Results on number of word-representable graphs

B.Tech Project Report Phase-II

Ripudaman Singh, Vansh Rathore

(Roll No. 200101090, 200101102)

under the guidance of

Benny George K.

Department of Computer Science and Engineering

Indian Institute of Technology Guwahati

January 2024 - April 2024

CERTIFICATE

This is to certify that the work contained in this project report entitled "Results on number of word-representable graphs" submitted by **Ripudaman Singh, Vansh Rathore** (Roll No.: 200101090, 200101102) to the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati towards partial requirement of Bachelor of Technology in Computer Science and Engineering has been carried out by him/her under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

Guwahati - 781 039	(Prof. Benny George K.)
April 2024	Project Supervisor

ABSTRACT

This is a report/roadmap to the BTech project on exploring the problem of determining whether a given graph is word-representable. By the end of Phase-I of this Btech Project, we were familiar with the graph isomorphism problem and got acquainted with word-representable graphs. In phase-II we sought to develop an algorithm which determines whether a graph is word-representable. Since the problem is NP-complete, we worked with up to 8-vertex graphs and observed the results on all non-isomorphic connected graphs. This algorithm serves as a tool which takes input as adjacency matrix of a graph and outputs whether it is word representable.

Contents

0.1 Introduction

Phase-II of the project builds upon the foundational concepts introduced in Phase-I, where we explored graph isomorphism using computational group theory. In this phase, we delve deeper into the study of graph structures and their representations, particularly focusing on word representable graphs and their connection to semi-transitivity.

0.1.1 Phase-I Recap

In Phase-I, we explored fundamental concepts in group theory such as Lagrange's Theorem and the Orbit-Stabilizer Theorem. These theorems provided essential groundwork for understanding group actions and their applications in graph theory. We delved into the Graph Isomorphism Problem, discussing its significance and challenges. We introduced the concept of automorphism groups and their relevance in identifying structural properties of graphs. Also, we examined the connection between the Graph Isomorphism Problem and the Graph Automorphism Problem, highlighting their computational complexity. Overall, Phase-I laid the theoretical foundation necessary for our subsequent exploration of word-representable graphs and their applications.

0.1.2 Word Representable Graphs

Word representable graphs are a class of graphs that can be represented by words over a finite alphabet, where each letter in the word corresponds to a vertex in the graph. A graph G = (V, E) is word-representable if there exists a word W over the alphabet V such that for each pair of distinct letters x and y, (x,y) $\in$ E if and only if the occurrences of the letters alternate in

W. As an example the words *abcdabcd*, *abcddcba*, and *abdacdbc* represent the 4-clique, K4; 4- independent set, K4; and the 4-cycle, C4, labeled by a, b, c, d in clockwise direction, respectively.

0.1.3 Semi-Transitivity and Word Representability

One of the key insights of Phase Two is the connection between word representable graphs and semi-transitivity. A graph is semi-transitive if it satisfies a partial transitivity condition, which can be expressed in terms of directed paths in the graph.

Let G = (V, E) be a graph. An acyclic orientation of G is **semi-transitive** if for any directed path v1 $\Rightarrow$ v2 $\Rightarrow \cdots \Rightarrow$ vk either

- there is no arc v1 $\Rightarrow$ vk, or
- the arc v1 $\Rightarrow$ vk is present and there are arcs vi $\Rightarrow$ vj for all 1 $\leq$ i $<$ j $\leq$ k. That is, in this case, the (acyclic) subgraph induced by the vertices v1, . . . , vk is a transitive clique (with the unique source v1 and the unique sink vk).

We aim to establish criteria for determining whether a graph is word representable based on its semi-transitive properties. This involves exploring the implications of semi-transitivity on the structural properties of graphs and devising algorithms to identify word representable graphs.

0.2 Methodology

The methodology adopted in Phase-II involves a combination of theoretical analysis and computational experiments. We first develop theoretical frameworks for characterizing word representable graphs and studying their

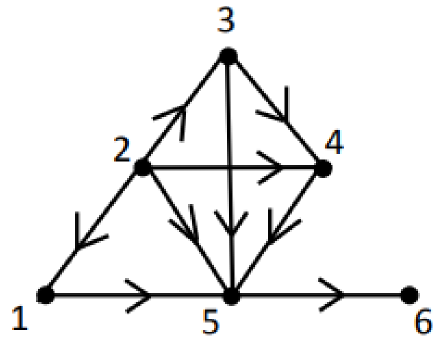

Figure 1: Example of a semi-transitive orientation

relationship with semi-transitive digraphs. Subsequently, we implement algorithms to test the word representability of graphs and validate our findings through computational experiments on a diverse set of graph instances.

0.2.1 Characterizing Word-Representable Graphs in Terms of Orientability

In this section, we present a characterization implying that word representability corresponds to a property of the graph obtained by directing the edges in the way described above so as to satisfy the semi-transitive nature of the orientation of the graph. We build our algorithm on the Theorem proved in [1] which states that:

A graph $G = (V, E)$ is semi-transitive if it admits a semi-transitive orientation.

We can alternatively define semi-transitive orientations in terms of induced subgraphs. A semi-cycle is the directed acyclic graph obtained by reversing the direction of one arc of a directed cycle. An acyclic digraph is a shortcut if it is induced by the vertices of a semi-cycle and contains a pair of non-adjacent vertices. Thus, a digraph on the vertex set $\{v_1, \ldots, v_k\}$ is a shortcut if it contains a directed path $v_1 \to v_2 \to \ldots \to v_k$, the arc $v_1 \to v_k$, and it is missing an arc $v_i \to v_j$ for some $1 \leq i < j \leq k$.

An orientation of a graph is semi-transitive if it is acyclic and contains no shortcuts.

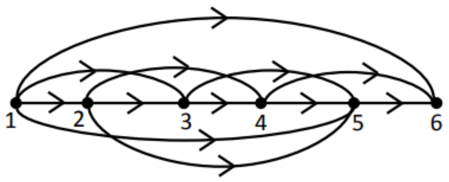

Figure 2: Example of a shortcut

0.2.2 Developing the algorithm

We decided to exploit the semi-transitive property of digraphs in order to devise an algorithm to check the word-representability of a given graph. Since this problem is NP-complete, our primary focus was not on the time complexity but on the verifiable nature of the results obtained from this algorithm. All the code and relevant files can be found in [4].

When we run the code written using C++ programming language, it prompts the user to enter the order of the graph or in other words, the number of vertices of the graph. Based on the order, it reads the file of given order and produces the result. The file contains an exhaustive list of adjacency matrices of all the non-isomorphic connected graphs of that particular order. For all practical reasons, the maximum number of vertices up to which this algorithm is tested is 8. The algorithm can be described as follows:

- **Input:** Order of the graph

- **Working:** Reads the appropriate file and for each adjacency matrix, it iterates over all the possible orientations of the graph for any semi-transitive orientation

- **Time Complexity:** O(2^{n^2}) where n is the number of vertices

- **Output:** Generates an output file that contains the answer for each of the given adjacency matrices. Returns YES if it finds at least one semi-transitive orientation and NO if it does not.

0.3 Results and Analysis

The results we obtained with our algorithm and their correctness can be verified from [3]. Here are the parallels drawn from the existing literature and our algorithm:

- There are no non word-representable graphs for graphs with 5 or less vertices and our code verifies the same.

- There is only one non word-representable graph on 6 vertices, which is the **wheel graph** W_5 and our code verifies the same.

- There are 25 non word-representable graphs on 7 vertices and our code verifies the same.

- Our code correctly identified 929 non word-representable graphs on 8 vertices.

The graphs taken as input are all non-isomorphic and connected. The code was run on an exhaustive list of 112 graphs on 6 vertices, 853 graphs on 7 vertices and 11117 graphs on 8 vertces, all of which can be verified from [2].

0.3.1 Computational Complexity

The computational complexity of our algorithms varies depending on the size and structure of the input graph. Let V be the number of vertices and E be the number of edges in the graph.

- **Graph Orientation**: Iterating over all the possible orientations of a given graph, which can be $\mathrm{O}(2^{V^2})$ in the worst case, making this the costliest computation.
- **Word Representation**: Generating words to represent graphs involves iterating over vertices and encoding adjacency information, resulting in a complexity of $O(V^2)$ in the worst case.
- **Semi-Transitivity Detection**: Detecting semi-transitivity involves analyzing directed paths and checking conditions, resulting in a complexity of $O(V^3)$ in the worst case.

0.4 Conclusion

Now we have a tool on our hands that can check the word-representability of a graph, given it has 8 or less vertices. The property of word-representabilty encompasses several important classes of graphs like 3-colorable graphs, cover graphs and comparability graphs which can now be studied with the help of this tool.

Given the problem is NP-complete, this is good progress and future optimizations can be done to expand this tool to check graphs with 9 or more vertices. We can also generate the words that represent these graphs as part of future work.

Acknowledgment

We would like to thank our project advisor Dr. Benny George K. and the Department of Computer Science and Engineering at the Indian Institute of Technology Guwahati for their support and resources throughout the project.

Bibliography

[1] Halldórsson, M. M., Kitaev, S., Pyatkin, A. (2015). Semi-Transitive Orientations and Word-Representable Graphs. ArXiv. /abs/1501.07108

[2] http://users.cecs.anu.edu.au/ bdm/data/graphs.html

[3] Sergey Kitaev and Vadim Lozin, "Words and Graphs", Springer, 2015.

[4] https://github.com/v4n5hgit/Word-Representability-Checker.git

[5] https://youtube.com/playlist?list=PLTtM9ThZ2L-dY4Gc$_eOn3oNYwdILeYnn3si = GgYEjmyhhZR4l1aQ$