Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 31 No. 7s (2024)

On Decomposing Any Graph G into VDT Graphs

M. Sivasankari[1], M. Yamuna[2]

[1]Department of Mathematics, SAS, VIT, Vellore, Tamil Nadu, India. Email: sivasankari.2019@vitstudent.ac.in

[2]Department of Mathematics, SAS, VIT, Vellore, Tamil Nadu, India. Email: myamuna@vit.ac.in

Corresponding author: myamuna@vit.ac.in

Article History:

Received: *04-06-2024*

Revised: *03-07-2024*

Accepted: *30-07-2024*

Abstract:

This article targets to develop an iterative procedure for decomposing any random graph G into subgraphs $H_1, H_2, ..., H_k$ such that each H_i is a VDT graph, $1 \leq i \leq k$. This iterative technique is based on developing a rooted tree, and then decompose G into VDT graphs using backtracking.

Keywords: Decomposition, BFS, Tree, Domination.

1. Introduction

In the literature of graph theory, graph decomposition is one interesting problem where any graph is decomposed into subgraphs [1]. Researchers have made different contributions to this domain. Samuel Issacraj et al. studied forks, trees formed by subdividing a star of size three once, providing conditions for fork-decomposition of some total graphs [2]. Ganguly et al. demonstrated the decomposition of the real line R into complementary sets with empty ratio sets [3]. Hasan et al. explored properties of stiff modules in QT AG modules [4]. Bhatt et al. computed a rank-1 tensor approximation [5]. Gao et al. applied graph theory to crystal structure prediction [6]. Guo et al. characterized hereditary vertex decomposable graphs, including well-known examples [7]. Bonizzoni et al. developed efficient algorithms for hypergraph decomposition [8]. Grohe et al. contributed to graph minor decomposition [9]. Mesady et al. [10] studied edge decomposition on circulant graphs while Dac et al. [11] explored a conjecture on digraph cycles. Most researchers attempt to decompose specific kind of graphs. The questions that arise is "Is it possible to decompose any given graph G? Can we decompose a random graph G into predefined type of graphs". In this article we develop an iterative technique to decompose any random graph into a specific type of graphs called VDT graphs.

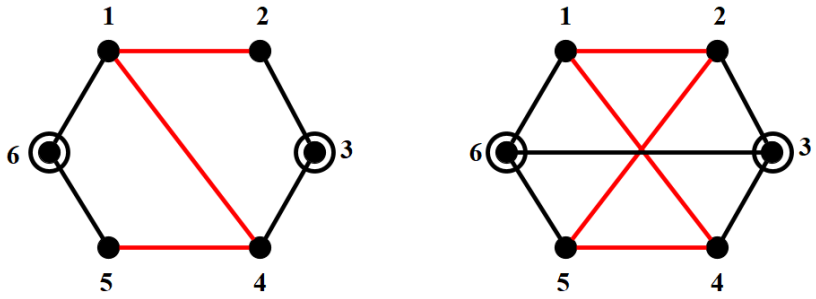
2. Preliminary Note

For any set S of vertices in G, the induced subgraph ⟨S⟩ is the maximal subgraph of G with vertex set S. A tree is a connected graph without cycles. Forks are trees formed by subdividing a star of size three exactly once. A path is a sequence of connected vertices. A comb is a graph shaped like a comb with a central path and hanging edges. Graph decomposition involves dividing a graph into smaller graphs such that every edge appears in exactly one of the smaller graphs. A dominating set D of G is a set of vertices of G such that every vertex in V – D is adjacent to a vertex in D. If D has the smallest possible cardinality of any dominating set of G, then D is called a minimum dominating set. The cardinality of any minimum dominating set for G is called a domination number of G and is denoted by γ(G). A γ − set denotes a dominating set for G with minimum cardinality [12]. In this

article we have developed an iterative procedure for decomposing any random graph G into subgraphs $H_1, H_2, ..., H_k$ such that each H_i is a VDT graphs, $1 \leq i \leq k$.

3. VDT Graph

A graph G is said to be VDT graph if G has at least one γ–set D such that ⟨V –D⟩ is a tree. We shall denote such graphs as VDT graphs and the γ – set D such that ⟨V – D⟩ is a tree as a $\gamma_{VDT}-$set.The graphs in Figure 1 is an example of VDT and non - VDT graph.



VDT graph Non – VDT graph

Fig 1. VDT and non-VDT graphs

Now let us prove few observations to decompose G into VDT graphs. In all these observations, let G be a VDT graph and D a $\gamma_{VDT}-$set for G.

Theorem 1 D contains all the pendant vertices.

Proof If possible, let v ∈ D such that v is a pendant vertex. Let u be a support vertex with respect to D. We know that both u, v cannot be contained in D since D′ = D − v itself is a γ - set for G, a contradiction to our assumption that D is a $\gamma_{VDT}-$set. So, if v ∈ D, u ∉ D, implies ⟨V − D⟩ is disconnected, a contradiction to our assumption that D is a $\gamma_{VDT}-$set. Hence D contains all the pendant vertices of G.

Theorem 2 For any support vertex u ∈ G, |N(u) ∩ X | = 1, where X is the set of all pendant vertices of G.

Proof If possible, let u ∈ G such that |N(u) ∩ X | ≥ 2. Without loss of generality let |N(u) ∩ X | = k, k ≥ 2. Let $X_1 \subseteq X = \{v_1, v_2, ..., v_k\}$ be the set of all pendant vertices adjacent to u. Then ⟨V − D⟩ is a disconnected graph with at least k + 1 components, a contradiction to our assumption that D is a $\gamma_{VDT}-$set with respect to G, which implies |N(u) ∩ X | = 1.

Theorem 3 The corona graph $G \circ K_1$ is a VDT graph if G is a tree.

Proof Let G be a tree. Consider $T \circ K_1$. Let $T \circ K_1 = Y_1 \cup Y_2 = \{u_1, u_2, ..., u_k\} \cup \{v_1, v_2, ..., v_k\}$ where $\langle Y_1 \rangle$ is a tree and $\langle Y_2 \rangle$ is a null graph. From Theorem 1, we know that D contains all the pendant vertices, implies $Y_2 \in D$. Also, by Theorem 2, we know that every u_i, $1 \leq i \leq k$, is a support vertex. By Theorem 2, we know that Y_1 does not belong to D. Also, $\langle Y_1 \rangle$ is a tree, implies $G \circ K_1$ is a VDT graph if G is tree.

In general $G \circ K_1$ need not be a VDT graph for any graph G. The graph in Figure 2 is an example of corona graph that is not VDT. From these observations we know that there exist a family of graphs $T \circ K_1$ that are VDT for any tree T. For our purpose of decomposition, we shall choose the VDT trees P_2, P_4, comb graphs. Now we continue further to determine a decomposition of any given graph into P_2, P_4, comb graphs.

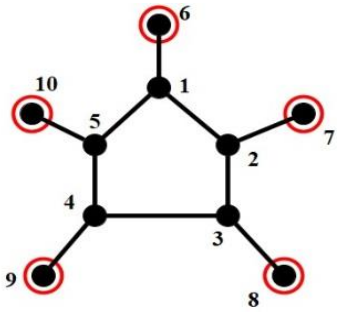


Fig 2. Non VDT Corona Graph

4. VDT Graph Decomposition

Let G be any graph with n vertices. For our iteration procedure let us consider the graph G in Figure 3 as illustration.

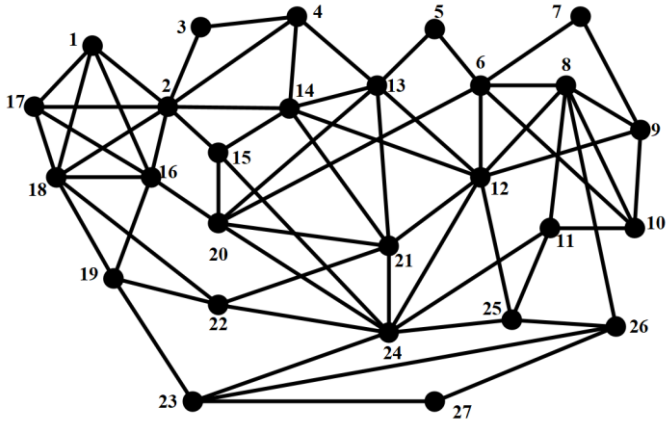


Fig 3. Graph G for VDT graph decomposition

Step 1 Determine the rooted tree T_1 for G from any arbitrary vertex using BFS algorithm. For our example the rooted tree is seen in Figure 4.

Step 2 Vertex Labelling of Tree T_1

Let us label the tree T_1 obtained by BFS algorithm.

1. Label the root vertex as v_{11}.
2. Label the vertices in level 2 as $v_{21}, v_{22}, ..., v_{2r_2}$ from left to right.
3. Label the vertices in level 3 as $v_{31}, v_{32}, ..., v_{3r_3}$ from left to right.
4. Continue this pattern and label the vertices at level k as $v_{k1}, v_{k2}, ..., v_{kr_k}$ from left to right.

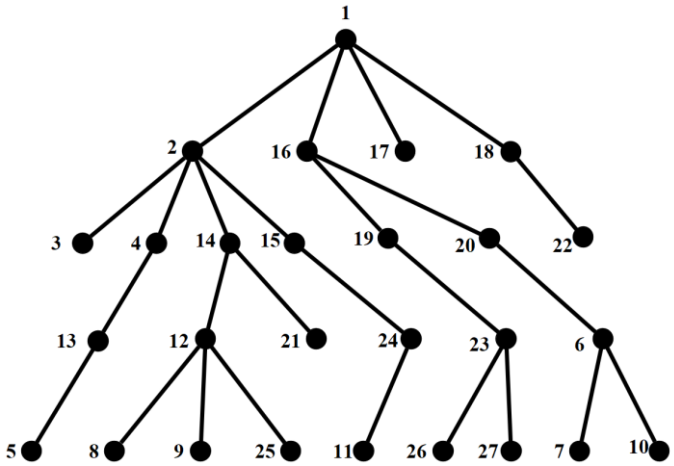


Fig 4. Tree T_1 using BFS algorithm

Step 3 Include the remaining edges to the tree T_1 as seen in Figure 5.

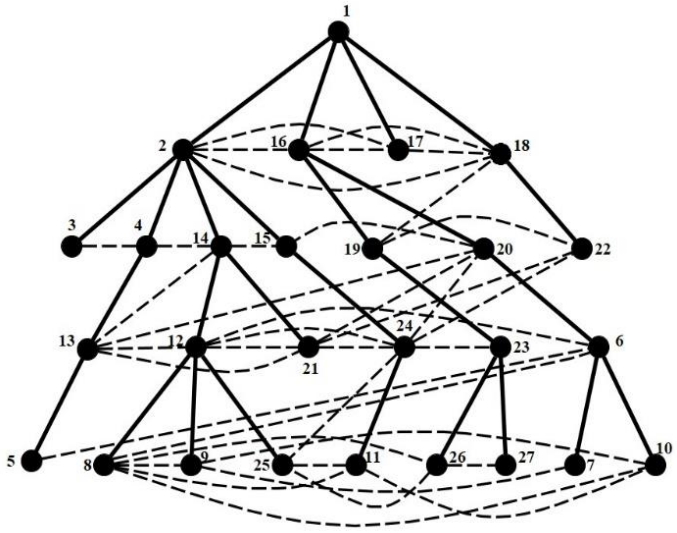


Fig 5. T_1 with remaining edges added

Step 4 Edge Labelling

Any edge $e = (v_{ij}, v_{ik})$ is labelled as $e_{ij\ ik}$. Note that here i represents the level of the vertices. $e_{11\ 32}$ denotes an edge between vertices v_{11} and v_{32}, the edge is between vertices in level 1 and level 3.

The partial structure of the tree T_1 labelling is as seen in Figure 6.

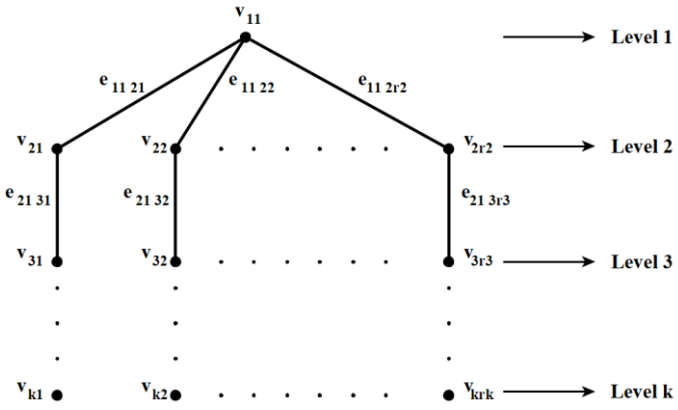


Fig 6. Labelling of T_1

Step 5 Starting from vertex v_{k1} trace a longest possible path by back tracking along the tree T_1 by choosing a new edge each time. Label the resulting path as $P_{k1} : v_{k1}$, PRED(v_{k1}), PRED(PRED(v_{k1})), ..., v_{11}. Let us now attempt to include edges to the path P_{k1} to generate a comb graph if possible.

Step 6 Let v_{ij} be any random vertex in path P_{k1}.

a. 1. Include a new edge $e_{ij\,ik}$ to P_{k1} where k is the smallest possible value, $2 \le k \le v_{i\,r_i}$ (if it exists) such that v_{ij} is adjacent to v_{ik}. Include v_{ik} and $e_{ij\,ik}$ to P_{k1}.

Else

2. Choose the smallest possible p, $2 \le p \le v_{i\,r_{i+1}}$ (if it exists) such that v_{ij} is adjacent to $v_{i+1\,p}$. Include $v_{i+1\,p}$ and $e_{ij\,i+1\,p}$ to P_{k1}.

Else

3. Choose the smallest possible q, $2 \le q \le v_{i\,r_{i-1}}$ (if it exists) such that vij is adjacent to $v_{i-1\,q}$. Include $v_{i-1\,q}$ and $e_{ij\,i-1\,q}$ to P_{k1}.

Repeat 1, 2 or 3 for every vertex in P_{k1} starting from v_{k1}. If there is at least one vertex say v_{ij} for which step 6a fails, we terminate step 6a. Else we continue step 6 until we reach vertex v_{11}. If step 6a is true for every vertex in P_{k1}, then we include w vertices and w edges to P_{k1} generated by step 6a where |w| = |V (P_{k1})|, $1 \le w \le k$. The resulting graph is a comb say $(P_{k1} \circ K_1)$. Else

b. Retain the path P_{k1}.

Step 7 Starting from vertex v_{k2} trace a longest path by back tracking along the tree T_1 by choosing a new edge each time. Label the resulting path as $P_{k2} : v_{k2}$, PRED(v_{k2}), PRED(PRED(v_{k2})), ..., v_{ij}. Repeat step 6a for path P_{k2} to generate a comb $(P_{k2} \circ K_1)$ if it exists or retain P_{k2} (Step 6b).

Step 8 Repeat step 5 for every v_{ki}, $3 \le i \le r_k$ to generate a sequence $X_1, X_2, ..., X_r$ where $X_i = \begin{cases} P_{ki} \circ K_1 & \text{or} \\ P_{ki} \end{cases}$

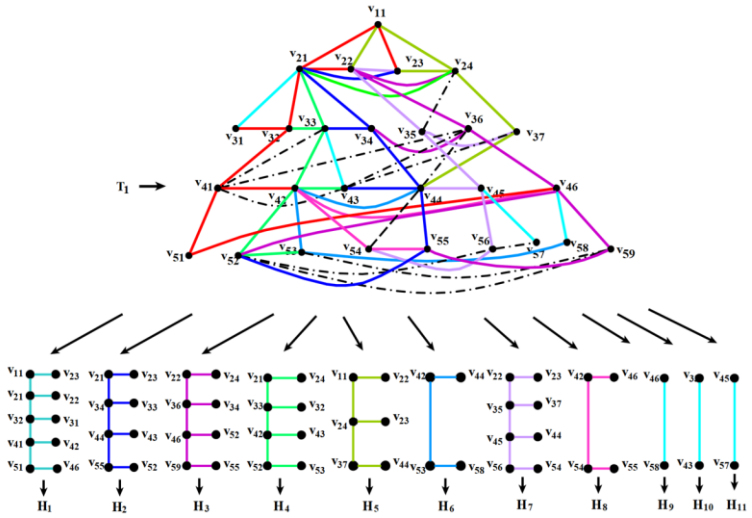


Fig 7. Iteration 1

Step 9 Repeat Step 5 to Step 8 for all the vertices in the level k – 1, k – 2, . . ., 2. Executing step 5 to step 8 once, we finally generate a sequence of subgraphs $\{H_1, H_2, ..., H_s\}$ where each $H_i = \begin{cases} P_{ki} \circ K_1 & \text{or} \\ P_{ki} \end{cases}$. For our example iteration 1 is given in Figure 7.

If all the edges of G are covered then terminate the procedure here.

Else

Step 10 Let $L = G - \{H_1 \cup H_2 \cup ... \cup H_s\}$. If L is a connected graph, then repeat step 1 to step 9. Else if L is disconnected graph, then repeat step 1 to step 9 on all the X components say $G_1, G_2, ..., G_x$. For our example the graph G is disconnected as seen in Figure 8.

Summarizing we repeat BFS algorithm multiple times to generate a sequence of trees $T_1, T_2, ..., T_z$ as seen in Figure 9. From these trees we then decompose graph G into subgraphs $H_1, H_2, ..., H_s$ where each $H_i = \begin{cases} P_{ki} \circ K_1 & \text{or} \\ P_{ki} \end{cases}$.

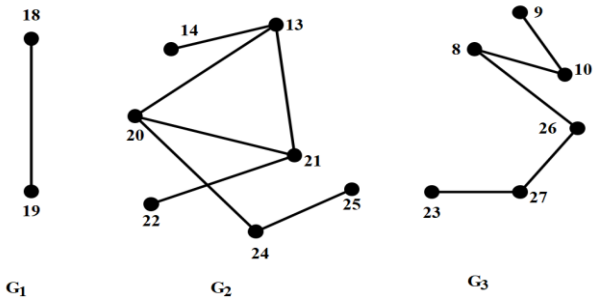


Fig 8. Subgraphs of graph G for VDT graph decomposition

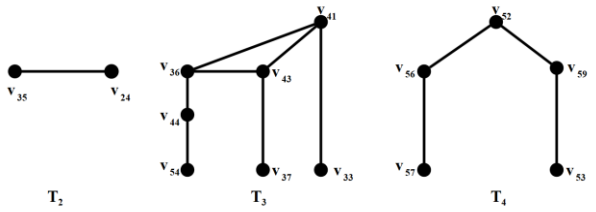


Fig 9. Rooted trees T_2, T_3, T_4 using Step 1

Figure 10 shows the decomposition for our example in iteration 2.

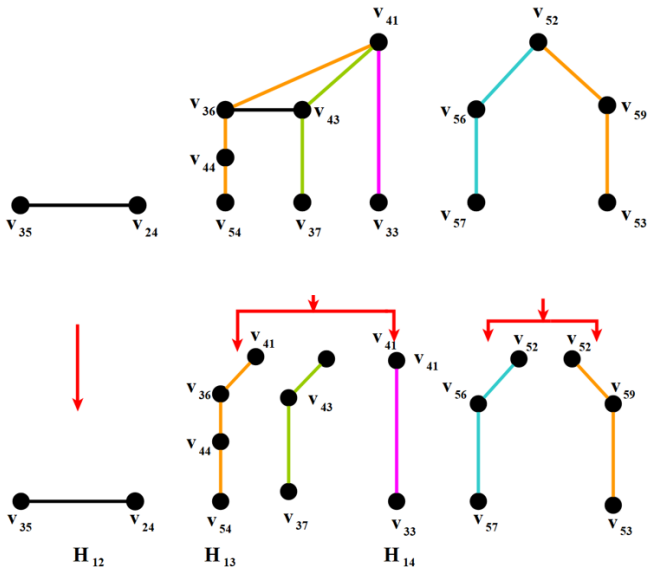


Fig 11. Iteration 2

Step 11 If step 10 covers all the edges of G, then terminate the iterative procedure. Else repeat step 1 to step 9 on all the resulting graphs until all the edges of G are covered.

Our example terminates in iteration 3 as seen in Figure 11 and

$H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8, H_9, H_{10}, H_{11}, H_{12}, H_{13}, H_{14}, H_{15}, H_{16}, H_{17}, H_{18}, H_{19}, H_{20}$

are the decompositions of graph G. Hence, we conclude the following theorem.

Theorem 4 Given any graph G, there exist a decomposition $H_1, H_2, \ldots, H_s$ such that each H_i , $1 \leq i \leq s$ is either comb graph or path.

Theorem 5 Given any graph G there exist a decomposition $\{H_1, H_2, ..., H_k\}$ for G such that each H_i is a VDT graph.

Proof Let G be any given graph. Determine a decomposition $X = H_1, H_2, \ldots, H_s$ for G using Theorem 4. We know that each H_i is either a comb or a path. If H_i is a comb then it is a VDT graph. If H_i is a path P_n then we continue as follows.

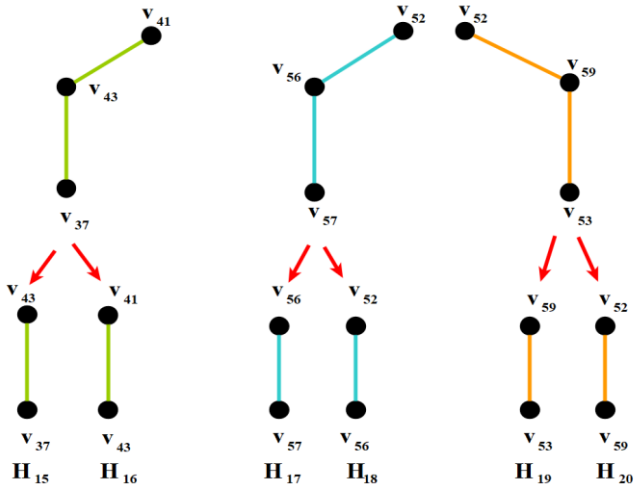


Fig 12. Iteration 3

i. Let $P_n = v_1\, e_1\, v_2\, e_2\, v_3 \ldots v_{n-1}\, e_{n-1}\, v_n$ where $e_{n-1} \equiv 0 \bmod 3$. Let us decompose P_n into paths $P_1, P_2, \ldots, P_{k1}$ where $P_1 = v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_4$; $P_2 = v_4\, e_4\, v_5\, e_5\, v_6\, e_6\, v_7; \ldots$ $P_{k1} = v_{n-3}\, e_{n-3}\, v_{n-2}\, e_{n-2}\, v_{n-1}\, e_{n-1}\, v_n$.

ii. If $e_{n-1} \equiv 1 \bmod 3$ then let us decompose P_n into paths $P_1, P_2, \ldots, P_{k2}$ where $P_1 = v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_4$; $P_2 = v_4\, e_4\, v_5\, e_5\, v_6\, e_6\, v_7$; .... $P_{k2} = v_{n-1}\, e_{n-1}\, v_n$.

iii. If $e_{n-1} \equiv 2 \bmod 3$ let us decompose P_n into paths $P_1, P_2, \ldots, P_{k3}$ where $P_1 = v_1\, e_1\, v_2\, e_2\, v_3\, e_3\, v_4$; $P_2 = v_4\, e_4\, v_5\, e_5\, v_6\, e_6\, v_7$; .... $P_{k3-2} = v_{n-5}\, e_{n-5}\, v_{n-4}\, e_{n-4}\, v_{n-3}\, e_{n-3}\, v_{n-2}$; $P_{k3-1} = v_{n-2}\, e_{n-2}\, v_{n-1}$; $P_{k3} = v_{n-1}\, e_{n-1}\, v_n$.

Hence from the decomposition X we obtain a new decomposition $Y = \{H_1, H_2, \ldots, H_k\}$, $k \geq s$ such that each H_i is either P_2, P_4, $P_n \circ K_1$, implies G can be decomposed into VDT graphs.

5. Conclusion

Graph decomposition has several significant applications. It reveals patterns and connectivity between vertices within graphs by revealing hierarchical relationships and substructures. Different methods and algorithms exist for graph decomposition. According to the graph, each approach offers unique insights and advantages. In this article, we presented an algorithm for determining a decomposition for any given graph into comb graphs, P_4 and P_2 which are VDT graphs. This algorithm can be used to study graph decomposition in a variety of contexts. It can help identify and understand the underlying structures of complex networks. Furthermore, it can be used to solve problems related to graph decomposition in an efficient manner.

References

[1] Narsingh Deo, Graph theory with Applications to Engineering and Computer Sciences, Prentice Hall India, 2016.
[2] Samuel Issacraj, A and Paulraj Joseph.J, Fork decomposition of some total graphs, Palestine journal of mathematics, Vol. 12(Special Issue II) (2023), 65–72.
[3] Ganguly.D. K. and Dhananjoy Halder, On decomposition of the real line in terms of ratio sets, Palestine journal of mathematics, Vol. 7(2) (2018), 624–627.
[4] Ayazul Hasan, Jules Clement Mba and Rafiquddin, Recent decomposition results on QT AG-modules, Palestine journal of mathematics, Vol. 7(2) (2018), 633–640.
[5] Vineet Bhatt and Kumar. S., A cas aided survey of CP decomposition and rank-1 approxition of a 3rd-order tensor, Vol. 3(2) (2014), 299–319.
[6] Hao Gao, Junjie Wang, Yu Han and Jian Sun, Enhancing crystal structure prediction by decomposition and evolution schemes based on graph theory, Fundamental Research, Vol 1(4) (2021), 466-471.
[7] Jin Guo, Meiyan Li and Tongsuo Wu, A new view toward vertex decomposable graphs, Discrete Mathematics, Vol 345(9) (2022), 112953.
[8] Paola Bonizzoni and Gianluca Della Vedova, An algorithm for the modular decomposition of hypergraphs, Journal of Algorithms, Vol 32(2) (1999), 65-86.
[9] Martin Grohe, Ken-ichi Kawarabayashi, and Bruce Reed, A simple algorithm for the graph minor decomposition logic meets structural graph theory, Proceedings of the 2013 Annual ACM SIAM Symposium on Discrete Algorithms (SODA), (2013), 414- 431.
[10] A. El-Mesady, Y.S. Hamed, H. Shabana, On the decomposition of circulant graphs using algorithmic approaches, Alexandria Engineering Journal, Vol 61(10) (2022) 8263-8275.
[11] Ngo Dac Tan, A decomposition for digraphs with minimum outdegree 3 having no vertex disjoint cycles of different lengths, Discussiones Mathematicae Graph Theory, Vol 43 (2023), 573 – 581.
[12] T. W. Haynes, S. T. Hedetniemi, P. J. Slater, Fundamentals of Domination in Graphs, Marcel Dekker, Inc., New York, 1998.