ON THE ENUMERATION OF A CLASS OF NON-GRACEFUL GRAPHS

ADAM DRAKE* AND TIMOTHY A. REDL†

UNIVERSITY OF HOUSTON-DOWNTOWN
HOUSTON, TEXAS

Abstract. A simple graph G is a graceful graph if there exists a graceful labeling of the vertices of G. If we cannot gracefully label the vertices of G, then G is a non-graceful graph. A result by Rosa provides a sufficient condition for a graph to be non-graceful: "If a graph G is simple, even, and has e edges, with $e \equiv 1$ or 2 (mod 4), then G is not graceful." This condition implies an infinite subclass of non-graceful graphs, which we define to be $\mathcal{R}$. By the degree-sum formula for graphs, the sum of the degrees of a graph G is equal to $2e$. We systematically enumerate graphs in $\mathcal{R}$ by first generating all even partitions of $2e$ (where $e \equiv 1$ or 2 (mod 4)) using Maple. We then use the Havel-Hakimi procedure to determine which of these generated "number sequences" are graphic sequences. These graphic sequences determine all of the simple, even, graphs in $\mathcal{R}$ (both connected and disconnected) with $e \equiv 1$ or 2 (mod 4) edges.
Key words: graceful labeling, graceful graph, non-graceful graph, graphic sequence

1. Introduction

Given a graph G, a *vertex labeling* of G is an assignment f of labels to the vertices of G that produces for each edge xy a label depending on the vertex labels $f(x)$ and $f(y)$. A vertex labeling f is called a *graceful labeling* of a graph G with e edges if f is an injection from the vertices of G to the set $\{0, 1, \ldots, e\}$ such that when each edge xy is assigned the label $|f(x) - f(y)|$ the resulting edge labels are distinct. A graph G is *graceful* if a graceful labeling of G exists. If no graceful labeling of G exists then G is called *non-graceful*. Figure 1 illustrates an example of a graceful graph (C_4) and a non-graceful graph (C_5).

* adrake@gmail.com, † redlt@uhd.edu
The authors thank the University of Houston-Downtown Scholars Academy and the United States Army Research Office for partially funding this research (ARO grant W911NF-1-0024).

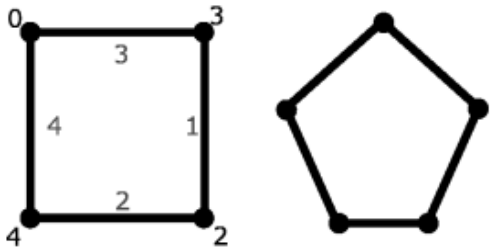

FIGURE 1. C_4 is graceful; C_5 is non-graceful

The study of graph labelings, specifically graceful ones, was introduced by Rosa [8] in 1967. Rosa defined a β-valuation of a graph G with e edges as an injection from the vertices of G to the set $\{0, 1, \ldots, e\}$ such that when each edge xy is assigned the label $|f(x) - f(y)|$ the resulting edge labels are distinct. Rosa generated the framework for these β-valuations; the term *graceful labeling* was not used until Golomb [3] studied such labelings in 1972.

The graceful labeling problem, specifically, is given a graph G, determine whether or not G is graceful. In order to prove that G is graceful one only need show that a graceful labeling of G exists. The other case is significantly more difficult. In order to prove that G is not graceful, it must be shown that no graceful labeling of G exists. Over the past 40 years, hundreds of papers on graceful labelings have been published. Certain classes of graphs have been proven to be graceful. For example, Rosa [8] proved in 1967 that all paths P_n and all cycles C_n (where $n \equiv 0$ or 3 (mod 4)) are graceful. In addition, complete graphs K_n are graceful if and only if $n \leq 4$ as proven by Golomb [3], and all wheels $W_n = C_{n-1} + K_n$ were proven graceful by Frucht [1] in 1979. It is important to note that a subgraph of a graceful graph need not be graceful. While Frucht's result proves that W_6 is graceful, its subgraph C_5 is not. Perhaps the most famous open problem in graceful graph labeling is the Ringel-Kotzig conjecture that all trees are graceful. While much research has been done on graceful graphs, an unpublished result of Erdös that was later proven by Graham and Sloane [4] states that almost all graphs are not graceful. Even though the majority of graphs have no graceful labeling, there have been comparatively few studies on the properties of non-graceful graphs.

2. Rosa's parity condition: our motivation

Rosa identified three basic reasons why a graph G might fail to be graceful: (1) G has too many vertices and not enough edges; (2) G has too many edges; or (3) G has the wrong parity [2]. As an example of (3), Rosa developed a useful parity condition for a simple graph G with e edges. He proved

that if every vertex of G has even degree and $e \equiv 1$ or 2 (mod 4), then G is not graceful. Rosa's parity condition serves as a sufficient condition for non-graceful graphs. We define $\mathcal{R}$ to be the infinite class of graphs that satisfy Rosa's parity condition, and are hence non-graceful.

Our goal is to systematically enumerate graphs in $\mathcal{R}$ in order to further examine the structure of graphs in $\mathcal{R}$ and attempt to determine additional properties common to graphs in the class.

3. Computational Enumeration

The process for enumerating all non-graceful graphs in $\mathcal{R}$ requires simple arithmetic and repetition. It would take an unreasonable amount of time for a human to accomplish this task. A natural solution is to employ computers to deal with the repetitive arithmetic. Given some value of $e \equiv 1$ or 2 (mod 4), we construct a process by which all possible degree sequences of a graph with e edges are generated by first constructing all partitions of $2e$ consisting of only even integers, and then evaluating the sequences one at a time using the Havel-Hakimi procedure [5] to determine which of these degree sequences are graphic. A *graphic sequence* is a list of non-negative numbers that is the degree sequence of some simple graph. The result of this process is a list of sequences that determine graphs in $\mathcal{R}$ (both connected and disconnected) with e edges, where $e \equiv 1$ or 2 (mod 4).

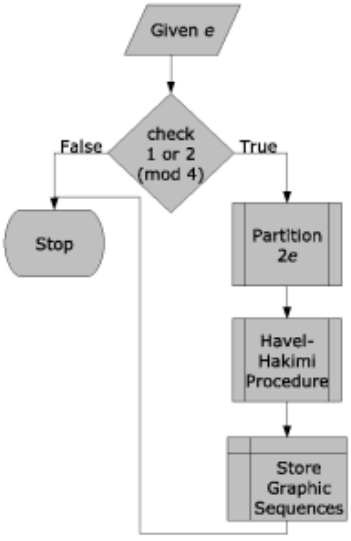

Figure 2. An overview of the enumeration procedure

A flowchart of our procedure can be seen in Figure 2. The first step in the procedure is to input a value of $e \equiv 1$ or 2 (mod 4) to Maple for partitioning. Maple generates all partitions of $2e$ consisting of only even integers. These partitions (sequences) are stored in a file where they are then passed to the Havel-Hakimi module. This module examines each sequence and uses the Havel-Hakimi procedure to determine whether it is graphic or not. Output of the Havel-Hakimi module consists of statistics on sequences for a particular value of e including number of sequences, number of graphic sequences, processing time, resource usage, and other information.

3.1. **Step 1: Partitioning.** The *degree* of a vertex v is the number of edges adjacent to v. The *degree sequence* of a graph G with n vertices is a sequence of length n whose elements are the degrees of the vertices of G. It is important to note that by the definition of degree sequence, each graph has a unique degree sequence when degrees are listed in non-increasing order. Also, two or more non-isomorphic graphs may have the same degree sequence. By the degree-sum formula for graphs, the sum of the elements in a degree sequence of a graph with e edges is $2e$. Because only even graphs satisfy Rosa's parity condition, we partition $2e$ such that every element of the degree sequence is an even number. To generate the even partitions of $2e$ we first use the partition function in Maple 10 to generate all partitions of $e \equiv 1$ or 2 (mod 4). Initially these partitions of e do not consist only of even numbers; they are a combination of odd and even numbers. We multiply all elements of each partition by 2, which yields partitions of $2e$ consisting of only even numbers. These partitions correspond precisely to the degree sequences of all even graphs in $\mathcal{R}$ with e edges. We then need to determine which of these sequences are graphic.

3.2. **Step 2: Havel-Hakimi Procedure.** Given a degree sequence d, one can systematically determine whether or not that sequence is graphic (i.e., represents a simple graph) as justified by the following theorem:

Theorem 1. *(Havel-Hakimi)* [5] *For $n > 1$, the non-negative integer list d of size n is graphic iff d' is graphic, where d' is the list of size $n-1$ obtained from d by deleting its largest element Δ and subtracting 1 from its next Δ largest elements.*

This theorem lends itself to an iterative procedure (the Havel-Hakimi procedure) which can be used to determine whether a degree sequence is or is not graphic. A degree sequence is graphic if and only if its largest and smallest elements are zero after a finite number of iterations of the Havel-Hakimi procedure. Given a degree sequence d of length n, the original sequence is graphic if successive applications of the Havel-Hakimi theorem produce a final sequence consisting of only zeros.

We implemented the Havel-Hakimi procedure using the Ruby programming language [9]. Ruby is a purely object-oriented scripting language developed in Japan by Yukihiro "Matz" Matsumoto. Ruby is an excellent language in almost all situations, especially those where rapid prototyping is needed. Using the power of Ruby we were able to write the code for the Havel-Hakimi procedure, including comments and diagnostic features, quickly and concisely. In addition, because Ruby is portable and supports operating system-independent threading, we can scale the code from a single CPU machine to multiple CPU machines or even to a distributed computing environment without major modification.

The output of the Havel-Hakimi module is a list of graphic sequences that determine graphs in $\mathcal{R}$ (both connected and disconnected) with e edges, where $e \equiv 1$ or $2 \pmod 4$.

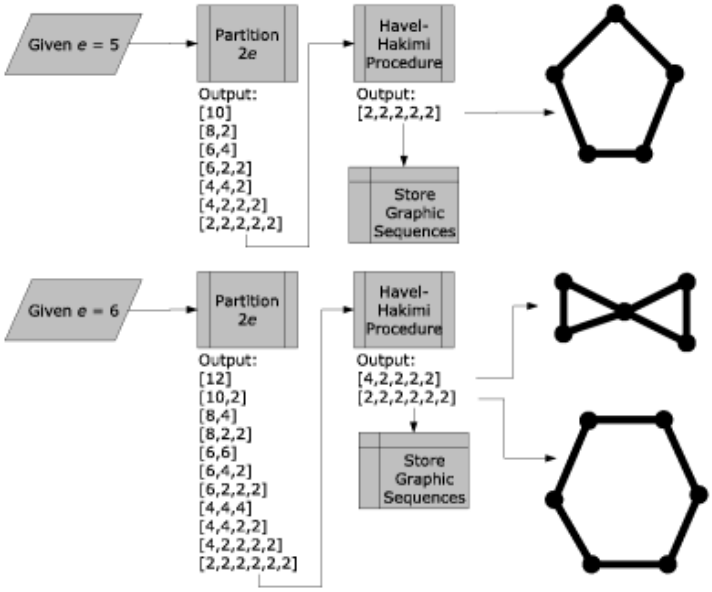

FIGURE 3. Examples of the procedure for e=5 and e=6

4. EXAMPLES (e=5 AND e=6)

Figure 3 illustrates a step-by-step example of our procedure for the cases e=5 and e=6. The partition step computes all 7 partitions (sequences) of $2e$ when e=5 and all 11 partitions (sequences) of $2e$ when e=6. These partitions are passed to the Havel-Hakimi module which examines the sequences and determines that for $e = 5$ there is only one graphic sequence from the

original set while for $e = 6$, there are two graphic sequences. These sequences determine the graphs shown to the right. It is important to note that the pictoral representation of the graphs is not unique.

5. Computational Results

As of this writing, we are able to successfully run our enumeration procedure and obtain results for values of e up to 62 on a commodity-hardware laptop. A list of results is shown in the table below. If more powerful hardware is used or if the procedure is adapted to a distributed computing environment then our ability to evaluate higher values of e will increase accordingly.

e	#Seq.	#Graphic	%Graphic	Part.	H-H	Total Time
5	7	1	14.28	0.00	0.00	0.00
6	11	2	18.18	0.00	0.00	0.00
9	30	5	16.66	0.00	0.00	0.00
10	42	8	19.04	0.00	0.00	0.00
13	101	16	15.84	0.00	0.00	0.00
14	135	22	16.29	0.00	0.00	0.00
17	297	45	15.15	0.00	0.01	0.01
18	385	61	15.84	0.00	0.02	0.02
21	792	118	14.89	0.00	0.05	0.05
22	1002	152	15.16	0.01	0.06	0.07
25	1958	279	14.24	0.02	0.14	0.16
26	2436	354	14.53	0.03	0.19	0.22
29	4565	627	13.73	0.12	0.41	0.53
30	5604	780	13.91	0.15	0.49	0.64
33	10143	1355	13.35	0.28	1.01	1.29
34	12310	1651	13.41	0.37	1.36	1.73
37	21637	2809	12.98	0.88	2.29	3.17
38	26015	3372	12.96	1.06	3.54	4.60
41	44583	5628	12.62	1.95	5.73	7.68
42	53174	6678	12.55	2.73	12.09	14.82
45	89134	10964	12.30	4.93	13.55	18.48
46	105558	12861	12.18	7.10	16.55	23.65
49	173525	20799	11.98	13.30	34.24	47.54
50	204226	24209	11.85	20.52	37.16	57.68
53	329931	38590	11.69	22.90	86.52	109.42
54	386115	44635	11.56	28.20	65.60	93.80
57	614154	70157	11.42	59.63	1011.78	1071.41
58	715220	80712	11.21	75.36	132.68	208.04
61	1121505	125173	11.16	161.33	260.31	421.64
62	1300156	143422	11.03	201.13	262.07	463.20

The table above provides results of our procedure and contains columns for edge count e, the total number of sequences, number of graphic sequences, percentage of sequences that are graphic, partitioning time (in seconds), Havel-Hakimi module time (in seconds), and total computing time (in seconds). We note that as e gets larger, the total number of sequences appears to grow exponentially. The number of graphic sequences also seems to grow exponentially though not as quickly. Figure 4 illustrates the growth rate of the number of total sequences and the number of graphic sequences with respect to increasing edge count e.

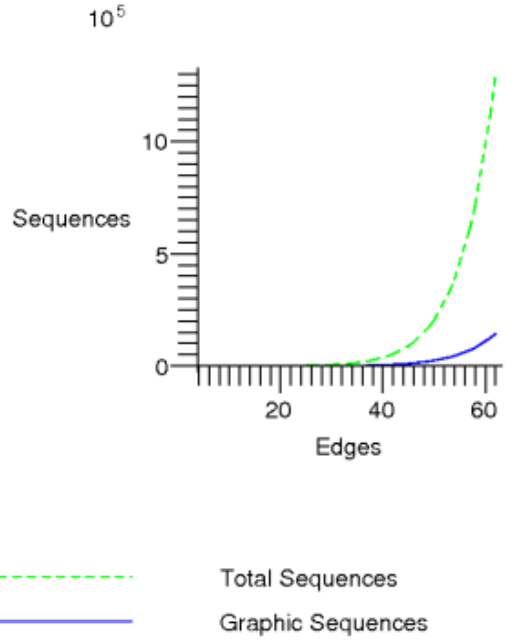

FIGURE 4. Number of Total and Graphic Sequences vs. Edges

The "% Graphic" column in our table of results raises an interesting question. As $e \to \infty$ does the percentage of graphic sequences ultimately decrease to zero? Figure 5 illustrates the ratio of the number of graphic sequences to the number of total sequences with respect to edge count e. As e increases, the percentage of graphic sequences decreases. This percentage could ultimately decrease to zero, or stabilize after some significantly large value of e. As results for larger values of e are obtained, we hope to resolve this question.

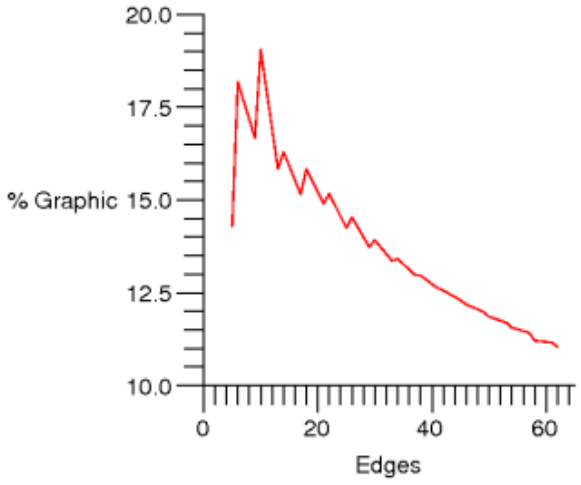

FIGURE 5. Percentage of Graphic Sequences vs. Edges

6. Conclusions and Future Research

We have constructed an enumeration procedure to generate non-graceful graphs satisfying Rosa's parity condition. However, our work on this procedure is not complete. We have many ideas regarding how the procedure can be improved. As shown in our table, processing requirements become greater as e gets larger. It is probable that more results can be obtained with the use of more efficient partitioning algorithms such as Knuth's partitioning algorithm P [6]. In addition to speeding up the partitioning step, we are interested in the effects of transitioning the Havel-Hakimi procedure from the Ruby programming language, which is an interpreted language, to a compiled language such as C or C++. This two-pronged approach of algorithm modification and language transition has the potential to greatly increase the speed of our procedure for large values of e.

Another modification that has potential for more results is to implement the procedure in a distributed computing environment. In this arrangement, slave nodes would fetch jobs from a master node. The partitioning step would remain largely the same, with the exception of a possible algorithm change. The Havel-Hakimi step would be quite different though, because instead of one slave node processing all the partitions for a given value of e, the total set of partitions would be broken into subsets of 100,000 partitions. All the partitions and subsequent data on total sequences, graphic

sequences, percentage of graphic sequence, etc. would be stored in a database. In addition to being conducive to distributed computing, the database would allow us to store information more effectively and make analysis and reporting much easier.

In addition to the main topics of this paper, our research led to an interesting related topic concerning graceful graph labelings. We define a *disgraced graph* to be a graceful graph that has been made non-graceful. As an extension of (1) in section 2, one way to disgrace a graceful graph is to add a sufficient number of isolated vertices to it. We have the following proposition.

Proposition 1. *Any graceful graph G with n vertices and e edges can be disgraced by adding a number of isolated vertices i such that $i > e - n + 1$.*

Proof. Consider a graceful graph G with n vertices and e edges. There are $e+1$ possible labels $\{0, 1, \ldots, e\}$ for the vertices of G. In a graceful labeling of G, exactly n of these $e + 1$ possible labels will be used, one for each vertex, leaving $e - n + 1$ labels unused. If we add i isolated vertices to G, where $i > e - n + 1$ we will have disgraced G, as it will now be impossible to gracefully label G; we do not have enough labels. □

As an extension of (2) in section 2, another way to disgrace a graceful graph if $n > 4$ is to "complete it." We have the following proposition.

Proposition 2. *Any graceful graph G with $n > 4$ vertices can be disgraced by adding enough edges to make G a complete graph K_n.*

Proof. All complete graphs K_n where $n > 4$ are non-graceful [3]. □

These are just two examples of ways to disgrace a graceful graph. What other ways might there be to disgrace a graceful graph? We are actively pursuing our research on this topic.

We look forward to continuing our work in the area of graph labelings, including developing the previously mentioned modifications to our procedure, and ultimately making further contributions to the field of graph labeling.

References

[1] Frucht, R. *Graceful Labeling of Wheels and Other Related Graphs*, **Annals New York Academy of Sciences** 319 (1979), 219-229

[2] Gallian, J. A. *A Dynamic Survey of Graph Labeling*, **The Electronic Journal of Combinatorics** 1-95.

[3] Golomb, S. W. *How to Number a Graph*, **Graph Theory and Computing**, R.C. Read, ed., Academic Press: 1972, 23-37

[4] Graham, R. L. and Sloane, N. J. A. *On Additive Bases and Harmonious Graphs*, **SIAM Journal on Algebraic and Discrete Methods 1 (1980)** 382-404.

[5] Hakimi, S. L. *On the Realizability of a Set of Intgers As Degrees of the Vertices of a Graph*, **SIAM J. Appl Math 10 (1962)**, 496-506

[6] Knuth, D. E. **The Art of Computer Programming, Vol. 4, Fascicle 3**, Addison-Wesley Profrssional: 2005

[7] Redl, T. *Graceful Graphs and Graceful Labelings: Two Mathematical Programming Formulations and Some Other New Results*, **Congressus Numerantium 164 (2003)**, 17-31

[8] Rosa, A. *On Certain Valuations of the Vertices of a Graph*, **Theory of Graphs, International Symposium**, Rome, July 1966, Gordon and Breach: 1967, 349-355

[9] Ruby Programming Language: http://www.ruby-lang.org