Generation of Specialized Graph Classes Using Hyper-edge Replacement P System

Vinodhini Krishnamoorthy, Meena Parvathy Sankar

***Abstract*—Graphs serve as flexible representations for capturing relationships between entities, while graph grammar offers a rule-based framework for transforming and generating graphs. In recent years, hyper-edge replacement graph grammar has emerged as a significant tool for generating both graphs and hypergraphs. George Paun introduced membrane computing, also known as the P system, as a computational paradigm that draws inspiration from biological systems. The intricate processes observed at the cellular level initially inspired the development of membrane computing. In this study, we leverage the ability of the hyper-edge replacement graph rewriting P system to generate new graphs using minimal-order hyper-edge rules, resulting in a large collection of graphs. Our primary objective is to produce cycle, tree, wheel, and broom graphs, as well as graphs that demonstrate resemblances to cycle, tree, and wheel graphs. This entails the generation of a multitude of graph types, including pan, prism, sunlet, tadpole, web, binary, ternary, k-ary trees, gear, helm, flower, and sunflower.**

***Index Terms*—Hyper-edge, Hyper-edge replacement graph grammar (HRG), Graph P system, Hyper-edge replacement graph rewriting P system (HRGRPS).**

I. Introduction

THE concept of graph grammars represents a natural extension of formal grammars [10], originally devised for string manipulation, to the realm of graph structures. This extension furnishes a systematic framework for precisely modeling local transformations within graphs, thereby enabling the formalization of graph rewriting processes. While node replacement and edge replacement [2] constitute fundamental primitives for graph transformation, hyper-edge replacement [5] offers a more comprehensive and expressive paradigm. Central to this approach are hyper-edges, atomic entities characterized by ordered sets of inbound and outbound connections to nodes, mediated by source and target functions, respectively. The pioneering work of Feder and Pavlidis in the early 1970s introduced hyper-edge replacement graph grammar, a foundational technique for rewriting both graphs and hypergraphs, which has since been extensively explored by researchers. This grammar facilitates the substitution of hyper-edges bearing non-terminal labels with graphs containing both terminal and non-terminal hyper-edge labels, thereby generating a vast array of graph structures. We denote the collective ensemble of graphs producible by this grammar as the "hyper-edge replacement language" (HRL), while the set of all hyper-edge replacement grammars is referred to as HRG.

Manuscript received March 11, 2024; revised September 04, 2024.

Vinodhini Krishnamoorthy is a research scholar in the Department of Mathematics, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India. (email:vk8271@srmist.edu.in).

Meena Parvathy Sankar is an Assistant professor in the Department of Mathematics, College of Engineering and Technology, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India. (email:meenap@srmist.edu.in).

Membrane computing [7], introduced in 1998, is an innovative computational paradigm inspired by biological processes. Rooted in the study of cellular mechanisms, this approach seeks to emulate various cellular activities, including protein interactions, enzyme functions, membrane dynamics, evolutionary processes, and molecular transport across membranes. Professor George Paun of Romania spearheaded the paradigm, commonly known as the P system. The initial theoretical framework was established with the first publication on the subject appearing in 2000. Membrane computing operates within the domain of distributed parallel computing devices [6], where activities within membranes occur concurrently. Information transfer is facilitated through a hierarchical structure, with messages conveying relevant data as they move from the inner to the outer regions of each membrane. The core components of membrane computing include the membrane structure, multisets, and rules. Multisets, which initially consist of enzymes, proteins, or chromosomes, can evolve into various entities such as strings, arrays, or graphs as they move across membranes. These multisets undergo transformations over time, governed by a set of rules as they transition between membranes, enabling the modeling of complex biological processes.

The integration of hyper-edge replacement grammar and P systems, known as hyper-edge replacement graph rewriting P systems (HRGRPS) [8], facilitates computations in a distributed and parallel manner. Hyper-edges represent connections between different membranes, and rewriting rules are applied to hyper-edges to enable the transformation and evolution of the graph structure. This system facilitates the modeling of complex interactions and parallel processing inside a framework based on graphs. It is well-suited for applications in bioinformatics, network modeling, and distributed computing.

This paper proposes the generation of an exhaustive collection of cycle graphs and their associated variants, encompassing pan, tadpole, prism, sunlet, and web graphs. Additionally, it seeks to create a comprehensive array of tree graphs, including binary, ternary, and k-ary trees, as well as wheel graphs and their corresponding derivatives, such as helm, gear, flower, sunflower, and broom graphs. The generation of these graphs will be facilitated through the utilization of the conventional hyper-edge replacement graph rewriting P systems (HRGRPS) model, augmented by minimal order rules. This endeavor aims to provide a systematic and rigorous framework for the creation and analysis of complex graph structures.

II. Preliminaries

This section provides an overview of the fundamental concepts and notations employed in this paper, serving as

a foundation for further exploration. The graphs examined in this study are characterized by their simplicity and undirected nature, adhering to a standard representation denoted by the tuple (V, E), where V and E signifies the set of nodes and edges respectively. In this context, each edge establishes a connection between exactly two nodes, but hypergraphs extend this concept by introducing hyper-edges, which can connect an arbitrary number of nodes, thereby enabling the representation of complex relationships. This is particularly pertinent in contemporary real-world social networks, where hypergraphs play a crucial role. Consequently, hyper-edges and hypergraphs are integral components of hyper-edge replacement graph grammar, which emerges as the most powerful grammar among existing alternatives.

Definition 2.1: A hypergraph [4] H defined over a given, but arbitrary, set of labels C comprises of a tuple (V, E, s, t, l), where:

- V indicates the finite set of nodes.
- E indicates the finite set of hyper-edges.
- s is the source function, associating each hyper-edge with its source nodes denoted by $s(e)$.
- t is the target function, associating each hyper-edge with its target nodes denoted by $t(e)$.
- l denotes the labeling function, assigning each hyper-edge a label.

Here, $e \in E$ and the collection of all hypergraphs defined over the label set C is symbolized by $\mathbb{H}_C$.

Definition 2.2: A hypergraph H belonging to the set $\mathbb{H}_C$ is labeled as a handle [4] if it satisifies the conditions: $E_H = \{e\}$, $s_H(e) = begin_H$, and $t_H(e) = end_H$. In this context, $begin_H$ and end_H are elements of V^*, which denotes the set of all finite sequences of nodes from V. The pair of numbers (m, n) classifies the handle, where m and n signifies the count of source nodes and target nodes associated with e repestively.

Definition 2.3: A hyper-edge replacement grammar (HRG), as described in [4] comprises four components (N, T, P, S), where:

- $N \subseteq C$ represents the set of non-terminal hyper-edge labels.
- $T \subseteq C$ represents the set of terminal hyper-edge labels.
- P denotes the set of production rules, which is finite and consisting of ordered pairs (A, R), where $A \in N$ and $R \in \mathbb{H}_C$.
- $S \in \mathbb{H}_C$ denotes axiom or the start graph with (1,1) handle.

Definition 2.4: The attaching nodes [8] are defined by a mapping $att : E \to V^*$, which assigns a sequence of pairwise distinct attachment nodes $att(e)$ to each $e \in E$.

Definition 2.5: For $H \in \mathbb{H}_C$, the set of nodes occurring in the sequence $ext_H = begin_H.end_H$ is called the set of external nodes [4] of H and is denoted by EXT_H.

Definition 2.6: If $\forall$ (A, R) belongs to P, $|EXT_R| \leq r$, then a hyper-edge replacement grammar is said to be of order [4] r for some r belongs to $\mathbb{N}$.

Definition 2.7: The hypergraph language [4] $L(HRG)$ generated by HRG consist of all terminal labeled hypergraphs which can be derived from S by applying the production rules in P.

$$L(HRG) = \{H \in \mathbb{H}_T | S \Rightarrow_P^* H\}$$

In the early 1970s, researchers introduced hyper-edge replacement, a straightforward approach for rewriting hypergraphs and graphs. They further developed this concept of hyper-edge replacement graph grammar into the hyper-edge replacement graph P system, investigating the generation of string graph languages through hyper-edge replacement graph grammars and the non-deterministic parallelism mode of rewriting within P systems.

Definition 2.8: A hyper-edge replacement graph rewriting P system (HRGRPS) [9] is a construct $\Pi = (N_H, V_H, T_H, \mu, M_1, M_2, .., M_n, R_1, R_2, .., R_n, (n, d), i_0)$ where:

- N_H is a finite set of node labels.
- V_H is a finite set of non-terminal and terminal hyper-edge labels.
- T_H is a finite set of terminal hyper-edge labels.
- μ is the membrane structure with n membranes.
- M_i is the finite set of (1,1) hyper-edges over V_H initially present in the region i, where $i = 1, 2, .., n$.
- d is the depth of the membranes, which are labeled with numbers from the set $\{1, 2, .., n\}$, with the skin membrane being labeled as 1.
- R_i is the hyper-edge replacement graph rule, denoted as $R_i = (A \to B(tgt))$, where A is replaced with graph B with the help of attachment instructions.
- $tgt \in \{here, out\} \cup \{in_j | 1 \leq j \leq n\}$.
- i_0 is the output membrane.

This paper will proceed to elucidate the definitions and present corresponding figures for the distinct categories of graphs that will be produced and analyzed herein.

Definition 2.9: A cycle graph [3] C_n, also referred to as an $n-cycle$, is a graph with n nodes that forms a single cycle connecting all nodes.

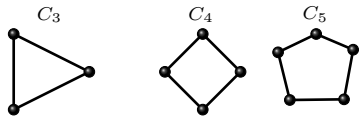

Fig. 1. Cycle graphs

Definition 2.10: The n-pan graph [1] is created by connecting a cycle graph C_n to a singleton graph using an edge.

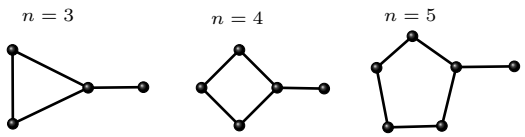

Fig. 2. Pan graphs

Definition 2.11: A prism graph [1], also called a circular ladder graph and denoted by CL_n, is a graph corresponding

to the skeleton of an n-prism.

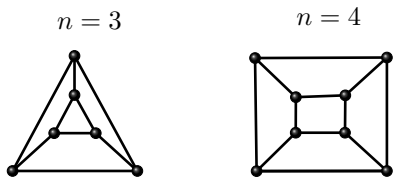

Fig. 3. Prism graphs

A pendant vertex is a vertex with degree one, connected to only one other vertex. A pendant edge is the edge connected to this vertex, with one endpoint having degree one.

Definition 2.12: The n-sunlet graph [1] is a graph with $2n$ vertices created by connecting n pendant edges to a cycle graph C_n.

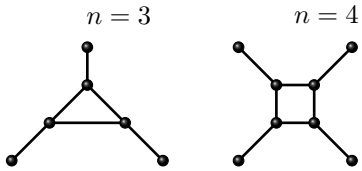

Fig. 4. Sunlet graphs

Definition 2.13: The (m, n)-tadpole graph [1], also known as a dragon graph or kite graph, is formed by linking a cycle graph C_m to a path graph P_n with an edge between them.

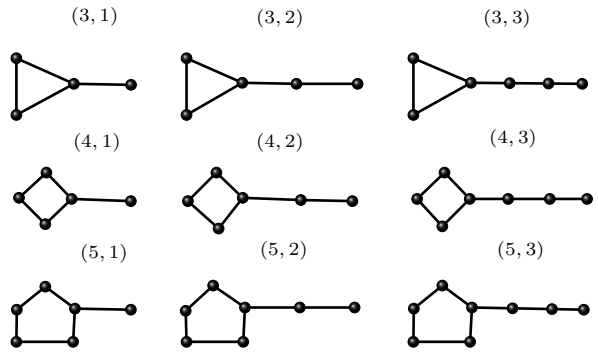

Fig. 5. Tadpole graphs

Definition 2.14: The n-web graph [1] is used as a reference to the stacked prism graph $Y_{(m,n)} = C_m \square P_n$, where C_m represents a cycle graph, P_n represents a path graph, and $\square$ denotes a graph Cartesian product.

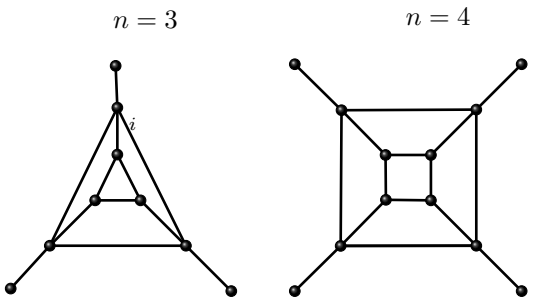

Fig. 6. Web graphs

Definition 2.14: A binary tree [14] is an ordered tree where each node has at most two children.

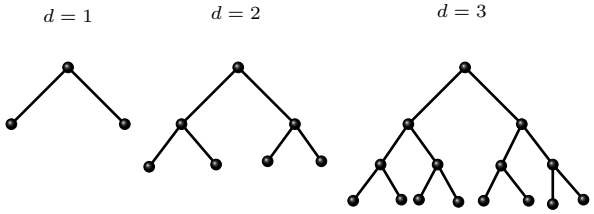

Fig. 7. Binary tree

Definition 2.15: A ternary tree [13] is an ordered tree where each node has at most three children.

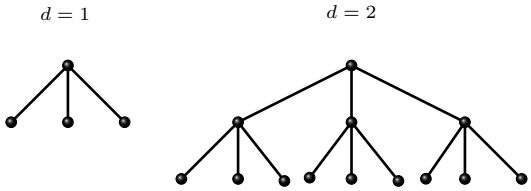

Fig. 8. Ternary tree

Definition 2.16: In graph theory, a k-ary tree [11] is indeed an ordered tree where each node has no more than k children.

Definition 2.17: The n-wheel graph [3] is a type of graph where all the vertices in a cycle are linked to a central universal vertex.

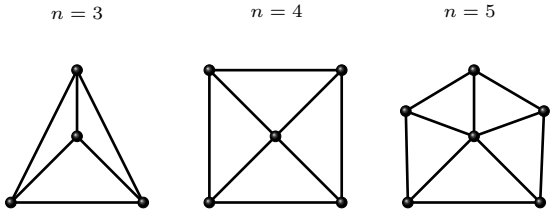

Fig. 9. Wheel graphs

Definition 2.18: The n-gear graph [1], often referred to as a bipartite wheel graph, is a wheel graph with one additional graph vertex inserted between each pair of neighboring graph vertices of the outer cycle.

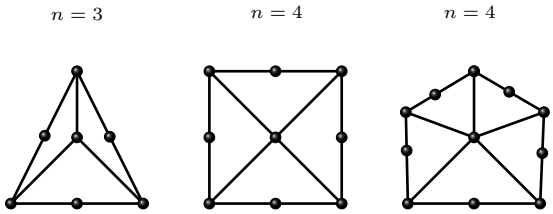

Fig. 10. Gear graphs

Definition 2.19: The helm graph [1] H_n is created by adding a pendant edge to each node of a n-wheel graph.

Definition 2.20: The flower graph [1] Fl_n is created by connecting each pendant vertex to the apex of the helm graph H_n.

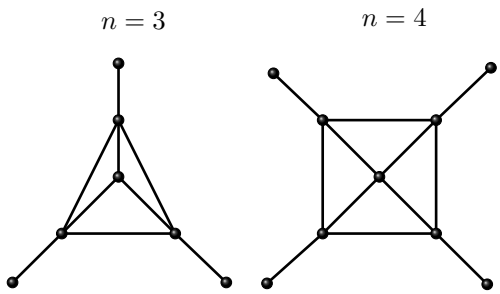

Fig. 11. Helm graphs

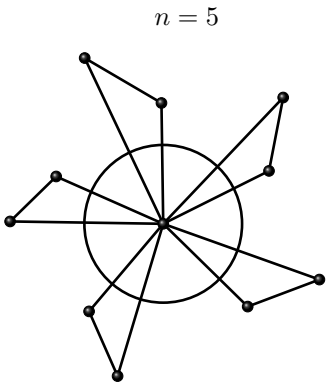

Fig. 12. Flower graph

Definition 2.21: The sunflower graph [1] Sf_n is created by adding n pendant edges to the apex of the flower graph Fl_n.

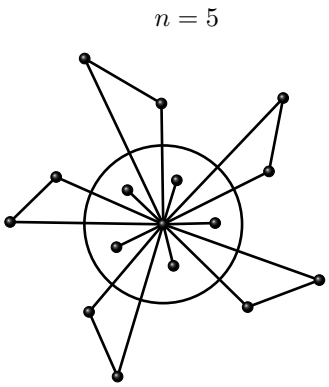

Fig. 13. Sunflower graph

Definition 2.22: A Broom Graph [14] $B_{n,d}$ is a graph consisting of n vertices. It contains a path P comprising d vertices and $(n-d)$ pendant vertices. These pendant vertices are adjacent to either the origin u or the terminus v of the path P.

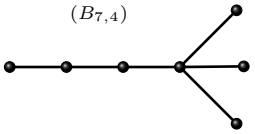

Fig. 14. Broom graph of order (7,4)

III. Generative System of special classes of graphs using HRGRPS

This section aims to generate an extensive collection of graphs associated with cycle, tree, and wheel structures utilizing the hyper-edge replacement graph rewriting P system (HRGRPS). The HRGRPS is instantiated in four distinct systems: Π_C for cycle-related graphs, Π_T for tree-related graphs, Π_B for classes of broom graphs, and Π_W for wheel-related graphs. Although the cycle-related graphs are generated using the same system Π_C, their disparate structural properties necessitate unique initial production rules. Consequently, cycle, pan, tadpole, and prism graphs are generated using a shared initial production rule, whereas sunlet and web graphs require a distinct initial production rule to accommodate their distinct features. Similarly, the tree-related graphs and broom graphs employ disparate initial rules for generating binary, ternary, and k-ary trees and classes of broom graphs, respectively. Analogously, wheel and gear graphs share an initial production rule, whereas helm, flower, and sunflower graphs require a distinct initial production rule to capture their unique features.

GENERATING SYSTEM FOR CYCLE-RELATED GRAPHS:

Theorem 3.1 The set of all cycle C_n, n-Pan, (m,n)-tadpole, n-prism, n-sunlet, and n-web graphs can be generated by hyper-edge replacement graph rewriting P system using three membranes with minimal order rules.

Proof: The HRGRPS is a construct
$\Pi_C = (\{1,2,3,4,5,6\}, \{S_1, S_2, S_3, S_4, a, b, c\}, \{a, b, c\}, [_1[_2]_2[_3]_3]_1], R_1, R_2, R_3, R_4, R_5, T_{S_1}, T_{S_2}, T_{S_3}, T_{S_4}, (3,1), 1)$

The System Π_C comprises of three membranes. Within the first membrane, there exists a non-terminal rule, denoted as R_1, which contains two graphs. The initial graph encompasses the non-terminals S_1, S_2 and S_3 and the secondary graph involves non-terminals S_1,S_2, S_3 and S_4. Moving to the second membrane, it contains non-terminal rule R_2 with non-terminal S_1 and terminal rules T_{S_1}, T_{S_3} and T_{S_4}. In the third membrane, non-terminal rules R_3 with non-terminal S_2, and R_4, with non-terminal S_4, and R_5, with non-terminal S_4 are present along with terminal rule T_{S_2}.

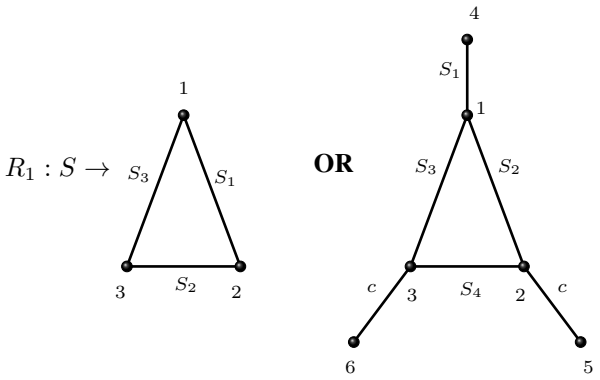

Fig. 15. Initial non-terminal production rule in the first membrane with handle S.

Initiating from the first graph in rule R_1 yields a set of graphs comprising cycle graphs, n-Pan graphs, (m,n)-tadpole graphs, and n-prism graphs. On the other hand, commencing with the second graph in rule R_1 results in the generation of n-sunlet graphs and web graphs.

The Attachment instructions for the given production rules are as follows:

- $R_2 : ((3, 1-1, 2), here)$ and $T_{S_1} : ((1, 1-2, 2), out)$
- $R_3 : ((2, 2-1, 3), here)$ and $T_{S_2} : ((1, 3-2, 2), in_2)$
- $R_4 : ((4, 1-6, 2), here)$ and $T_{S_3} : ((1, 1-2, 3), here)$
- $R_5 : ((1, 1-2, 2), here)$ and $T_{S_4} : ((3, 1-2, 2), here)$

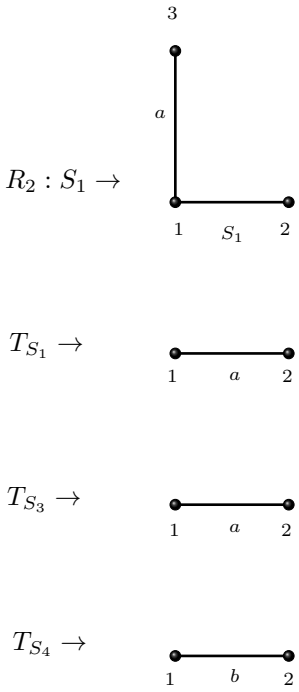

Fig. 16. Non-terminal and terminal production rules in the second membrane.

Generating procedure for the set of all cycle graphs:

To generate a cycle graph of order 3, the initial graph of rule R_1 is applied to the handle S within the first membrane. The resultant graph produced, proceeds to the third membrane, where it applies the terminal rule T_{S_2} once, before transitioning to the second membrane. In the second membrane, it applies the terminal rules T_{S_3}, T_{S_1} once and comes out to the skin membrane.

For the generation of a cycle graph of order 4, the initial graph of R_1 is applied once. And it enters the third membrane, where it employs the rule R_3 once, before advancing to the second membrane. Inside the second membrane, it utilizes the termimal rules T_{S_3} and T_{S_1} once before it comes out.

In a general case, the cycle graph of order n is generated by using the initial graph of R_1 once. Subsequently, the non-terminal rule R_3 is apllied n-3 times within the third membrane, along with the terminal rule T_{S_2} once. Upon entering the second membrane, the terminal rules T_{S_3} and T_{S_1} are each used once.

Generating procedure for the set of all pan graphs:

The initial graph from rule R_1 is utilized on the handle S in the first membrane to produce a pan graph of order 3. The produced graph then transitions to the third membrane. Within this third membrane, it employs the terminal rule T_{S_2}, subsequently advancing to the second membrane. In the second membrane, the produced graph adheres to rule R_2 and remains there, following which it applies the terminal rules T_{S_3} and T_{S_1} and comes out.

In a general sense, the pan graph of order n is generated by employing the initial graph of R_1 once, resulting in a graph featuring non-terminals S_1, S_2, and S_3. Subsequently, it progresses to the third membrane, where it utilizes R_3 $n-3$ times along with

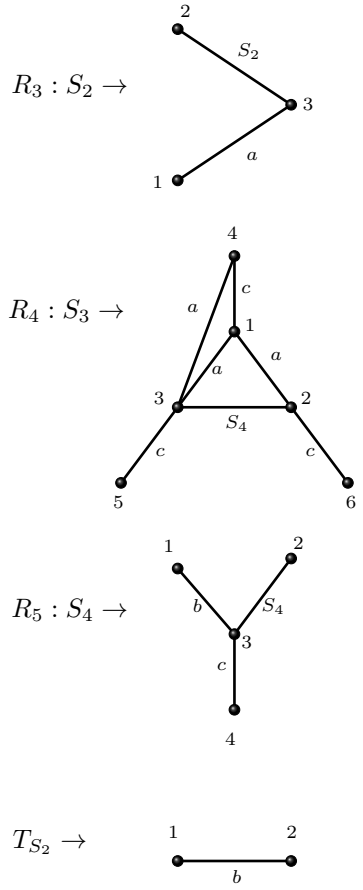

Fig. 17. Non-terminal and terminal production rules in the third membrane.

T_{S_2} once. Afterward, it transitions to the second membrane, where it employs each of the rules R_2, T_{S_3} and T_{S_1} once to generate the desired n^{th} order.

Generating procedure fornthe set of all tadpole graphs:

The (m, n) order tadpole graph is created by initiating the process with the initial graph of R_1 once. The resulting graph then proceeds to the third membrane, where it undergoes R_3 n-3 times and utilizes T_{S_2} once. Following this, it advances into the second membrane, applying the R_2 rule m times, T_{S_3} and T_{S_1} once.

Generating procedure for the set of all prism graphs:

The prism graph of order n is produced through the initial application of the graph from R_1. Subsequently, this resultant graph traverses into membrane three, where it engages R_4 once, as well as employing R_3 and R_5 rules once each. Following this, it makes use of T_{S_2} before transitioning into the second membrane. Once within the second membrane, it elegantly employs the terminal rules T_{S_3}, T_{S_4}, and T_{S_1} each once.

Generating procedure for the set of all sunlet graphs:

The sunlet graph of the n^{th} order is produced by utilizing the secondary graph of R_1 on the handle S within the first membrane, following which it transitions to the third membrane. There, it applies rule R_5 n-3 times and T_{S_2} once before moving to the second membrane. Within the second membrane, it proceeds to employ all terminal rules T_{S_3}, T_{S_4}, and T_{S_1} once.

Generating procedure for the set of all web graphs:

The web graph of order n is generated by employing the secondary graph of R_1 on the handle S in the first membrane. It then progresses to the third membrane, where it applies rule R_4 once, R_5 $2n$-6 times, and T_{S_2} once before transitioning to the second membrane. Within the second membrane, it proceeds to utilize each terminal rule T_{S_3}, T_{S_4}, and T_{S_1} once.

■

GENERATING SYSTEM FOR TREE-RELATED GRAPHS:

Theorem 3.2 The set of binary, ternary and k-ary trees can be generated by hyper-edge replacement graph rewriting P system using three membranes with minimal order rules.

Proof: The HRGRPS is a construct
$\Pi_T = (\{1,2\}, \{S_1, a, b\}, \{a, b\}, [_1[_2]_2][_3]_3]_1], R_1, R_2, T_{S_1}, (2,1), 1)$

The System Π_T consist of two membranes. Within the first membrane, there exists a non-terminal rule, denoted as R_1 and it contains three graphs. All the three graphs encompasses the non-terminal S_1. The second membrane consist of a non-terminal rule R_2 with non-terminal S_1 and a terminal rule T_{S_1}. In this case R_2 is made up of three graphs.

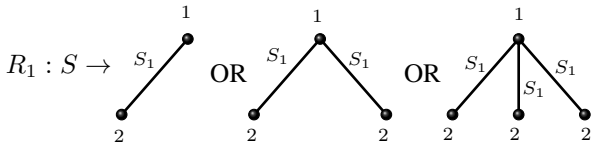

Fig. 18. Initial non-terminal production rule in the first membrane with handle S for generating the set of all binary and ternary trees.

Starting with r_1 of R_1 produces a set of binary graphs with a single child, and starting with r_2 of R_1 produces a set of binary graphs with two children. A set of ternary graphs is produced by taking r_3 of R_1.

The Attachment instruction for the given production rule are as follows:

$$R_2 : ((1, 1-1, 2), here) \text{ and } T_{S_1} : ((1, 1-2, 2), out)$$

Generating procedure for the set of binary trees:

- **Binary tree with one child:**
  To generate a binary tree with one child at a depth of 2, apply the initial rule r_1 of R_1 within the first membrane. Subsequently, it enters the second membrane and executes the rule r_4 once, residing within the second membrane due to the existence of the target 'here'. It employs the terminal rule T_{S_1} and emerges out.
  To generate a binary tree with a single child at a depth of n, execute the first rule r_1 from R_1 in the first membrane. Afterwards, it passes through the second membrane and iteratively applies the rule r_4 $n-1$ times, while employing the terminal rule T_{S_1}, until finally exiting.
- **Binary tree with two children:**
  Applying the initial graph r_2 of rule R_1 to the handle S in the first membrane yields a binary tree of depth 2. Once it enters the second membrane, the generated resultant graph applies the non-terminal rule r_5 of R_2 and terminal rule T_{S_1} once.
  The binary tree of depth n with two children can be obtained by applying the rule r_2 of R_1 once and r_5 of R_2 $n-1$ times and T_{S_2} once.

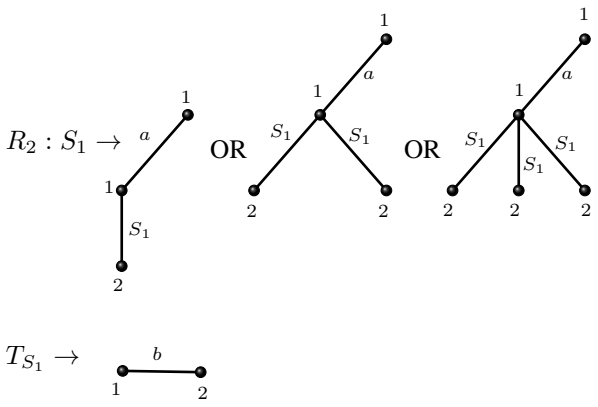

Fig. 19. Non-terminal and terminal production rule in the second membrane.

Generating procedure for the set of ternary trees:

- **Ternary tree with one child:**
  The initial graph r_3 of rule R_1 is applied to the handle S in the first membrane in order to create a ternary tree of depth 2. The resulting graph applies the non-terminal rule R_2 when it enters the second membrane. A ternary tree of depth two with one child is produced if r_4 of R_2 is used twice. The ternary tree of depth n with one child will be generated from applying the previously stated rule $n-1$ times.
- **Ternary tree with two children:**
  To generate a ternary tree of depth 2 with two children if it employs r_5 of R_2 and T_{S_1} once. We shall obtain the ternary tree of depth n with two children if we apply the non-terminal rule r_5 of R_2 $n-1$ times and T_{S_1} once.
- **Ternary tree with three children:**
  To generate a ternary tree of depth 2 with three children if it employs r_6 of R_2 and T_{S_1} once. We shall obtain the ternary tree of depth n with three children if we apply the non-terminal rule r_6 of R_2 $n-1$ times and T_{S_1} once.

Generating procedure for the set of k-ary trees:
For generating the set of k-ary trees, start with the generalized rule of having k children as the initial graph. After entering the second membrane it uses the non-terminal rule with k children and terminal rule T_{S_1} and emerges out.

■

GENERATING SYSTEM FOR WHEEL-RELATED GRAPHS:

Theorem 3.3 The set of all n-wheel, n-gear, helm H_n, flower Fl_n and sunflower Sf_n graphs can be generated by hyper-edge replacement graph rewriting P system using three membranes with minimal order rules.

Proof: The HRGRPS is a construct
$\Pi_W = (\{1,2,3,4,5,6\}, \{S_1, S_2, S_3, a, b, c\}, \{a, b, c\}, [_1[_2]_2][_3]_3]_1], R_1, R_2, R_3, R_4, R_5, T_{S_1}, T_{S_2}, T_{S_3}, (3,1), 1)$

The System Π_W consist of three membranes. Two non-terminal rules, designated R_1 with non-terminal S_1 and R_3 with non-terminals S_2 and S_3, make up the system's first membrane. R_1 contains two graphs: the initial graph and the secondary graph. Both graphs include the non-terminal S_1 and the second membrane has terminal rules T_{S_1} and T_{S_3}, as well as non-terminal rules R_2 with non-terminal S_1, and R_5 with non-terminal S_3. In this case, R_2 and R_5 are made up of two graphs: the primary and the secondary graphs. Moving on to the third membrane, it has a terminal rule T_{S_2} and a non-terminal rule R_4 with non-terminals S_2 and S_3.

Beginning with the initial graph in R_1, the set of all wheel graphs is generated. Furthermore, commencing with the secondary graph in R_1 produces the complete set of gear graphs. Starting with the rule R_3, the set of helm, flower, and sun-flower graphs is generated.

The Attachment instruction for the given production rules are as follows:

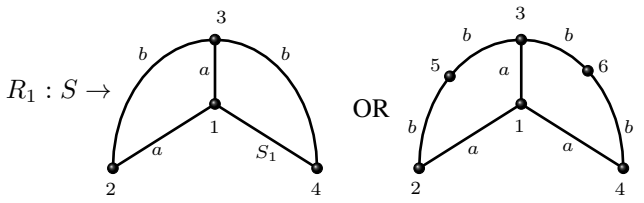

Fig. 20. Initial non-terminal production rule in the first membrane with handle S for generating the set of all wheel and gear graphs.

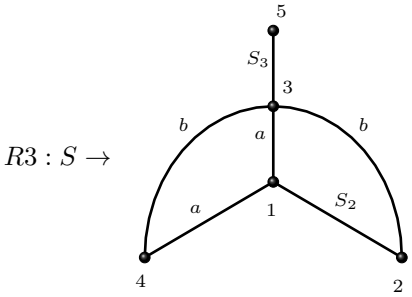

Fig. 21. Initial non-terminal production rule in the first membrane with handle S for generating the set of all helm, flower and sunflower graphs.

- $R_2 : ((1, 1-2, 4), here)$ and $T_{S_1} : ((1, 1-2, 4), out)$
- $R_4 : ((1, 1-2, 4), here)$ and $T_{S_2} : ((1, 1-2, 4-4, 2), in_2)$
- $R_5 : ((1, 3-2, 5-3, 1), here)$ and $T_{S_3} : ((1, 3-2, 5), out)$

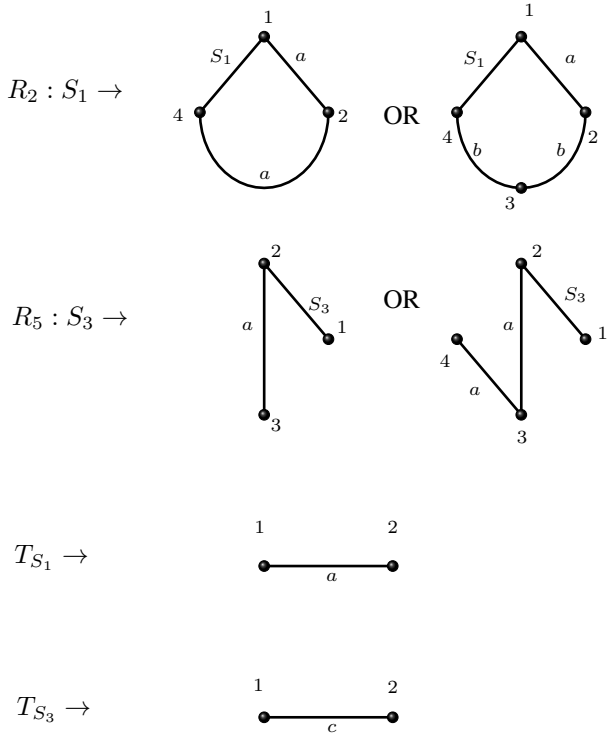

Fig. 22. Non-terminal and terminal production rules in the second membrane.

Generating procedure for the set of all wheel graphs:

In order to produce a wheel graph of order 3, the handle S in the first membrane is subjected to the initial graph of rule R_1. After entering the second membrane and applying the primary graph of non-terminal rule R_2 once, the graph persists in that region as the specified target is 'here'. Finally, it applies the terminal rule T_{S_1} only once before exiting the output membrane.

To produce a wheel graph of order 4, the initial graph of R_1 is utilized only once. Additionally, it traverses the second membrane, where it applies the primary graph of non-terminal rule R_2 twice and the terminal rule T_{S_1} once.

The wheel graph of order n is produced by iterating the initial graph of R_1 once and applying the primary graph of non-terminal rule R_2 and terminal rule T_{S_1} in the second membrane n-2 times and once, respectively.

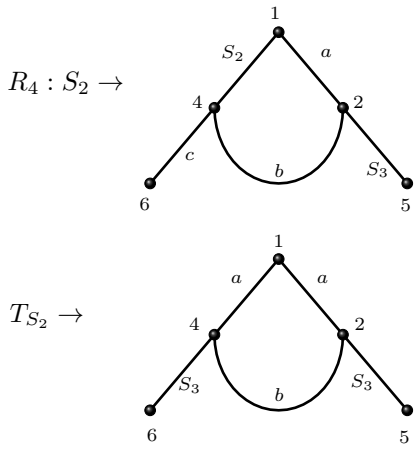

Fig. 23. Non-terminal and terminal production rules in the third membrane.

Generating procedure for the set of all gear graphs:

The gear graph of order n is generated by iterating the initial graph of R_1 once and applying the secondary graph of non-terminal rule R_2 and terminal rule T_{S_1} in the second membrane n-2 times and once, respectively.

Generating procedure for the set of all helm graphs:

Applying rule R_3 to the handle S in the skin membrane generates a helm graph of order 3. The graph produced passes through the third membrane, applies the terminal rule T_{S_2} once, and proceeds to the second membrane. There, it applies the rule T_{S_3} thrice and then exits.

The helm graph of order n is produced by using the non-terminal rules R_3 once, R_4 n-3 times and the terminal rules T_{S_2} once and T_{S_3} n times.

Generating procedure for the set of all flower graphs:

The flower graph of order n is produced by using the non-terminal rules R_3 once, R_4 n-3 times and initial graph of R_5 n times and the terminal rules T_{S_2} once and T_{S_3} n times.

Generating procedure for the set of all sunflower graphs:

The sunflower graph of order n is produced by using the non-terminal rules R_3 once, R_4 n-3 times and secondary graph of R_5 n times and the terminal rules T_{S_2} once and T_{S_3} n times.

■

GENERATING SYSTEM FOR CLASSES OF BROOM GRAPHS:

Theorem 3.4 The set of all broom graphs $B_{n,d}$ can be generated by hyper-edge replacement graph rewriting P system using three membranes with minimal order rules.

Proof: The HRGRPS is a construct Π_B= $(\{1, 2, 3, 4\}, \{S_1, S_2, a, b\}, \{a, b\}, [_1[_2]_2][_3]_3]_1], R_1, R_2, T_{S_1}, T_{S_2}, (3, 1), 1)$

The system denoted as Π_B consists of three membranes.

The initial membrane has a non-terminal rule R_1 that involves the non-terminals S_1 and S_2. The second membrane is composed of terminal and non-terminal rules, denoted as T_{S_1} and R_2 accordingly, where S_1 is the non-terminal symbol associated with R_2. The third membrane consists of terminal and non-terminal rules T_{S_2} and R_3, respectively, with the non-terminal S_2.

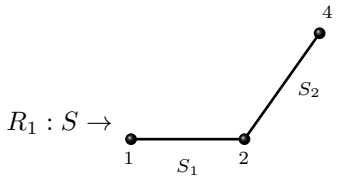

Fig. 24. Initial non-terminal production rule R_1 in the first membrane with the handle S.

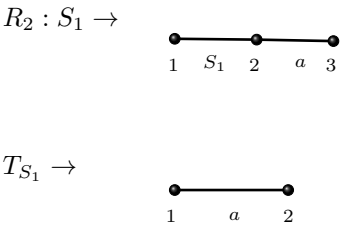

Fig. 25. Non-terminal and terminal production rules R_2 and T_{S_1} in the second membrane.

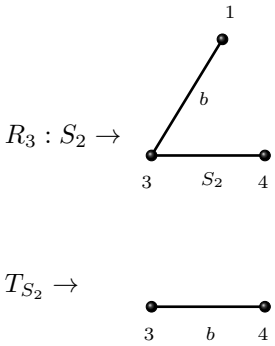

Fig. 26. Non-terminal and terminal production rules R_3 and T_{S_2} in the third membrane.

The Attachment instruction for the given production rules are as follows:

- $R_2 : ((2, 1-3, 2), here)$ and $T_{S_1} : ((1, 1-2, 2), in_3)$
- $R_3 : ((3, 3-4, 1), here)$ and $T_{S_2} : ((3, 3-4, 4), out)$

Generating procedure for the broom graphs of order $B_{n,2}$:

For generating a broom graph with an order of $B_{3,2}$ employ the non-terminal production rule R_1 on the handle S placed in the initial membrane. After that, it passes through the second membrane and employs the rule T_{S_1} to terminate the non-terminal S_1. Subsequently, it proceeds to the third membrane and employs the rule T_{S_2} to terminate the non-terminal S_2.

The broom graph of order $B_{4,2}$ is generated by applying the rule R_1 to the handle S. Following that, it uses the rule T_{S_1} once in the second membrane and applies the rules R_3 and T_{S_2} once in the third membrane.

To generate a broom graph of order $B_{5,2}$ is generated by applying the rule R_1 to the handle S. Subsequently, it applies the rule T_{S_1} once in the second membrane and employs the rule R_3 twice, as well as T_{S_2} once in the third membrane.

The broom graph of order $B_{n,2}$ is generated by using the rule R_1 to the handle S. Next, it applies the rule T_{S_1} once in the second membrane and uses the rule R_3 $n-3$ times and T_{S_2} once in the third membrane.

Generating procedure for the broom graphs of order $B_{n,3}$:

The broom graph of order $B_{4,3}$ is produced by applying the non-terminal production rules R_1 to the handle S in the initial membrane. After that, it applies the non-terminal rule R_2 and the terminal rule T_{S_1}, once in the second membrane. Then, it enters the third membrane uses non-terminal rule R_3 once and terminal production rule T_{S_2} once.

To produce the broom graph of order $B_{5,3}$, apply the non-terminal production rules R_1 to the handle S in the initial membrane. Then, it applies the non-terminal rule R_2 and the terminal rule T_{S_1}, once in the second membrane. After that, it enters the third membrane uses non-terminal rule R_3 twice and terminal production rule T_{S_2} once.

To produce the broom graph of order $B_{6,3}$, apply the non-terminal production rules R_1 to the handle S in the initial membrane. Next, it applies the non-terminal rule R_2 and the terminal rule T_{S_1}, once in the second membrane. Subsequently, it enters the third membrane uses non-terminal rule R_3 thrice and terminal production rule T_{S_2} once.

The broom graph of order $B_{n,3}$ is produced by applying the non-terminal production rules R_1 to the handle S in the initial membrane. Then, it applies the non-terminal rule R_2 and the terminal rule T_{S_1}, once in the second membrane. After that, it enters the third membrane uses non-terminal rule R_3 $n-3$ times and terminal production rule T_{S_2} once.

Generating procedure for the broom graphs of order $B_{n,4}$:

For generating the broom graph of order $B_{5,4}$, use the rule R_1 twice in the first membrane, R_2 twice and T_{S_1} once in the second membrane and R_3 and T_{S_2} once in the third membrane.

The broom graph of order $B_{n,4}$ is generated by applying the non-terminal rules R_1 once, R_2 twice, R_3 $n-3$ times and the terminal rules T_{S_1} and T_{S_2} once.

Generating procedure for the broom graphs of order $B_{n,d}$:

The broom graph of order $B_{n,4}$ is generated by applying the non-terminal rules R_1 once, R_2 $d-2$ times, R_3 $n-3$ times and the terminal rules T_{S_1} and T_{S_2} once.

■

IV. Conclusion

The proposed research aims to develop a comprehensive repository of cycle, tree, and wheel graphs, along with their analogous structures, including pan, tadpole, prism, sunlet,

and web graphs, as well as binary, ternary, and k-ary trees, gear, helm, flower, sunflower, and broom graphs. Cycle and wheel graphs are well-regarded for their applicability in analyzing complex game structures, particularly those involving sequential player interactions, thus aiding in the examination of strategic decision-making, equilibria, and outcomes. Both game theory and networking widely utilize binary and ternary trees, providing robust frameworks for representing and managing intricate information structures. Moreover, the modeling of social and communication networks employs broom graph structures, which facilitate the analysis of coalition formation and power dynamics within these networks. In game theory, these graphs assist in the visualization and optimization of decision-making processes, while in networking, they play a critical role in ensuring efficient data handling and routing. Consequently, future research will concentrate on generating all locally similar graphs to cycle and wheel graphs, which hold significant implications for network formation and resource allocation within the context of game theory.

REFERENCES

[1] A. Brandstadt, V. B. Le and J. P. Spinrad, "Graph Classes: A survery," *Society for Industrial and Applied Mathematics*, 1999.

[2] F. Drewes, H. J. Kreowski, and A. Habel, "Handbook of graph grammars and computing by graph transformation," *World Scientific Publishing*, pp. 95-162, 1997.

[3] S. Pemmaraju and S. Skiena, "Cycles, stars, and wheels," *Computational Discrete Mathematics Combinatorics and Graph Theory in Mathematica*, pp. 248-249, 2003.

[4] A. Habel, "Hyperedge replacement: grammars and languages," *Springer Science and Business Media*, vol. 643, 1992.

[5] A. Habel, "Introduction to hyperedge-replacement grammars. Hyperedge Replacement: Grammars and Languages," pp. 5-42, 1992.

[6] G. Paun, G. Rozenberg, "A guide to membrane computing," *Theoretical Computer Science*, vol. 287, no. 1, pp. 73-100, 2002.

[7] G. Paun, "Membrane computing," *Scholarpedia*, vol. 5, no. 1, p.9259, 2010.

[8] M. P. Sankar, N. G. David, and D. G. Thomas, "Hyperedge replacement graph P system," *Sixth International Conference on Bio-Inspired Computing: Theories and Applications, IEEE*, pp. 272-277, 2011.

[9] M. P. Sankar, N. G. David, "On Generative Power of Rewriting Graph P System," *In Artificial Intelligence and Evolutionary Computations in Engineering Systems, Springer*, Singapore, pp. 419-429, 2020.

[10] R. Nakano, "Error Correction of Enumerative Induction of Deterministic Context-free L-system Grammar," *IAENG International Journal of Computer Science*, vol. 40, no. 1, pp. 47-52, 2013.

[11] S. Cleary, "Restricted rotation distance between k-ary trees," *Journal of Graph Algorithms and Applications*, pp. 19-33, 2023.

[12] J. L. Pfaltz, "Representing graphs by Knuth trees," *Journal of the ACM(JACM)*, pp. 361-366, 1975.

[13] F. Frati, "Straight-line orthogonal drawings of binary and ternary trees," *InGraph Drawing: 15th International Symposium, Springer*, pp. 76-87, 2007.

[14] P. Ghosh, A. Pal, "Some results of labeling on broom graph," *Journal of Advances in Mathematics*, vol. 9, no. 9, 2015.