Anatolian Journal of Computer Sciences

Volume:5 No:2 2020 © Anatolian Science

pp:62-70 ISSN:2548-1304

New Algorithms for Minimum Dominating Set in Any Graphs

Ali KARCI

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

Received Date : May 18, 2020.
Acceptance Date : Jun. 15, 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 is to propose some basic methods for solving problem of obtaining minimum dominating set which is one of these problems. In order to construct such fundamentals, a special spanning tree for given graph (except trees) was obtained by using proposed method, and it was called as Kmax tree. The fundamental cut-sets of given graph were obtained by using this spanning tree. The dominance of each node in the given graph were computed with respect to these fundamental cut-sets, node degrees in graph and corresponding Kmax tree for the first time in this paper. The dominance of each node illustrates whether that node will take place in minimum dominating set or not. For the sake of obtaining the minimum dominating set, there are three algorithms (proposed in this study for the first time) which are used in sequential order. Application of these algorithm concludes in obtaining the minimum dominating set which is an NP-hard and NP-complete problem. The proposed method in this study verified that this problem can be solved with these deterministic algorithms.

Keywords: *Dominating Sets, Fundamental Cut-Sets, Efficient Algorithms.*

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 Kn=(V,E) is a graph where $\forall vi,vj \in V$, $(vi,vj) \in E$, $i \neq j$. A maximum clique is a clique such that there is no clique with more nodes. A node vi is said to be neighbour of vj if $(vi,vj) \in E$. 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. In another word, the dominating set can be emphasized as follow:

G=(V,E) and $D \subseteq V$ is a dominating set if and-only if D satisfies any one of the following equivalent conditions (Thulasiraman et al, 2016).

a) N[D]=V.

b) For every vertex $v \in V-D$, $d(v,D) \geq 1$.

c) For every vertex $v \in V-D$, there exists a vertex $u \in D$ such that v is adjacent to u.

d) For every vertex $v \in V$, $|N[v] \cap D| \geq 1$.

e) For every vertex $v \in V\text{-}D$, $|N[v] \cap D| \geq 1$.

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. 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.

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 dominant 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.

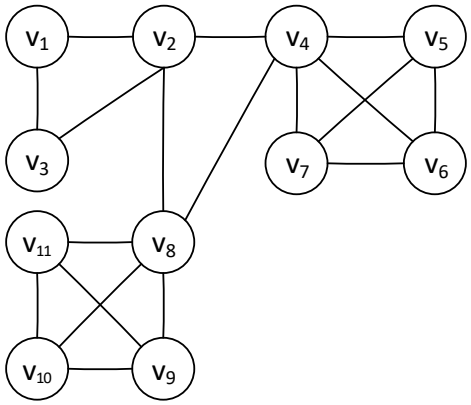

Figure 1. Any graph G=(V,E) consists of three cliques.

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 a special spanning tree of given graph and by using this spanning tree, the dominant 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. The depth-first search algorithm is searching from root to the leftmost leaf. If the

solution/goal is found, then the algorithm is terminated, otherwise, the process goes to next leaf, 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 descending order. Any node with maximum degree (assume it is vi) is selected as root node for spanning tree T of given graph G. The node in N(vi) are added to spanning tree T as children of vi. The children of vi are expanded from maximum degree to minimum degree. The obtained tree is called as **Kmax Tree** (Karcı Maximum Spanning Tree). All expanded children are added to children queue. The construction of Kmax is a mixture of breadth-first search and depth-first search.

Algorithm 1: Construction of Kmax Tree

```
Kmax_Tree(A,AT,D)
1. Q←∅
2. r←max(D) // D is degree matrix
3. T=(V,E1) and E1←∅
4. i←1,…, |V|
5.     j←1,…,|V|
6.           AT(i,j)←0
7.  EnQueue(Q,r)
8.  Q2←∅
9. for i←1,…,|N(r)|
10.     EnQueue(Q2,vi),   vi∈N(r)
11.     if AT(r,s)=0 then
12.        AT(r,s)=1, AT(s,r)=1
13.        EnQueue(Q2,s)
14. while Q2≠∅
15.      v←DeQueue_Max(Q2,A,AT)
16.      EnQueue(Q,v)
17.    for i←1,…,|N(v)|
18.        if not(vi∈Q2) and vi∈N(v)
19.            EnQueue(Q2,vi)
20.            AT(v,vi)=1, AT(vi,v)=1
21.     Remove_Zero(Q2,A,AT)
```

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.

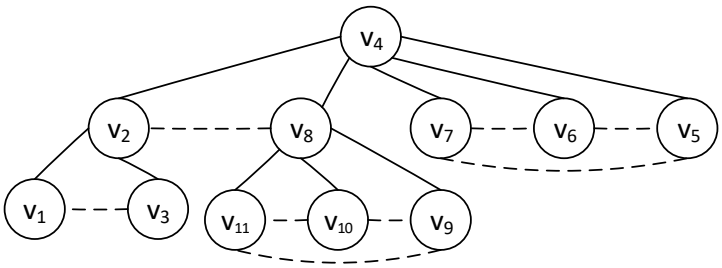

Figure 2. Kmax Tree of graph in Fig.1 (dashed edges are not part of tree).

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},v_{11}} and E={(v_1,v_2), (v_1,v_3), (v_2,v_3), (v_2,v_4), (v_2,v_8), (v_4,v_8), (v_4,v_5), (v_4,v_6), (v_4,v_7), (v_5,v_6), (v_5,v_7), (v_6,v_7), (v_8,v_9), (v_8,v_{10}), (v_8,v_{11}), (v_9,v_{10}), (v_9,v_{11}), (v_{10},v_{11})}. The construction of Kmax Tree can be explained in the following steps:

Step 1: The nodes of maximum degrees are v_4 and v_8. v_4 is added to Kmax Tree.

Step 2: Expand v4 and add nodes in N(v4) to Kmax Tree.

Step 3: The nodes v_2, v_8, v_7, v_6, v_5 are added to Kmax Tree, and they constitute second level. The node of maximum degree in N(v_4)={ v_2, v_8, v_7, v_6, v_5} (in the second level) is v_8. For the third level, v_8 is expanded firstly, then v_2 is expanded. Then there is no remaining node, and construction of Kmax Tree is concluded (obtained tree is seen in Fig.2).

3. AN EFFICIENT ALGORITHM FOR OBTAINING DOMINANT NODES

Assume G=(V,E) is a graph, and connected, Ec⊆E is a set of edges whose removal from graph G concludes in G is not connected, then Ec 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 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 dominant nodes in graph G, Kmax Tree are used. The obtained tree is 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 dominance 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.

Algorithm 2: Construction of Fundamental Cut-Sets

```
1. Assume e=(u,v)
2. Q2←∅
3. EnQueue(Q2,u)
4. while Q2≠∅
5.     u←DeQueue(Q2)
6.     EnQueue(Q,u)
7.     for i←1,…,n      //number of nodes in G=(V,E)
8.         if AT(u,i)=1 and i≠v
9.             EnQueue(Q2,i)
10.  while Q≠∅
11.      u←DeQueue(Q)
12.      for i←1,…,n // n is the number of nodes in G=(V,E)
13.          if A(u,i)=1 and i∉Q
14.              C(u,k)=B(u,k) // k illustrates the edge e=(u,i)
```

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.3 illustrates the results of algorithm for Kmax Tree.

Leaf Fundamental Cut-Sets: C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8.

C_1={(v_1,v_2), (v_1,v_3)} and C_2={(v_1,v_3), (v_2,v_3)}

C_3={(v_8,v_{11}), (v_9,v_{11}), (v_{10},v_{11})} and C_4={(v_8,v_{10}), (v_9,v_{10}), (v_{10},v_{11})}

C_5={(v_8,v_9), (v_9,v_{10}), (v_9,v_{11})} and C_6={(v_4,v_7), (v_5,v_7), (v_6,v_7)}

C_7={(v_4,v_6), (v_5,v_6), (v_6,v_7)} and C_8={(v_4,v_5), (v_5,v_6), (v_5,v_7)}

Internal Fundamental Cut-Sets: C_9, C_{10}.

C_9={(v_2,v_4), (v_2,v_8)}

C_{10}={(v_2,v_8), (v_4,v_8)}

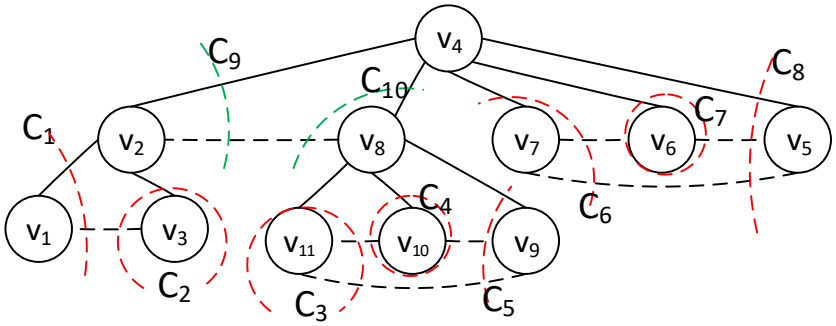

Figure 3. Fundamental cut-sets by using Kmax Tree (red cut-sets are leaf fundamental cut-sets; green cut-sets are internal fundamental cut-sets).

The appearance of each edge in the fundamental cut-sets constitutes the **Cut-Set Dominance value** of corresponding node. Each node has a node degree in graph, and this is called **Graph Dominance** of corresponding node. Each node has a node degree in spanning tree, and this called the **Spanning Dominance** of corresponding node. The dominant value of a node can be obtained by summation of these Graph Dominance and Spanning Dominance (the value obtained from fundamental cut-sets). In order to obtain dominant value of a node, Kmax Tree is used as spanning tree.

Before giving an algorithm to compute dominances 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 Bij=1 is the node vi and edge ej are incident and 0 otherwise. The cut-set Cnxm matrix is a submatrix of B such that Cij=1 is the cut-set Ci contains edge ej, 0 otherwise. After application of Algorithm 3 to any graph G with respect to Kmax Tree, π vector contains the dominant values of nodes. The maximum value in π corresponds to the most dominant node. π will be used for computing dominating set. All these algorithms are polynomial algorithms, so, there are efficient algorithms for determining dominant set for given graph.

In order to obtain minimum dominating set for graph in Fig.1, Algorithm 3 can be used repeatedly. The incidence matrix for the corresponding graph is seen in Fig.4. There are 11 nodes so, there will be 10 fundamental cut-sets. The Kmax tree for the corresponding graph is seen in Fig.3, and the Cut-Sets Matrix for this spanning tree is also seen in Fig.5.

Algorithm 3: Computing Dominance

1) Assume that G=(V,E) and T=(V,E_1) is a Kmax Tree Tree of graph G.
2) B is the incidence matrix, and C_{max} corresponds to Kmax Tree.
3) $E_{max} = B * C_{max}^T$ where E_{max} corresponds to Dominance 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) $\pi(v_i) = \pi(v_i) + E_{max}(i,j)$
8) i←1,…,n
9) $\pi(v_i) = \pi(v_i) + d_G(v_i) + d_T(v_i)$ 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.

i.e. $\pi(v_i)$=Spanning Dominance+Graph Dominance.

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17	e18
v_1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
v2	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
v3	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
v4	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0
v5	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
v6	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
v7	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
v8	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	1	0
v9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
v10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
v10	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

Figure 4. Incidence matrix for graph in Fig.1.

	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17	e18
C_1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
C3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
C4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
C5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
C6	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
C7	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
C8	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
C9	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
C10	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Figure 5. Cut-Sets matrix for graph in Fig.1.

In order to obtain the minimum dominating set for graph seen in Fig.1, the following steps of application of Algorithm 4 can be followed.

Step 1: Applying Algorithm 3 once, and then the vector π=[5 5 5 11 9 9 9 11 9 9 9] is obtained. There are two the largest values correspond to nodes v_4 and v_8. Assume D is dominating set and D={v_8}, and then remove all edges incident to v_8 from Kmax tree.

Step 2: $D \cup N(v_8)=\{v_2,v_4,v_8,v_9,v_{10},v_{11}\} \neq V=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11}\}$. So, Algorithm 3 will be applied to Partial Kmax tree seen in Fig.5 again.

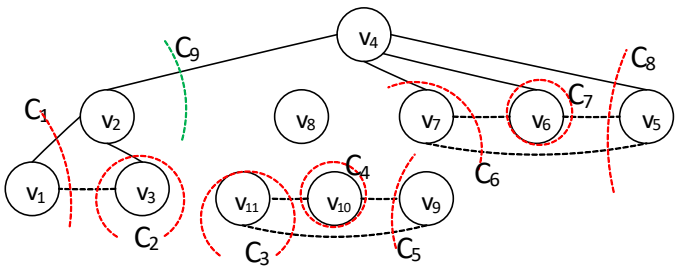

Figure 6. Partial Kmax Tree

The vector π=[7 9 6 14 9 9 9 0 8 8 8] is obtained, and the largest value 14 corresponds to node v_4, and $D=D\cup\{v_4\}=\{v_4,v_8\}$ and $D\cup N(D)=\{v_2, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\} \neq V=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$. The all edges incident to node v4 are removed from Kmax tree.

Step 3: Algorithm 3 is applied to partial Kmax tree seen in Fig.7., and the obtained vector is π=[6 9 6 0 8 8 8 0 8 8 8]. The largest value in vector corresponds to v_2, and v_2 is added to dominating set. $D=D\cup\{v_2\}=\{v_2,v_4,v_8\}$ and $D\cup N(D)=\{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11}\}=V$, and process is terminated and D is the answer.

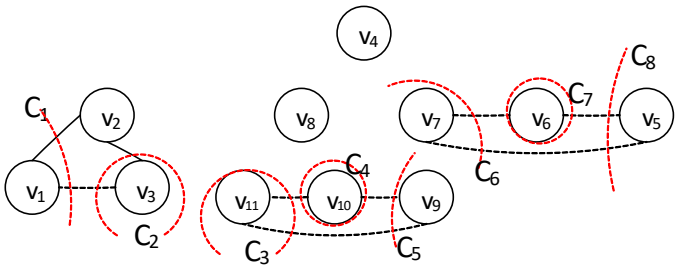

Figure 7. Partial Kmax Tree.

Theorem 1: *The complexity of Algorithm 1 is polynomial with respect to the numbers of nodes and edges in the given graph.*

Proof: Algorithm started with the node of largest degree, and this step requires the computation of node degrees and sorting of these values. Assume that each node degree is represented as di, 1≤i≤n, and the summation of degrees is equal to twice the number of edges.

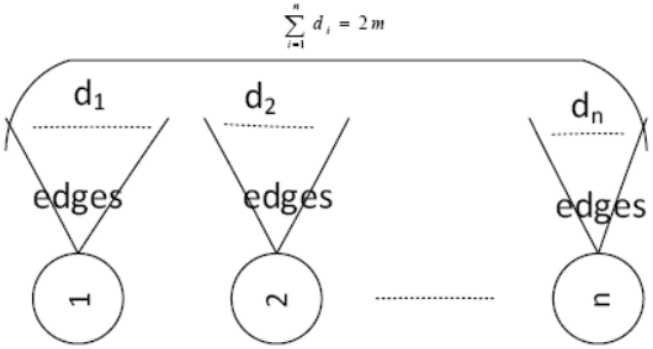

Figure 8. Summation of degrees.

The complexity of summation of degrees is O(m) and there are n node degrees, and the sorting of these degrees requires the complexity as O(nlgn). Thus, the complexity of construction of Kmax tree is O(nlgn+m) ■

Theorem 2: *The loose upper bound for complexity of Algorithm 2 is polynomial with respect to the numbers of nodes and edges in the given graph and it is O(n^2+m).*

Proof: The Algorithm 2 constructs fundamental cut-sets for corresponding Kmax tree. There are two types of cut-sets: Leaf fundamental cut-sets and internal fundamental cut-sets. The construction of leaf fundamental cut-sets requires at most m (number of edges) steps, since the summation of degrees of leaf nodes is at most equal to m. The construction of internal fundamental cut-sets requires at most n(n-1) steps, since the number of branches in Kmax is equal to n-1. Construction of an internal fundamental cut-set start at a branch traverse one part of Kmax tree, and the loose upper bound for this process is at most n(n-1). This case is seen in Fig.9. so, the complexity of this algorithm is O(n^2+m) ■

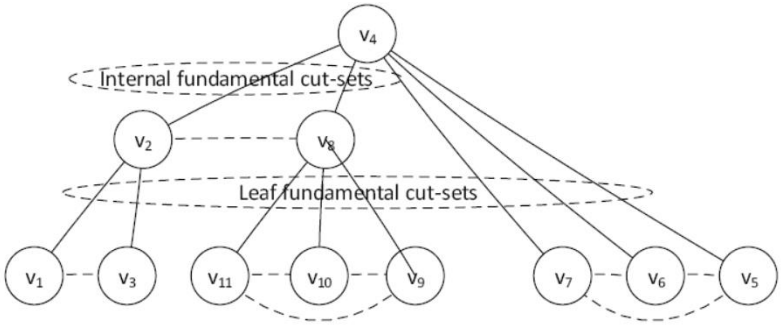

Figure 9. Kmax tree, leaf and internal fundamental cut-sets.

Theorem 3: *The loose upper bound for complexity of Algorithm 3 is $O(n^2+nm)$ where n is the number of nodes and m is the number of edges.*

Proof: The algorithm 3 includes three important steps such as 3, 4 and 8. Step 3 requires the complexity O(nm), step 4 requires the complexity $O(n^2)$, and finally, step 8 requires the complexity O(n). So, the complexity of Algorithm 3 is $O(nm+n^2+n)=O(nm+n^2)$ ■

Algorithm 4: Computing Minimum Dominating Set
1) Assume that G=(V,E) and T=(V,E_1) is a Kmax Tree of graph G.
2) D is minimum dominating set, D=∅.
3) N(D) is a set of neighbours of D and N(D)= ∅.
4) Obtain Kmax by executing Algorithm 1.
5) Obtain effectiveness of nodes by executing Algorithm 2.
6) While D∪N(D)≠V (D∪N(D)⊂V)
7) π←sort(π) in descending order.
8) D=D∪{$v_{\pi(1)}$}
9) N=N∪{neighbours of $v_{\pi(1)}$}
10) Remove all edges incident on $v_{\pi(1)}$ from Kmax and Cut-Sets
11) π←executing Algorithm 3 (Incident and Cut-sets matrices are updated with respect to removed edges)
12) The set D is result and contains minimum dominating set for given graph.

The minimum dominating set for given graph is computed by using Algorithm 4. The first step is to construct the Kmax tree for the given graph, and then the fundamental cut-sets are obtained. While obtaining cu-sets, the incidence matrix and cut-sets matrix are obtained. These both matrices are used to compute the dominant value of each node, and nodes are sorted in descending order with respect to node effect. The first node is selected as element of minimum dominating set, and all edges incident on this node are removed from Kmax tree and cut-sets matrix is updated with respect to this new situation. If the selected node and its neighbours are cover all nodes, then process is terminated, otherwise, effects of nodes are re-computed. The next the most effectiveness is selected as element of dominating set, all edges incident on it are removed from Kmax tree. If graph node set is not cover, then this process is repeated one more time.

Theorem 4*: The loose upper bound for complexity of Algorithm 3 is $O(n^2+nm)$ where n is the number of nodes and m is the number of edges.*

Proof: the proof of this theorem is left to reader, since it is trivial ■

Theorem 5*: The method proposed in this study obtains the minimum dominating set for a given graph.*

Proof: The analytically proving this theorem is an open question. Intuitively it is an accurate claim, but it needs proof in details. We will give the proof this theorem in our next study (studies) ■

4.CONCLUSIONS

This study includes very important algorithms for solving NP-hard and NP-complete problems such as minimum dominating. The proposed method in this paper includes construction of a special spanning tree (Kmax tree) for the given graph. Then the fundamental cut-sets are computed by using this tree. Kmax tree and cut-sets matrix are used for computation of minimum dominating set repeatedly. All algorithms used in the proposed method are polynomial algorithms (deterministic algorithms), this verifies that the proposed method is an efficient method where there are no efficient methods in literature until today.

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.