Anatolian Journal of Computer Sciences

Volume:5 No:2 2020 © Anatolian Science

pp:137-143 ISSN:2548-1304

Finding Innovative and Efficient Solutions to NP-Hard and NP-Complete Problems in Graph Theory

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

Received Date : Apr. 07, 2020.
Acceptance Date : Apr. 16, 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 prepare some basic methods for solving such problems (min dominating set, max independent set, max clique, etc.). In order to construct such fundamentals, the effectiveness and ineffectiveness of all nodes in the given graph are computed. Then these values will be used in solving NP-Hard problems of graphs.

Key Words: *Dominating Sets, Fundamental Cut-Sets, Efficient Algorithms, Independent Sets, Max Cliques.*

1.Introduction

There are many methods which are used for problem solving in science and engineering problems such as differential equations, Bayesian networks, graphs, etc. Graph is a mathematical model for solving problems in science and engineering. Especially, the events, objects, etc. and their relationships cam be modelled by using graphs. The mathematical model graph can be defined as follow.

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.*

There are many graph problems in literature such as independent sets, dominating set, minimum vertex covers sets, maximum clique, etc. and all of them are NP-complete problems.

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 clique in a graph G is subgraph H of G where H is a complete graph. A complete graph $K_n=(V,E)$ is a graph where $\forall v_i,v_j \in V$, $(v_i,v_j) \in E$, $i \neq j$. A maximum clique is a clique such that there is no clique with more nodes. 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. A set $D \subseteq V$ of vertices in G is a dominating set if each vertex in the complement of D is adjacent to at least one vertex in D. The minimum cardinality of D is called dominating number of G.

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 Lclaw-free graphs (fixed L). 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. The dominating-χ-color number of graph G is the dominating sets among all $\chi(G)$-colourings of a graph G and is denoted by $d\chi(G)$. Bresar and Movarraei (Bresar and Movarraei, 2018) tried to prove that the dominating-χ-colour number equals to 1 if and only if its domination number is greater than 2 in split graphs. 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 treewidth 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.

Bron and Kerbosch (Bron and Kerbosch, 1973) developed an algorithm to find all cliques in a graph. There are some other studies on determining cliques and/or using cliques. Naude (Naude, 2016) investigated pivoting Bron-Kerbosch algorithm to determine all cliques in a given undirected graph. Yu and Liu (Yu and Liu, 2017) developed a linear-time algorithm for candidate map constructor for maximal clique enumeration in large sparse graphs which generates a new candidate map as a result.

There are many studies on dominating set finding. The construction of dominating set for given graphs were studied in (Alikan and Peng, 2013). Some researchers dealt with the bounds on the maximum number of minimum dominating sets (Connolly and et al, 2016). The uniform clutters and dominating sets are aim of another study in literature (Marti-Farre et al, 2019). The known exact algorithm for dominating set has complexity as O(1.4957n) (Rooij and Bodlaender, 2011). The study on independent domination is another literature study (Goddard and Henning, 2013). There is not unique minimal dominating set for all types of graphs, and enumerating the minimal dominating sets is another study (Golovach et al, 2016). There are some studies for obtaining efficient algorithm in certain graphs' types such as efficient sets in circular graphs (Deng et al, 2017). These are some of studies performed by researchers and there are many remaining studies on this concept.

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). The spanning trees used in this study were defined for the first time.

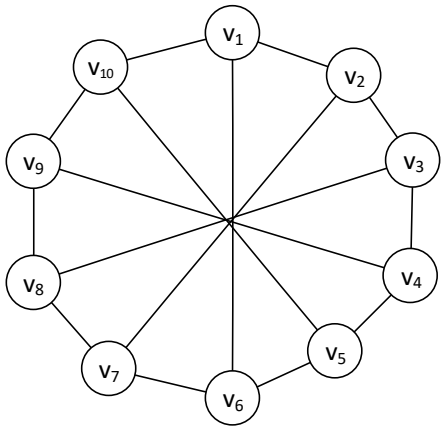

Figure 1. A circulant graph G=(V,E).

2.New Spanning Tree Types 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: *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 two special spanning trees of given graph and by using these spanning trees, the effective and ineffective nodes can be determined. So, there are data structures for given graph, and these data structures will be used to obtain fundamental cut-sets in the subsequent section.

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.

***Definition 3**: 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 descending order. Any node with maximum 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 maximum degree to minimum degree. The obtained tree is called as **Kmax Tree** (Karcı Maximum Tree).*

Algorithm 1: Construction of Kmax/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 descending/ascending order, and V_1=sort(V).
3. $T=(V_T,E_T)$ and $V_T=\emptyset$, $E_T=\emptyset$.
4. Assume $v=V_1(1)$ is the first element of V_1, $V_T=\{v\}$ and $V_1=V_1-\{v\}$.
5. While $V_1\neq\emptyset$
6. For all $u_i\in N(u)$, $u\in V_T$, u is leaf in current form of T, sort N(u) in descending/ascending order, assume u_j is the node of maximum/minimum degree in N(u), and $E_T=E_T\cup\{(u,n_j)\}$, $V_T=V_T\cup\{n_j\}$, $V_1=V_1-\{u_j\}$, $N(u)=N(u)-\{u_j\}$.

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

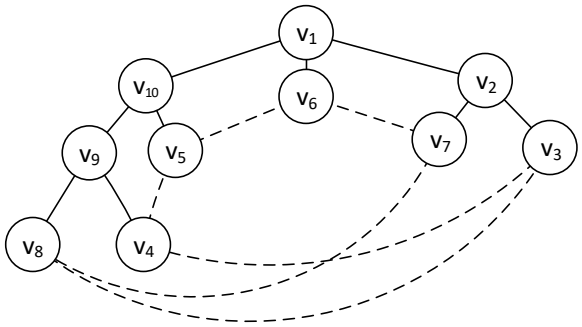

Figure 2. Kmax Tree of graph in Fig.1 (dashed edges are not part of tree).

Fig.1 contains a graph G=(V,E) where $V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10}\}$ and $E=\{(v_1,v_2), (v_2,v_3), (v_3,v_4), (v_4,v_5), (v_5,v_6), (v_6,v_7), (v_7,v_8), (v_8,v_9), (v_9,v_{10}), (v_{10},v_1), (v_1,v_6), (v_2,v_7), (v_3,v_8), (v_4,v_9), (v_5,v_{10})\}$. The construction of Kmax Tree can be explained in the following steps:

Step 1: The nodes of maximum degree is selected as root (all nodes have equal degrees), and v_1 can be selected as root.

Step 2: Expanding v_1 yields the second level of Kmax tree and add nodes in $N(v_1)=\{v_{10},v_6,v_2\}$ to Kmax Tree.

Step 3: The nodes v_{10}, v_6, and v_2 have same degrees, so, v_{10} is expanded and nodes v_9, v_5 are added to Kmax Tree for third level, then the remaining degree of v_6 is one and remaining degree of v_2 is 2. So, v_2 is expanded and $N(v_2)=\{v_7,v_3\}$ are added to Kmax tree as third level nodes. There is no remaining degree of v_6.

Step 4: Finally, v_9 is expanded and $N(v_9)=\{v_8,v_4\}$ are added to Kmax Tree as fourth level nodes. Then there is no remaining node, and construction of Kmax Tree is concluded (obtained tree is seen in Fig.2).

***Definition 4**: 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 2: 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 as next level nodes. 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.

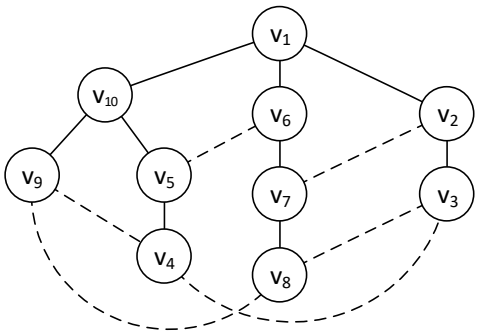

Figure 3. Kmin Tree of graph in Fig.1 (dashed edges are not part of tree).

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}} and E={(v_1,v_2), (v_2,v_3), (v_3,v_4), (v_4,v_5), (v_5,v_6), (v_6,v_7), (v_7,v_8), (v_8,v_9), (v_9,v_{10}), (v_{10},v_1), (v_1,v_6), (v_2,v_7), (v_3,v_8), (v_4,v_9), (v_5,v_{10})}. The construction of Kmax Tree can be explained in the following steps:

Step 1: All nodes of graph in Fig.1 have equal degrees, and v_1 is selected as root node and added to Kmin Tree.

Step 2: Expand v_1 and add nodes in N(v_1)={v_{10},v_6,v_2} to Kmin Tree as second level nodes.

Step 3: The nodes v_2, v_6 and v_{10} have remaining degrees as 2. Expanding v_{10} decreases remaining degree of node v_6 to 1. Then v_6 is expanded and finally v_2 is expanded. N(v_{10})={v_9,v_5}, N(v_6)={v_7} and N(v_2)={v_3} are added to Kmin Tree as third level nodes.

Step 4: The nodes in third level are v_9, v_5, v_7 and v_3. v_9 and v_3 have remaining degrees as 2 and v_5 and v_7 have remaining degrees as 1. Due to this case, v_5 is expanded and after that v_7 is expanded. N(v_5)={v_4} and N(v_7)={v_8} are added to Kmin Tree as fourth level nodes. There is no remaining node and algorithm 2 is terminated (obtained tree is seen in Fig.3).

3. An Efficient Algorithm for Obtaining Effectiveness / Ineffectiveness

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, 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 5: *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 effective / ineffective nodes in graph G, Kmax Tree / Kmin Tree are used. Both of trees are spanning trees of given graph G. In order to determine the node types, fundamental cut-sets can be used. It ıs 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 effectiveness/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.

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

Leaf Fundamental Cut-Sets (dashed red line): C_1, C_2, C_3, C_4, C_5, C_6.

$C_1=\{(v_8,v_9), (v_8,v_7), (v_8,v_3)\}$
$C_2=\{(v_4,v_9), (v_4,v_5),(v_4,v_3)\}$
$C_3=\{(v_5,v_4), (v_5,v_6), (v_5,v_{10})\}$
$C_4=\{(v_6,v_1), (v_6,v_5), (v_6,v_7)\}$
$C_5=\{(v_7,v_2), (v_7,v_6), (v_7,v_8)\}$
$C_6=\{(v_3,v_2), (v_3,v_4), (v_3,v_8)\}$

Internal Fundamental Cut-Sets (dashed green line): C_7, C_8, C_9.

$C_7=\{(v_9,v_{10}), (v_4,v_5),(v_4,v_3),(v_8,v_7),(v_8,v_3)\}$
$C_8=\{(v_1,v_{10}), (v_5,v_6), (v_4,v_3),(v_8,v_7),(v_8,v_3)\}$
$C_9=\{(v_1,v_2),(v_6,v_7),(v_4,v_3),(v_8,v_7),(v_8,v_3)\}$

Algorithm 3: 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 leafs 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.**

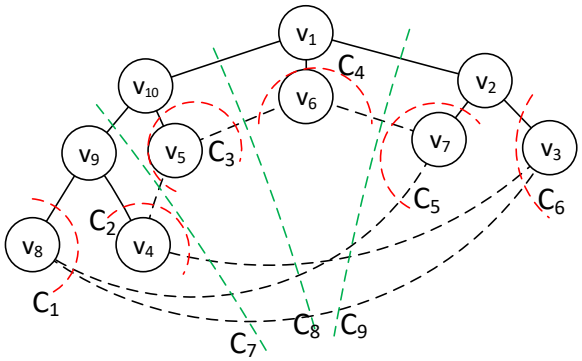

Figure 4. Fundamental cut-sets by using Kmax Tree (red cut-sets are leaf fundamental cut-sets; green cut-sets are internal fundamental cut-sets).

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 obtain effectiveness of a node, Kmax Tree is used as spanning tree. In order to determine the ineffectiveness of a node, Kmin Tree is used as a spanning tree.

Before giving an algorithm to compute effectiveness and 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.

After application of Algorithm 4 to any graph G with respect to Kmax Tree (Kmin Tree), π vector contains the effectiveness of nodes, and μ vector contains the ineffectiveness of nodes. The maximum value in π corresponds to the most effective node and the minimum value in μ corresponds to the most ineffective node. π will be used for computing dominating set and 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.

Algorithm 4: Computing Dominance Values

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

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. $\pi(v_i)$ =Cut-Set Effectiveness+Graph Effectiveness+Spanning Effectiveness.

4.Conclusions

This study includes some important fundamentals for solving NP-Hard and NP-Complete problems in given graphs. At this aim, there are two new spanning trees and their construction algorithms were introduced in this paper. By using these trees, the fundamental cut-sets of graph can be obtained. These sets, spanning tree and graph will be used to solve maximum independent set, minimum dominating set, maximum clique problems in the future studies.

References

Alikhan, S., Peng, Y.-H., "Construction of Dominating Sets of Certain Graphs", Journal of Discrete Mathematics, Vol:2013, Article ID:587196, 2013.

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

Bresar, B., Movarraei, N., "On the number of maximal independent sets in minimum colorings of split graphs", Discrete Applied Mathematics, Vol:247, pp:352-356, 2018.

Bron, C., Kerbosch, J.,"Algorithm 457: Finding all cliques of an undirected graph", ACM Communication, Vol:16, pp:575-577, 1973.

Connolly, S., Gabor, Z., Godbole, A., Kay, B., Kelly, T.,"Bounds on the Maximum Number of Minimum Dominating Sets", Discrete Mathematics, Vol:339, pp:1537-1542, 2016.

Deng, Y.-P., Sun, Y.-Q., Liu, Q., Wang, H.-C.,"Efficient Dominating Sets in Circular Graphs", Discrete Mathematics, Vol:340, pp:1503-1507, 2017.

Goddard, W., Henning, M.A., "Independent domination in Graphs: A Survey and Recent Results", Discrete Mathematics, Vol: 313, pp:839-854, 2013.

Golovach, P.A., Heggernes, P., Kante, M.M., Kratsch, D., Villanger, Y.,"Enumerating Minimal Dominating Sets in Chordal Bipartite Graphs", Discrete Applied Mathematics, Vol:199, pp:30-36, 2016.

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.
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.
Marti-Farre, J., Mora, M., Ruiz, J. L.,"Uniform Clutters and Dominating Sets of Graphs", Discrete Applied Mathematics, Vol:263, pp:220-233, 2019.
Naude, K.A.,"Refined pivot selection for maximal clique enumeration in graphs", Theoretical Computer Science, vol:613, pp:28-37, 2016.
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.
Rooij, J.van, Bodlaender, H.L., "Exact algorithms for dominating set", Discrete Applied Mathematics, Vol:159, pp:2147-2164, 2011.
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.
Thulasiraman, K.KT., Arumugan, S., Brandstadt, A., Nishizeki, T., "Handbook of Graph Theory, Combinatorial Optimization and Algorithms", CRC Press, Sendai, Japan, 2016.
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.
Yu,T., Liu, M.,"a linear time algorithm for maximal clique enumeration in large sparse graphs", Information Processing Letters, vol:125, pp:35-40, 2017.