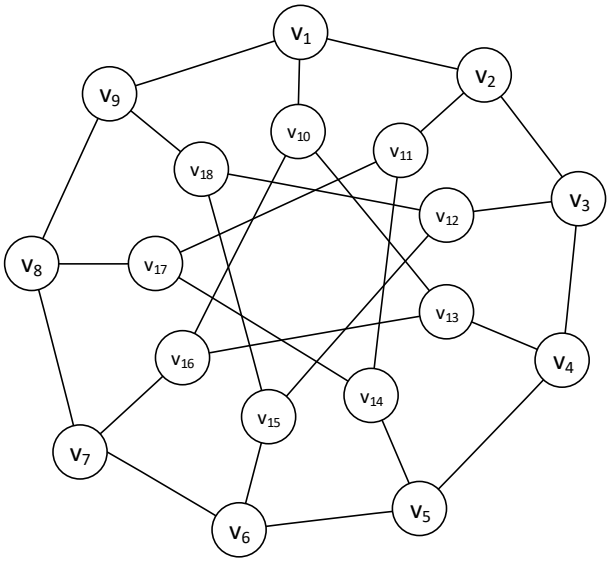

Figure 1. Any graph G=(V,E).

2.NEW SPANNING TREE TYPE AND EFFICIENT ALGORITHMS FOR CONSTRUCTION

An acyclic graph does not include cycle and a connected acyclic graph is called tree, otherwise it is called forest (forest is outside of scope of this study). A spanning tree of a connected graph G is a tree of having the all nodes of graph G (Definition 2).

Definition 2: *A spanning tree is a subset of graph G, which has all the vertices covered with minimum possible number of edges.*

In this study, we will define a special spanning tree of given graph whose construction is based on breadth first search technique. This tree is used to construct fundamental cut-sets where defining any path in graph uses at least any/some edges in at least one fundamental cut-set. Due to this case, determination of effect of any node requires using fundamental cut-sets, node degree in graph, and node degree in corresponding spanning tree.

Breadth-first search is a search technique in artificial intelligence for investigation of solution/goal. Breadth-first search consists of searching through a tree one level at a time, and then going to next down level for searching, and so on.

G=(V,E) is a given graph where $V=\{v_1,v_2,\ldots,v_n\}$. The set V is sorted with respect to the node degrees of nodes in V in ascending order. Any node with minimum degree (assume it is v_i) is selected as root node for spanning tree T of given graph G. The node in $N(v_i)$ are added to spanning tree T as children of v_i. The children of v_i are expanded from minimum degree to maximum degree. The obtained tree is called as **Kmin Tree** (Karcı Minimum Tree).

Algorithm 1: Construction of Kmin Tree
1. G=(V,E) and G is a simple graph and connected 2. Sorting the set V with respect to node degrees in ascending order, and V_1=sort(V). 3. T=(V_T,E_T) and V_T=∅, E_T=∅. 4. Assume v=V_1(1) is the first element of V_1, V_T={v} and V_1=V_1-{v}. 5. While V_1≠∅ 6. For all u_i∈N(u), u∈V_T, u is leaf in current form of T, sort N(u) in ascending order, assume u_j is the node of minimum degree in N(u), and E_T=E_T∪{(u,n_j)}, V_T=V_T∪{n_j}, V_1=V_1-{u_j}, N(u)=N(u)-{u_j}.

Algorithm 2 can be used for construction of Kmin Tree. The idea of Algorithm 2 to construct Kmin Tree is the node of minimum degree being root or one of nodes of minimum degrees being root of tree. Then adding the neighbours of root to Kmin tree. The child of root with minimum degree (expandable degree) is expanded first. This process continuous until all nodes are added to tree. Fig.3 illustrates Kmin Tree of graph seen in Fig.1, and it is also a spanning tree.

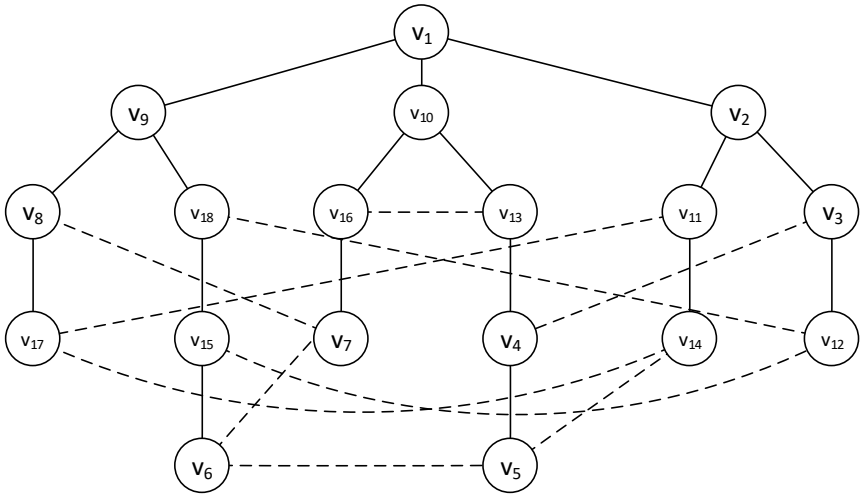

Figure 2. Kmin Tree of graph in Fig.1 (dashed edges are not part of tree).

Fig.1 contains a graph G=(V,E) where V={v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{14},v_{15},v_{16},v_{17},v_{18}} and E={(v_1,v_2), (v_1,v_9), (v_1,v_{10}), (v_2,v_3), (v_2,v_{11}), (v_3,v_4), (v_3,v_{12}), (v_4,v_5), (v_4,v_{13}), (v_5,v_6), (v_5,v_{14}), (v_6,v_7), (v_6,v_{15}), (v_7,v_8), (v_7,v_{16}), (v_8,v_9), (v_8,v_{17}), (v_9,v_{18})}. The construction of Kmin Tree can be explained in the following steps:

Step 1: The all nodes have equal degrees and v_1 is added to Kmin Tree.

Step 2: Expand v_1 and add nodes in N(v_1) to Kmin Tree as second level, and N(v_1)={v_2,v_9,v_{10}}.

Step 3: The degrees of v_2,v_9 and v_{10} are equal and expansion sequence is not important. They can be expanded in any sequence. N(v_2)∪N(v_9)∪N(v_{10})={v_8,v_{18},v_{16}, v_{13},v_{11},v_3} are added to Kmin tree as third level nodes.

Step 4: v_{13} and v_{16} have minimum remaining degrees, and they are expanded first. Expansions of both nodes make remaining degrees of other nodes decrease. So, they are expanded from minimum to maximum remaining degrees. If they have different number of children, the node of less children will be expanded firstly, then the other (obtained tree is seen in Fig.2).

3. AN EFFICIENT ALGORITHM FOR EFFECTIVENESS OF NODES

Assume G=(V,E) is a graph, and connected, E_c⊆E is a set of edges whose removal from graph G concludes in G is not connected, then E_c is a cut-set for G. There are some special cut-sets for graph G and the remaining cut-sets can be represented as a linear combination of these cut-sets by applying ring-sum operator, then these cut-sets are called fundamental cut-sets of graph G. Assume that T is a spanning tree of graph G=(V,E), T=(V,E_1) and E_1⊆E. All edges in T are called branches, and all edges are not included in T are called chords (all these edges are included in graph G). The fundamental cut-sets are defined by using the spanning tree. If |V|=n, then there are n-1 fundamental cut-sets.

Definition 3: *The fundamental cut-set of a connected graph G contains exactly one branch of corresponding spanning tree T, and remaining are chords.*

In order to determine the ineffective nodes in graph G, Kmin Tree is used. This tree is a spanning tree of given graph G. In order to determine the node types, fundamental cut-sets can be used. It is known that all cut-sets can be represented as linear combinations of fundamental cut-sets by using ring-sum operator. This means that all paths in this graph should include fundamental cut-sets, and this inclusion determines the ineffectiveness of nodes which are part of cut-sets. At this aim, the fundamental cut-sets should be obtained. There are two types of fundamental cut-sets with respect to given spanning tree, and they are explained in Algorithm 2.

Algorithm 2: Construction of Fundamental Cut-Sets
1) Assume that G=(V,E) and T=(V,E_1) is a Kmax Tree/Kmin Tree of graph G. 2) For all leaves of T, construct leaf cut-set: Assume $v_i \in V$ and it is a leaf of T. The edges coincide on v_i constitute the first type fundamental cut-set (**leaf fundamental cut-set**). 3) Otherwise, assume that (v_i,v_j) is a branch in T. v_i and the all nodes reachable from v_i in T are added to V_1 and remaining are added to V_2 ($V_1 \cup V_2=V$, and $V_1 \cap V_2=\varnothing$) All edges whose end points are not in V_1 or V_2; one of them is in V_1 and the other is in V_2, constitute **internal fundamental cut-set**.

This algorithm can be used to determine ineffective node by using Kmin Tree. Fig.4 illustrates the results of algorithm for Kmin Tree.

Leaf Fundamental Cut-Sets: C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16} and C_{17}.

C_1={(v_{17},v_{11}), (v_{17},v_8), (v_{17},v_{14})}
C_2={(v_6,v_{15}), (v_6,v_7), (v_6,v_5)}
C_3={(v_7,v_{16}), (v_7,v_6), (v_7,v_8)}
C_4={(v_5,v_4), (v_5,v_6), (v_5,v_{14})}
C_5={(v_{14},v_{11}), (v_{14},v_5), (v_{14},v_{17})}
C_6={(v_{12},v_3), (v_{12},v_{18}), (v_{12},v_{15})}

Internal Fundamental Cut-Sets: C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16} and C_{17}.

C_7={(v_8,v_9), (v_8,v_7), (v_{17},v_{11}), (v_{17},v_{14})}
C_8={(v_{18},v_{15}), (v_{15},v_{12}), (v_6,v_7), (v_6,v_5)}
C_9={(v_{18},v_{15}), (v_{15},v_{12}), (v_6,v_7), (v_6,v_5), (v_9,v_{18}), (v_{18},v_{12})}
C_{10}={(v_1,v_9), (v_{18},v_{12}), (v_8,v_7), (v_6,v_7), (v_{15},v_{12}), (v_{17},v_{11}), (v_{17},v_{14}), (v_6,v_5)}
C_{11}={(v_{10},v_{16}), (v_{16},v_{13}), (v_8,v_7), (v_6,v_7) }
C_{12}={(v_{13},v_4), (v_6,v_5), (v_5,v_{14}), (v_4,v_3) }
C_{13}={(v_{13},v_4), (v_6,v_5), (v_5,v_{14}), (v_4,v_3), (v_{10},v_{13}), (v_{16},v_{13})}
C_{14}={(v_1,v_{10}), (v_8,v_7), (v_6,v_7), (v_4,v_3), (v_6,v_5), (v_5,v_{14})}
C_{15}={(v_2,v_{11}), (v_{17},v_{11}), (v_5,v_{14}), (v_{17},v_{14}) }
C_{16}={(v_2,v_3), (v_4,v_3), (v_{18},v_{12}), (v_{15},v_{12}) }
C_{17}={(v_1,v_2), (v_{18},v_{12}), (v_4,v_3), (v_{17},v_{11}), (v_5,v_{14}), (v_{17},v_{14}), (v_{15},v_{12})}

The appearance of each node in the fundamental cut-sets constitutes the **Cut-Set Effectiveness** of corresponding node. Each node has a node degree in graph, and this is called **Graph Effectiveness** of corresponding node. Each node has a node degree in spanning tree, and this called the **Spanning Effectiveness** of corresponding node. The effectiveness of a node can be obtained by summation of these three effectiveness parameters. In order to determine the ineffectiveness of a node, Kmin Tree is used as a spanning tree (the node with less effectiveness value is regarded as ineffectiveness node).

Before giving an algorithm to compute ineffectiveness of nodes, the incidence matrix for an undirected graph should be explained. Assume that B is the incidence matrix for graph G=(V,E) where |V|=n, and |E|=m. The incidence matrix B is an nxm matrix such that B_{ij}=1 is the node v_i and edge e_j are incident and 0 otherwise. The cut-set C_{nxm} matrix is a submatrix of B such that C_{ij}=1 is the cut-set C_i contains edge e_j 0 otherwise.

Algorithm 3: Computing Ineffectiveness

1) Assume that G=(V,E) and T=(V,E_1) is a Kmin Tree of graph G.
2) B is the incidence matrix, and C_{min} corresponds to Kmin Tree.
3) $E_{min} = B * C_{min}^T$ where E_{min} corresponds to Ineffectiveness and C_{min}^T is the transpose of C_{min}.
4) i←1,…,n
5) $\mu(v_i) = 0$
6) j←1,…,m
7) $\mu(v_i) = \mu(v_i) + E_{min}(i,j)$
8) i←1,…,n
9) $\mu(v_i) = \mu(v_i) + d_G(v_i) + d_T(v_i)$ where $d_G(v_i)$ is the node degree of v_i in G, $d_T(v_i)$ is the node degree in Kmax Tree (Kmin Tree).

i.e. $\mu(v_i)$=Cut-Set Ineffectiveness+Graph Effectiveness+Spanning Effectiveness.

After application of Algorithm 3 to any graph G with respect to Kmin Tree, μ vector contains the ineffectiveness values of nodes. The minimum value in μ corresponds to the most ineffective node. μ will be used for computing independent set of a given graph. μ will be also used to compute the independent set of a complement graph of given graph and this result corresponds to max clique of given graph. All these algorithms are polynomial algorithms, so, there are efficient algorithms for determining effective and ineffective nodes in given graph.

Example: Graph illustrated in Fig.1 can be used to depict the application of these algorithms. Algorithm 3 was iterated to obtain ineffectiveness values for all nodes and one node was selected at each iteration. The edges incident on selected node were removed from graph and corresponding Kmin tree was updated respectively. The last column of Table 1 illustrates the selected node at corresponding iteration. The nodes have equal ineffective values are subject to selection according to node with less branches incident on.

Table 1. Application of Algorithm 3 to graph seen in Fig.1.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	S
μ	9	9	12	12	20	21	**7**	12	9	9	12	16	9	18	13	9	16	12	v_7
	$N(v_7)=\{v_6,v_8,v_{16}\}$																		
	9	9	12	12	20	15	--	7	9	9	12	15	9	18	12	7	16	12	v_{13}
	$N(v_7)\cup N(v_{13})=\{v_6,v_8,v_{16},v_4,v_{10}\}$																		
	9	9	9	8	12	11	--	7	9	8	12	16	--	13	13	6	16	12	v_3
	$N(v_7)\cup N(v_{13}) \cup N(v_3)=\{v_6,v_8,v_{16},v_4,v_{10},v_2,v_{12}\}$																		
	9	8	--	6	12	11	--	7	9	8	11	11	--	12	11	6	14	10	v_9
	$N(v_7)\cup N(v_{13})\cup N(v_3)\cup N(v_9)=\{v_6,v_8,v_{16},v_4,v_{10},v_2,v_{12},v_1,v_{18}\}$																		
	8	8	--	6	10	9	--	6	--	8	10	7	--	11	9	6	12	7	v_{15}
	$N(v_7)\cup N(v_{13})\cup N(v_3)\cup N(v_9)\cup N(v_{15})=\{v_6,v_8,v_{16},v_4,v_{10},v_2,v_{12},v_1,v_{18}\}$																		
	8	8	--	6	7		--	6	--	8	7		--	7	--	6	6		v_5
	$N(v_7)\cup N(v_{13})\cup N(v_3)\cup N(v_9)\cup N(v_{15})\cup N(v_5)=\{v_6,v_8,v_{16},v_4,v_{10},v_2,v_{12},v_1,v_{18},v_{14}\}$																		
	8	8	--	6	6		--		--	8	7		--	6	--	6	6		v_{17}
	$N(v_7)\cup N(v_{13})\cup N(v_3)\cup N(v_9)\cup N(v_{15})\cup N(v_5)\cup N(v_{17})=\{v_6,v_8,v_{16},v_4,v_{10},v_2,v_{12},v_1,v_{18},v_{14},v_{11}\}$																		

The maximum independent set for graph seen in Fig.1 is $\{v_7,v_{13},v_3,v_9,v_{15},v_5, v_{17}\}$, and resulting graph is seen in Fig.3.

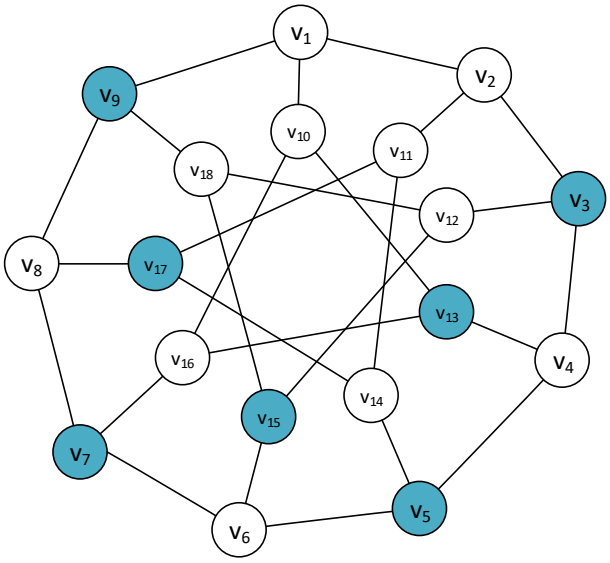

Figure 3. The maximal independent set for graph seen in Fig.1.

4.CONCLUSIONS

This study includes some important fundamentals for solving NP-Hard and NP-Complete problems in given graphs. At this aim, there is a new spanning trees and its construction algorithm was introduced in this paper. By using this tree, the fundamental cut-sets of graph can be obtained. These sets, spanning tree and graph will be used to solve maximal independent set.

REFERENCES

Brandstadt, A., Mosca, R., "Maximum weight independent set for $\mathscr{A}$claw-free graphs in polynomial time", Discrete Applied Mathematics, Vol:237, pp:57-64, 2018.

Jarden, A., Levit, V.E., Mandrescu, E.,"Critical and maximum independent sets of a graph", Discrete Applied Mathematics, Vol: 247, pp:127-134, 2018.

Karci, A., Karci, Ş.,"Determination of Effective Nodes in Graphs", International Conference on Science, Engineering & Technology, Mecca, Saudi Arabia, pp:25-28, 2020.

Karci, A., "Finding Innovative and Efficient Solutions to NP-Hard and NP-Complete Problems in Graph Theory", Anatolian Science – Journal of Computer Science, 2020.

Laflamme, C., Aranda, A., Soukup, D.T., Woodrow, R.,"Balanced independent sets in graphs omitting large cliques", Journal of Combinatorial Theory, Series B, Vol:137, pp:1-9, 2019.

Lin, M.-S.,"Counting independent sets and maximal independent sets in some subclasses of bipartite graphs", Discrete Applied Mathematics, Vol:251, pp:236-244, 2018a.

Lin. M.-S., "Simple linear-time algorithms for counting independent sets in distance-hereditary graphs", Discrete Applied Mathematics, Vol: 239, pp:144-153, 2018b.

Lin, M.-S., Chen, C.-M.,"Linear-time algorithms for counting independent sets in bipartite permutation graphs", Information Processing Letters, Vol:122, pp:1-7, 2017.

Oh, S., "Maximal independent sets on a grid graph", Discrete Mathematics, Vol:340, pp:2762-2768, 2017.

Peramau, G., Perkins, W.,"Counting independent sets in cubic graphs of given girth", Journal of Combinatorial Theory, Series B, Vol:133, pp:2018.

Sah, A., Sawhhney, M., Stoner, D., Zhao, Y.,"The number of independent sets in an irregular graph", Journal of Combinatorial Theory, Series B, Vol:138, pp:172-195, 2019.

Wan, P., Tu, J., Zhang, S., Li, B., "Computing the numbers of independent sets and matchings of all sizes for graphs with bounded treewidth", Applied Mathematics and Computation, Vol:332, pp:42-47, 2018.

Wan, P., Chen, X., Tu, J., Dehmer, M., Zhang, S., Emmert-Streib, F.,"On graph entropy measures based on the number of independent sets and matchings", Information Sciences, Vol:516, pp:491-504, 2020.