Geometry of Horospheres in Graphs

Noah Jillson †, Katerina Stuopis †, Kaicheng Xue †, Qianruixi Wang †, Daniel Levitin †, Tullia Dymarz †

†University of Wisconsin - Madison

Introduction

This project aimed to understand horospheres in hyperbolic graphs. We hoped to find some algorithms to solve geometric problems in Cayley Graphs. We started with the problem of path-finding in an order-4 pentagonal tiling. We developed and implemented general algorithms to calculate distances between points.

Background

We began our study by making sense of the symmetries present within the order-4 pentagonal tiling. In particular, we focused on the five lines of symmetries that intersect to form the pentagon at the center of the tiling pictured below.

Consider an arbitrary symmetry and tile denoted σ_i and t respectively. For every tile t, there exists a corresponding tile t' such that $\sigma_i(t)$ maps t to t'. And upon the application of σ_i, t' will also land on t. Thus, all five symmetries are in fact bijective where σ_i is its own inverse. Since this project is concerned with repeated applications of these symmetries, we can apply our knowledge of graphs to the order-4 tiling by considering its dual graph, the order-5 pentagonal tiling.

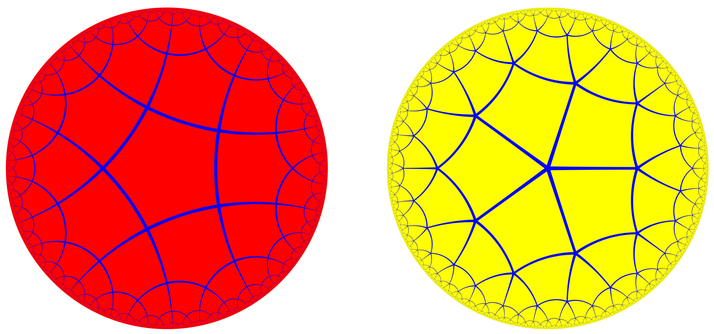

Fig. 1: {4,5} Hyperbolic Tiling (Order-4 Pentagonal Tiling)[3] (left) and Its Dual Graph[4] (right)

Since each vertex in our dual graph corresponds to a pentagonal face in our tiling, we adopt a naming convention for our dual graph vertices. By labeling the lines of reflection A through E, each vertex on the dual graph can be thought of as repeated applications of these reflections upon the central tile. Thus, the we call the center vertex the "empty string" or ϵ, and all other vertices a lowercase string of reflections in order from last to first applied. This naming scheme allows for the left-most letters of the name to communicate in which quintant of the tiling the vertex lies, and each subsequent letter describes the positional sub-quintants. When applying a reflection, we pre-pend the reflection's label to all vertex names in the dual graph. This allows us to locate every vertex pre and post reflection. It is important to note that letters on the right of a vertex name relates to edge adjacency and letters on the left relate to reflectional "near-ness". That is, acb and ac are edge connected, and ecb can be reached through the application of E to cb.

Lastly, within the hyperbolic tiling the Busemann function is defined as follows

$$B_\gamma(w) = \lim_{n\to\infty} (d(\gamma(n), w) - d(\gamma(n), 0))$$

where w is a vertex name, or "word", d is the distance metric, and $\gamma(n)$ is a repeated application of reflections n times. With this understanding, we can finally describe the focus of our research—horospheres. A horosphere is characterized by the level set of a Busemann function. Specifically, we defined $\gamma(n) = (ac)^n$. To evaluate our Busemann function, we implemented an algorithm as follows: Start by concatenating w_1 with the inverse of w_2 (just the reverse order of reflections); Then we simplify the new word as much as possible given an order to our alphabet resulting in w'; The length of w' tells us how far apart w_1 and w_2 are.

Results

To improve time complexity in our horosphere generation algorithm, we utilized a finite state machine (FSM) to pre-process each vertex name. The FSM pictured below is generated such that every path through the machine results in a perfectly ordered vertex name. This allows us to escape edge cases in horosphere generation where characters in the vertex name commute forward. For example $aceba$ is on the horosphere of -1 not 0 as $aceba \equiv acaeb$, and three letters will cancel and only two are added.

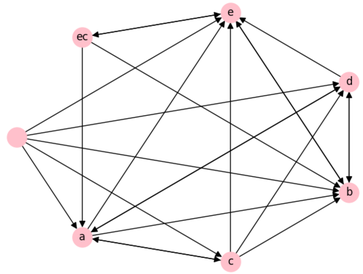

Fig. 2: Word Acceptor Given Ordering $\{a, c, b, d, e\}$

Implementing the FSM as a jump table, we can construct all ordered strings of a given length that do not start with a or c. Then, pre-pending an alternating seqence of a and c to our constructed word w will result in a vertex on the horosphere when the a, c sequence is the same length as w. Lastly, we were able to connect all vertices on the horosphere that were distance two apart.

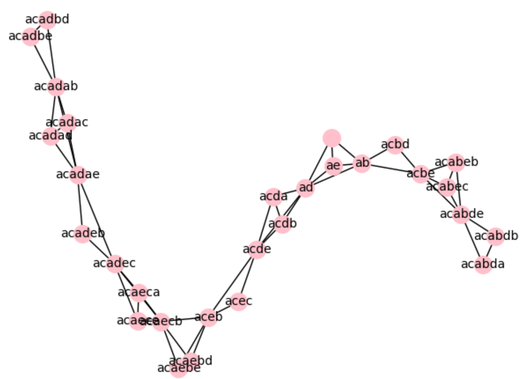

Fig. 3: Order-4 Pentagonal Tiling Horosphere Piece $(depth = 3)$ $(\gamma(n) = (ac)^n)$

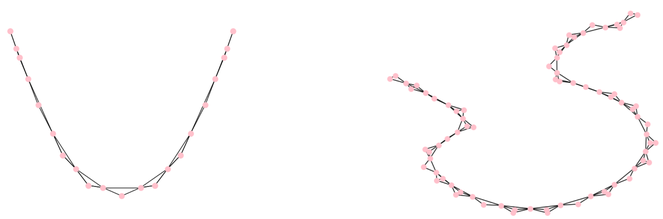

Fig. 4: Horospheres using NetworkX[1] spectral layout (left) and spring layout (right)

Discussion

By comparing the FSMs for the pentagonal tiling over the ordering $\{a, b, c, d, e\}$ and $\{a, c, b, d, e\}$, we can see that certain alphabets are advantageous. Using our standard alphabetical order we end up with nine states for our finite state machine, while having c before b in the alphabet results in only seven states. We also have an algorithm that accurately creates FSMs for multiple graphs. Note below the FSM for a Square Tiling.

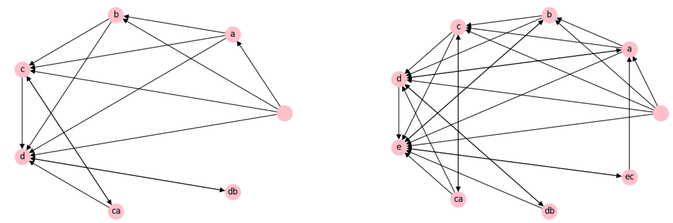

Fig. 5: Word Acceptors Given Orderings $\{a, b, c, d\}$ (left) and $\{a, b, c, d, e\}$ (right)

Future Directions

Moving forward, we would like to generalize our finite state machine to a heptagonal flag triangulation of a torus with no a squares, as shown below. This graph has 21 nodes each with degree six, so this comes with challenges in making a FSM. We would also like to work on our pictures of horospheres, so we are experimenting with different algorithms, to find points along the horosphere.

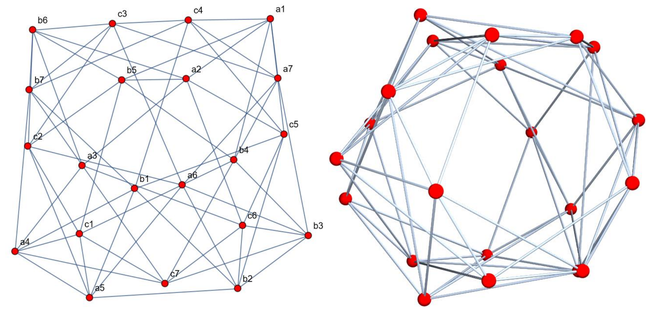

Fig. 6: Heptagonal Flag Triangulation of A Torus with No Squares Flat and 3D Renderings[2]

References

[1] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[2] Wolfram Research Inc. *Mathematica, Version 13.1*. Champaign, IL, 2022. URL: https://www.wolfram.com/mathematica.

[3] Parcly Taxel. *Order-4 Square Tiling*. URL: https://commons.wikimedia.org/wiki/User:Parcly_Taxel.

[4] Parcly Taxel. *Order-5 Pentagonal Tiling*. URL: https://commons.wikimedia.org/wiki/User:Parcly_Taxel.