TWMS J. App. and Eng. Math. V.14, N.3, 2024, pp. 1099-1108

GENERATION OF GRAPHS USING HYPER-EDGE REPLACEMENT GRAPH REWRITING P SYSTEM

K. VINODHINI[1], M. P. SANKAR[1*], §

Abstract. Hyper-edge replacement graph grammar is considered a kernel in generating graphs and hypergraphs, and it has stood out enough to be noticed in recent years. George Paun proposed Membrane computing, frequently known as P system, as a bio-inspired model or a model of natural computing. The field of Membrane computing is first roused by the manner in which nature processes at the cell levels. In this paper, using the hyper-edge replacement graph rewriting P system, significant graphs like Snail graphs, Caterpillar graphs, Comb graphs, Fire Cracker graphs, and Wheel graphs are generated with hyper-edge rules of minimum order.

Keywords: Hyper-edge, Hyper-edge replacement graph grammar (HRG), Graph P system, Hyper-edge replacement graph rewriting P system (HRGRPS).

AMS Subject Classification: 68Q42, 68R10, 68Q10.

1. Introduction

The notion of formal grammars on strings has been outstretched to grammars on graphs, resulting in graph grammars. It provides a tool for modeling the local transformation of graphs in a mathematically accurate manner. Node replacement and edge replacement are the two most primordial options for rewriting a graph, but hyper-edge replacement is included in the more general situation [1]. An atomic object that has an ordered set of inbound tentacles attached to the nodes via the source function and an ordered set of outbound tentacles attached to the nodes via the target function is known as a hyper-edge [2]. Hyper-edge replacement graph grammar is a rudimentary methodology for rewriting graphs and hypergraphs. Feder and Pavlidis instituted this term in the early seventies, and many analysts in the later part of the seventies have colossally explored it. In this grammar, the hyper-edges with non-terminal labels are replaced by graphs with terminal and non-terminal hyper-edge labels. The set of all graphs generated by the above-mentioned grammar is called Hyper-edge replacement language. The notion HRG

[1] Department of Mathematics, SRM Institute of Science and Technology, Kattankulathur, 603203, Chengalpattu, TamilNadu, India.
e-mail: vk8271@srmist.edu.in; ORCID: https://orcid.org/0000-0001-9161-3440.
e-mail: meenap@srmist.edu.in; ORCID: https://orcid.org/0000-0002-2860-7862.
* Corresponding author.
§ Manuscript received: August 08, 2022; accepted: March 15, 2023.

stands for all Hyper-edge replacement grammars, while HRL stands for all Hyper-edge replacement languages.

In 1998, a novel Membrane computing [3] paradigm was proposed, which was educed from biology. This computing was proposed because it endeavored to model what occurs in a cell: the behavior of proteins, enzymes, membrane structure, how they evolve, and how they peregrinate through the membranes as they transit. It was introduced by Professor George Paun of Romania, and accordingly, it is known as the P system. Initially, a theoretical model was created, and the first article was published in 2000. It belongs to the family of distributed parallel computing devices because whatever happens within the membrane occurs parallel [4]. Then the message passes from inside to outside of each membrane, and while crossing the membrane, they carry some information. Membrane structure, multisets, and rules are considered as the three indispensable aspects of Membrane computing. Here, multisets may be enzymes, proteins, or chromosomes later on strings, arrays, graphs, etc. These multisets of items mutate throughout time as they transit from one membrane to another, and a set of rules governs this.

The set of all Snail graphs and Wheel graphs cannot be generated by HRG of minimum order (order less than 3) [5]. The proposed idea of this paper is to generate with smallest possible order and to build the set of all Caterpillar graphs as well as some notable graphs such as Comb graphs and Fire Cracker graphs using the conventional model (HRGRPS).

2. PRELIMINARIES

The graphs which have been employed in this paper are simple and undirected. The standard representation of a graph G is a tuple (V, E) where V denotes the node set and E denotes the edge set in which each edge connects precisely two nodes, whereas a hypergraph is a graph in which each edge called hyper-edge connects an arbitrary number of nodes instead of two nodes. Today's real-world social networks heavily rely on hypergraphs. So, the hyper-edge and hypergraphs are used as atomic items in hyper-edge replacement graph grammar, which is most powerful among all other existing grammars. This section recalls some basic notions regarding hypergraphs and hyper-edge replacement graph grammar.

Definition 2.1. *[5] A hypergraph H over an arbitrary, but fixed set of labels C consisting (V, E, s, t, l) where V denotes the node set (finite), E denotes the hyper-edge set (finite), s is the source function which assigns each hyper-edge source nodes denoted by $s(e)$ and t is the target function which assigns each hyper-edge target nodes denoted by $t(e)$ where $e \in E$ and l is the labelling function which gives each hyper-edge a label. The set of all hypergraphs over the label set C is denoted by $\mathbb{H}_C$.*

Definition 2.2. *[5] A hypergraph $H \in \mathbb{H}_C$ is said to be a handle if $E_H = \{e\}$, $s_H(e) = begin_H$ and $t_H(e) = end_H$. Here $begin_H, end_H \in V^*$ where V^* denotes the set of all finite sequence of nodes over V.*

The ordered pair (m, n) denotes the type of the handle, in which m denotes the number of source nodes of e and n denotes the number of target nodes of e.

Definition 2.3. *[5] A hyper-edge replacement grammar denoted by HRG consist of four components (N, T, P, S) where*

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

Definition 2.4. *[6] The attaching nodes are defined as 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. *[5] For* $H \in \mathbb{H}_C$*, the set of nodes occuring in the sequence* $ext_H = begin_H.end_H$ *is called the set of external nodes of* H *and is denoted by* EXT_H*.*

Definition 2.6. *[5] If* $\forall\ (A, R)$ *belongs to* P*,* $|EXT_R| \leq r$ *then a hyper-edge replacement grammar is said to have an order* r *for some* $r \in \mathbb{N}$*.*

Definition 2.7. *[5] The hypergraph language* $L(HRG)$ *generated by* HRG *consist of all terminal labeled hypergraphs which can be derived from* S *by applying productions of* P*,*

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

A simple method of rewriting hypergraphs and graphs called "hyper-edge replacement" was first introduced in the early 1970s. The idea of hyper-edge replacement graph grammar is extended to the hyper-edge replacement graph P system as a result of studies on hyper-edge replacement graph grammars generating string graph languages and by the non-deterministic parallelism mode of rewriting P system.

Definition 2.8. *[7] A hyper-edge replacement graph rewriting P system (HRGRPS) 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 by numbers in the set* $\{1, 2, .., n\}$ *with the skin membrane being labelled as 1,*
R_i *denotes the finite set of hyper-edge replacement graph rules in membrane i such that the rules in* R_i *is of the form* $(A \to B(tgt))$ *where A is replaced with the graph B with the help of attachment instructions,*
$tgt \in \{here, out\} \cup \{in_j | 1 \leq j \leq n\}$*, and*
i_0 *is the output membrane.*

Definition 2.9. *[5] A Snail graph is a graph in which the edges are rotated to create a spiral appearance like a snail's head.*

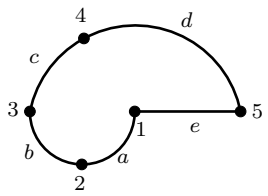

Fig 2.1: Snail graph

Definition 2.10. *[8] A Caterpillar graph, Caterpillar tree, or just "Caterpillar," is a tree with all its vertices within one distance of a central path.*

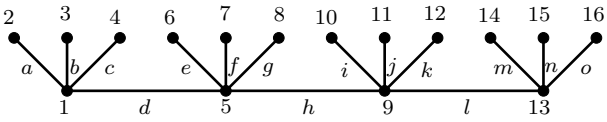

Fig 2.2: Caterpillar graph

Definition 2.11. *[9] A Comb graph is formed by connecting a single pendant edge to each vertex of a path.*

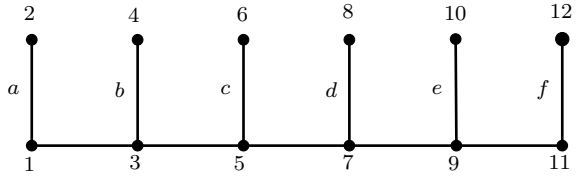

Fig 2.3: Comb graph

Definition 2.12. *[8] A Fire cracker graph is a graph formed by concatenating stars by connecting one leaf from each.*

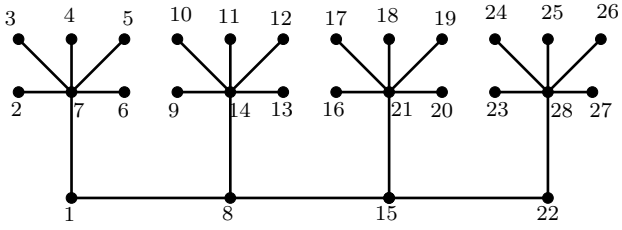

Fig 2.4: Fire cracker graph

Definition 2.13. *[5] A Wheel graph is a graph in which all the vertices in a cycle are connected by a single universal vertex.*

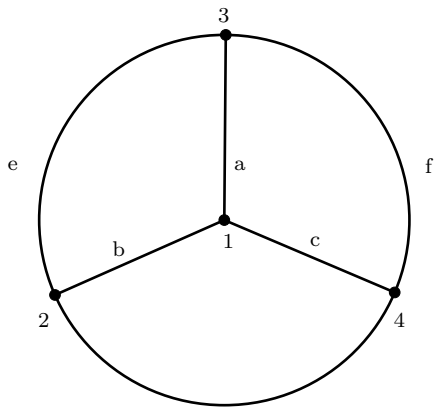

Fig 2.5: Wheel graph

3. MAIN RESULTS

This section generates Snail graphs, Caterpillar graphs, Comb graphs, Fire cracker graphs, and Wheel graphs using the hyper-edge replacement graph rewriting P system. All the graphs mentioned above have the same generating system (HRGRPS), but due to their various structural differences, they have different labels and production rules. Since the structures of the Caterpillar, Comb, and Fire Cracker graphs are identical, the attachment instructions are the same for all these three graphs. Thus, the generation of these classes of graphs is accomplished in a single theorem. Due to the complex structure (an additional node appears for each spiral rotation) of the Snail graph, the generation of the Snail graphs cannot be categorized under the previously listed classes. Similarly, the production rules and attachment instructions for the Wheel graphs are also entirely distinct from all other graphs. Hence, the set of all Snail graphs and Wheel graphs is generated in a separate theorem.

Generally, R_i denotes the finite set of production rules in the membrane i. For avoiding confusions, the notation R_{ij} represents the j^{th} rule in the i^{th} membrane. For instance, R_{12} represents the 2^{nd} rule in the 1^{st} membrane.

The attachment nodes is of the form (a, b). Here, a represents the node label in the right hand side rule of R_{ij} and b represents the node label in the left hand side rule of R_{ij}.

Theorem 3.1. *The set of all Snail graphs can be generated by hyper-edge replacement graph rewriting P system using two membranes with rules of order 2.*
The HRGRPS is a construct

$$\Pi_S = (\{1,2,3,4,5\}, \{S_1, a, b\}, \{a,b,c,d,e,f,g,h,i\}, [_1[_2]_2]_1], R_1, R_2, (2,1), 1).$$

Generally, R_1 denotes the set of production rules in the first membrane and R_2 denotes the set of production rules in the second membrane. Here, R_1 consist of only one rule denoted by $R_{11}$and R_2 consists of two rules R_{21}, and R_{22}.

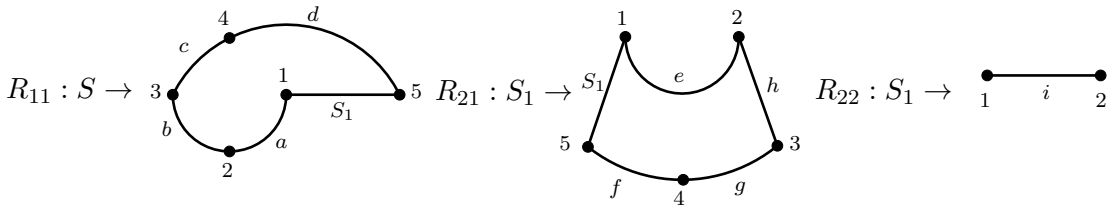

Fig 3.1.1: The production rules for generating Snail graphs

Attachment nodes for R_{21}*:* $((2, H), (3, T), (1, A), here)$ *and* R_{22}*:* $((1, H), (2, T), out)$.

Proof. At first, the node label in R_{11} designated as 1 is the Head node(H), the node label in R_{11} designated as 5 is the Tail node(T), and the attachment node(A) is the node next to the head node on the left. After using the production rule R_{21}, the head node and tail node succeeds by one and two respectively for the connecting purpose, and the attachment node remains as stated previously.

Initially, S is the $(1, 1)$ handle in the skin membrane. A graph with the non-terminal S_1 is produced after R_{11} is utilized and enters the second membrane. After entering, it can apply either R_{21} or R_{22}. If it chooses R_{22}, then the resultant graph produced by applying R_{11} becomes the left hand side rule of R_{22}. The attachment nodes for R_{22} is $((1, H), (2, T), out)$. Here the 1 and 2 represents the node labels in the right hand side rule of R_{22} and H and T represents the node labels in the left hand side rule of R_{22}. Now by applying R_{22}, the Snail graph with a head component is constructed and emerges to the skin membrane, since the target is *out*. For generating the Snail graph of order one, R_{11} is applied to the graph with handle S to generate a graph with non-terminal S_1. If it chooses R_{21}, then the resultant graph produced by applying R_{11} becomes the left hand side rule of R_{21}. The attachment nodes for R_{21} is $((2, H), (3, T), (1, A), here)$. Here the $2, 3$, and 1 represents the node labels in the right hand side rule of R_{21} and H, T, and A represents the node labels in the left hand side rule of R_{21}. Now by applying R_{21} once, it builds a Snail graph of order one with the non-terminal S_1 and stays in the membrane two since the target is *here*. The resultant graph uses the terminal rule R_{22} in membrane two and thus generates the Snail graph of order one.The process repeats until the desired order is generated.

In general, R_{11} is applied to handle S to construct a graph with non-terminal S_1. It employs R_{21} n times after entering membrane two and it utilizes terminal rule R_{22}, thus generates the Snail graph of order n. □

Theorem 3.2. *The set of all Caterpillar graphs, Comb graphs, and Fire Cracker graphs can be generated by hyper-edge replacement graph rewriting P system using two membranes with rules of order 2.*
The HRGRPS for Caterpillar, Comb, and Fire Cracker graphs is a construct

$$\Pi_C = (\{1, 2, 3, 4, 5, 6, 7, 8\}, \{S_1, S_2, S_3, a, b, c\}, \{a, b, c\}, [_1[_2]_2]_1], R_1, R_2, (2, 1), 1).$$

Generally, R_1 *denotes the set of production rules in the first membrane and* R_2 *denotes the set of production rules in the second membrane. Here,* R_1 *consists of three rules* R_{11}, R_{12}*, and* R_{13} *and* R_2 *consists of six rules* $R_{21}, R_{22}, R_{23}, R_{24}, R_{25}$*, and* R_{26}*.*

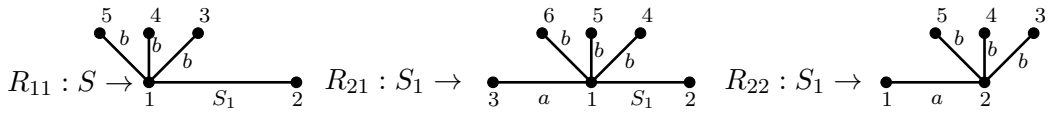

Fig 3.2.1: The production rules for generating Caterpillar graphs

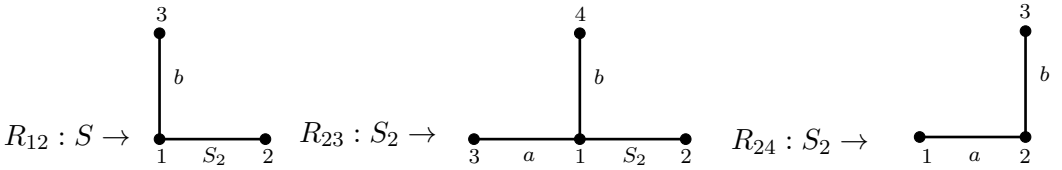

Fig 3.2.2: The production rules for generating Comb graphs

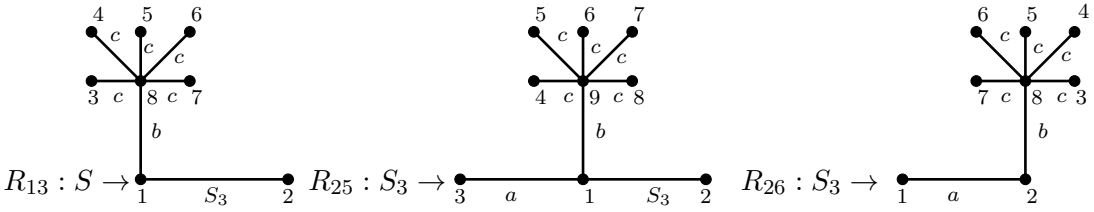

Fig 3.2.3: The production rules for generating Fire Cracker graphs

Attachment nodes for R_{21}, R_{23}, *and* R_{25}: $((3,1),(1,2),here)$ *and* R_{22}, R_{24}, *and* R_{26}: $((1,1),(2,2),out)$.

Proof. The system Π_C consists of two membranes. In first membrane, R_1 has three rules R_{11}, R_{12}, and R_{13} and in second membrane, R_2 has six rules $R_{21}, R_{22}, R_{23}, R_{24}, R_{25}$, and R_{26}, in which R_{22}, R_{24}, and R_{26} are terminal rules. Intially, S is the (1,1) handle in the first membrane.

- If the rule R_{11} is chosen, the set of all Caterpillar graphs are generated by using R_{21} and R_{22}.
- If the rule R_{12} is chosen, the set of all Comb graphs are generated by using R_{23} and R_{24}.
- If the rule R_{13} is chosen, the set of all Fire Cracker graphs are generated by using R_{25} and R_{26}.

Generation of the set of all Caterpillar graphs:

For generating the set of all Caterpillar graphs, the rule R_{11} with non-terminal S_1 is applied to the (1,1) handle S in the first membrane. The resultant graph produced enters the second membrane. In membrane two, it can apply either R_{21} or R_{22}. If it applies R_{22} with the help of the attachment nodes (as explained in the Theorem 3.1), the Caterpillar graph of order two is generated. In general, R_{11} with the non-terminal S_1 is applied to the handle S. Then it uses R_{21} $n-2$ times and R_{22} once to generate the Caterpillar graph of order n. The HRL generated by the set of all Caterpillar graphs is $L(HRGRPS) = \{a^{n-1}b^{3n} | n \geq 2\}$.

Generation of the set of all Comb graphs:

For generating the set of all Comb graphs, the rule R_{12} with non-terminal S_2 is applied to the (1,1) handle S in the first membrane. The resultant graph produced enters into the second membrane. In membrane two, it can apply either R_{23} or R_{24}. If it applies R_{23} with the help of the attachment nodes (as explained in the Theorem 3.1), the Comb graph of order two is generated. In general, R_{12} with the non-terminal S_2 is applied to the handle S. Then it uses R_{23} $n-2$ times and R_{24} once to generate the Comb graph of order n. The HRL generated by set of all Comb graphs is $L(HRGRPS) = \{a^{n-1}b^n | n \geq 2\}$.

Generation of the set of all Fire Cracker graphs:

For generating the set of all Fire Cracker graphs, the rule R_{13} with non-terminal S_3 is applied to the (1,1) handle S in the first membrane. The resultant graph produced enters into the second membrane. In membrane two, it can apply either R_{25} or R_{26}. If it applies R_{25} with the help of the attachment nodes (as explained in the Theorem 3.1), the Fire Cracker graph of order two is generated. In general, R_{13} with the non-terminal S_3 is applied to the handle S. Then it uses R_{25} $n-2$ times and R_{26} once to generate the Fire Cracker graph of order n. The HRL generated by set of all Fire Cracker graphs is $L(HRGRPS) = \{a^{n-1}b^n c^{5n} | n \geq 2\}$.

□

Theorem 3.3. *The set of all Wheel graphs can be generated by hyper-edge replacement graph rewriting P system using two membranes with rules of order 2.*
The HRGRPS is a construct

$$\Pi_W = (\{1,2,3,4\}, \{S_1, S_2, a, b\}, \{a, b\}, [_1[_2]_2]_1], R_1, R_2, (2,1), 1).$$

Generally, R_1 denotes the set of production rules in the first membrane and R_2 denotes the set of production rules in the second membrane. Here, R_1 consists of three rules R_{11}, R_{12}, and R_{13} and R_2 consist of one rule R_{21}.

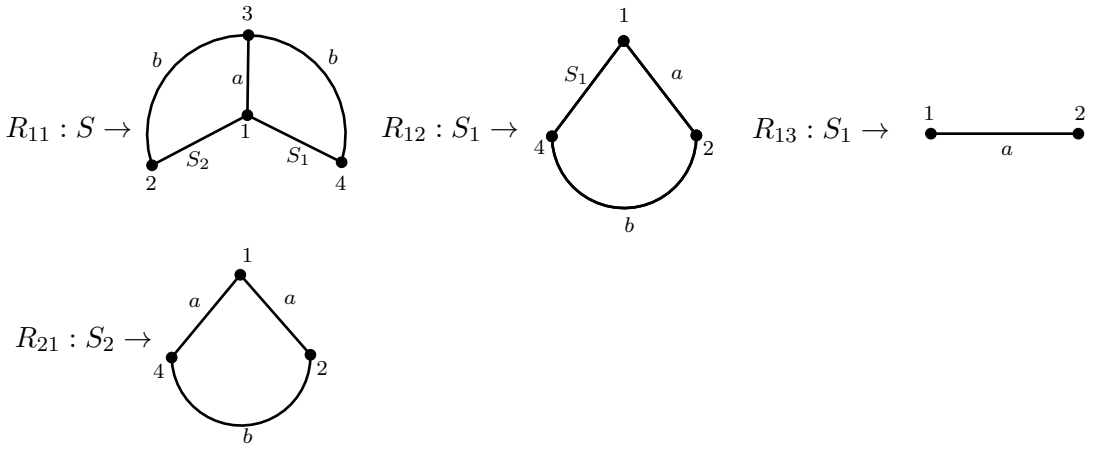

Fig 3.3.1: The production rules for generating Wheel graphs

Attachment nodes for R_{12}: $((1,H),(2,T_2),here)$, R_{13}: $((1,H),(2,T_2),in_2)$ and R_{21}: $((1,H),(2,T_2),(4,T_1),out)$.

Proof. Initially, the node labeled as 1 in R_{11} is considered as the head node(H), while node 2 in R_{11} (the node associated to the left of 1) is the main tail node, marked by T_1,

and node 4 in R_{11} is taken to be the second tail node, denoted by T_2. After using R_{12} in membrane one, T_2 increases by 1, but the head and tail node (T_1) stays constant.

The first membrane consists of a (1,1) handle labeled S. It uses R_{11} to create a graph with non-terminals S_1 and S_2. Since the skin membrane contains two additional rules, it has a choice in applying rules. If R_{13} is used, the non-terminal S_1 terminates there and forms a graph with the non-terminal S_2, before entering the second membrane. S_2 is terminated by R_{21} in the second membrane, which then produces the Wheel graph of order 3. For generating the Wheel graph of order 4, use R_{11} to generate a graph with non-terminals S_1 and S_2. It utilizes R_{12} and stays there, then it enters membrane two by applying R_{13}. After using the terminal rule R_{21} in membrane two, the resultant graph is sent out.

In general, the n^{th} order Wheel graph is generated by utilizing R_{11} to produce a graph with non-terminals S_1 and S_2. The rule R_{12} is applied $n-3$ times and R_{13} is applied once to build a graph with S_2. Then by entering second membrane it utilizes R_{21} once and the final output is collected in the skin membrane. The HRL generated by this system is $\{a^n b^n | n \geq 3\}$. □

4. Conclusion and Future work

The generation of the set of all Snail and Wheel graphs is accomplished in this proposed work with the smallest possible order. In addition, some prominent graphs like Caterpillar graphs, Comb graphs, and Fire Cracker graphs are generated in a single construct, which greatly aid in studying the topological features of benzenoid hydrocarbons, particularly resonance interactions among individual hexagons of a benzenoid system. In the future, we plan to generate some prominent classes of Wheel and Cycle graphs using this P system.

References

[1] Drewes, F., Kreowski, H. J. and Habel, A., (1997), Handbook of graph grammars and computing by graph transformation, World Scientific Publishing, pp. 95-162.
[2] Habel, A., (1992), Introduction to hyperedge-replacement grammars, Hyperedge Replacement: Grammars and Languages, pp. 5-42.
[3] Paun, G., (2010), Membrane computing, Scholarpedia, 5(1).
[4] Paun, G. and Rozenberg, G., (2002), A guide to membrane computing, Theoretical Computer Science, 287(1), pp. 73-100.
[5] Habel, A., (1992), Hyperedge replacement: grammars and languages, Springer Science and Business Media, Vol(643).
[6] Sankar, M. P., David, N. G. and Thomas, D. G., (2011), Hyperedge replacement graph P system, Sixth International Conference on Bio-Inspired Computing: Theories and Applications, IEEE, pp. 272-277.
[7] Sankar, M. P. and David, N. G., (2020), On Generative Power of Rewriting Graph P System, Artificial Intelligence and Evolutionary Computations in Engineering Systems, Springer, Singapore, pp. 419-429.
[8] Gallian, J. A., (2012), Graph labeling, The electronic journal of combinatorics, DS6-Dec.
[9] Dhanalakshmi, S. and Parvathi, N., (2018), Mean square cordial labelling related to some acyclic graphs and its rough approximations, Journal of physics: Conference series, IOP Publishing, 1000(1).

Vinodhini K. completed her master's degree in 2019, Department of Mathematics, University of Madras, India, M.Phil. in 2021 at Presidency College, Chennai, India. Currently she is a Ph.D. student in the Department of Mathematics, SRM Institute of Science and Technology, Kattankulathur, India. She is working in graph grammars especially in hyper-edge replacement graph grammar.

Dr. Meena Parvathy Sankar completed her Ph.D. in 2020, Department of Mathematics, Madras Christian College, Chennai, India. She is currently working as an assistant professor in the Department of Mathematics, SRM Institute of Science and technology, Kattankulathur, India. Her area of research is graph grammars and membrane computing.