Clustering edge-transitive tetravalent graphs

Lisi Qarkaxhija
Data Science student
University of Primorska - FAMNIT
Koper, Slovenia
lisqarkaxhija@gmail.com

***Abstract*—The goal of this project was two-fold. The first one goal was to scrape the data and prepare it in the specific format that was required by MathData Hub. The other goal was to explore tools for doing data science in Python. We used the K-means clustering on the dataset. We mastered the contemporary tools and we are ready for more serious challenges.**

***Index Terms*—clustering, K-means, graphs, census, Python**

I. Introduction

We would like to understand the behavior of graphs in the paper "Recipes for Edge-Transitive Tetravalent Graphs" by Steve Wilson and Primož Potočnik [1]. This paper accompanies the Census of Edge-Transitive Tetravalent Graphs, available at http://jan.ucc.nau.edu/~swilson/C4FullSite/BigTable.html, which is a collection of all known edge-transitive tetravalent graphs on up to 512 vertices. The data from the latter website was scraped, cleaned, manipulated and submitted to the MathData Hub project [2]. This project is accompanied by the paper "Towards a Unified Mathematical Data Infrastructure: Database and Interface Generation" by Katja Berčič, Michael Kohlhase, and Florian Rabe [2]. The main purpose of the MathData Hub project is to provide a scalable database for mathematical objects, and we managed to add more information about the specific type of graphs mentioned above.

Let us try to understand the graphs used in this census in more detail. A *graph* in this census is a pair $G = (V, E)$, where V is a set of vertices, and E is a set of two-sets of vertices, whose elements are called edges. If edges are ordered pairs, they are called *darts*. An *automorphism (symmetry)* of a graph G is a permutation of its vertices that preserves edges, i.e. a map $f\colon V(G) \to V(G)$ such that $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(G)$. For a graph G, the set of all automorphisms forms a group under composition, denoted $\mathrm{Aut}(G)$. A *vertex-transitive graph* is an undirected graph in which every vertex may be mapped by an automorphism to any other vertex. Similarly, the notion *edge (dart)-transitive graph* insinuates that every edge (dart) may be mapped to any other edge by an automorphism.(If a graph is dart-transitive then it is also vertex- and edge-transitive.) A graph is k-valent if every vertex has exactly k neighbors in the graph. If the graph is vertex and edge-transitive but not dart-transitive then it is called *half-arc-transitive*. If the graph is edge-transitive but not vertex-transitive then it is *semi-symmetric*.

The dataset that we scraped contains 3565 dart-transitive graphs, 920 half-arc-transitive graphs and 2879 semi-symmetric graphs for a total of 7364. The dataset includes 12 columns which are: tag (e.g. graph $C4[20, 5]$ is the 5-th in the list of graphs having 20 vertices), number of vertices, number of edges, type of symmetry the graph has, worthiness, bipartiteness, order of the symmetry group, order of a vertex stabilizer, order of a dart stabilizer, the number of semi-transitive orientations, girth, diameter and the canonical name of the graph.

II. Scraping and Data Manipulation

The data that needed to be scraped was an HTML table where some of the entries were hyperlinks. Jupyter Notebook and Python 3 along with its libraries were used for this task. *Pandas* [14] is a popular library which we used for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. *NumPy* [15] library helps with large, multi-dimensional arrays and matrices, and includes mathematical functions to operate on these arrays. Scraping webpages was made possible by the library *requests* [8]. To parse HTML and XML documents we used *Beautiful Soup* [4] and *lxml.html* [9]. A very useful tool for scraping are regular expressions [16], which we can utilize with the *re* library. Visualizations were made possible using the two famous libraries for python: *Matplotlib* [10] and *Plotly* [11]. The census also includes constructions of graphs in Magma [17] which we transformed to *graph6* [3] format with the help of *NetworkX* [12]. In order to upload all the scraped and manipulated data to MathData Hub [2] we needed to prepare a schema, data and provenance json files by using the *json* [13] library. The most important machine learning library used was *scikit-learn* [5], [6].

III. Clustering

Clustering is the process of finding clusters (i.e. subsets), in a dataset. The goal of clustering is to partition the dataset into subsets in such a way that observations within subsets are more similar with each other than compared to the observations in other subsets. We will cluster the census data and try to find out which graphs are more similar to one-another. This is an unsupervised problem because we are trying to determine the clusters based on the dataset. There are various algorithms for the task of clustering. The method that we tried out here is *K-means clustering* [7]. We chose K-means for a couple of reasons which are: it scales to large datasets, guarantees convergence, and is less expensive than other clustering algorithms.

To perform K-means algorithm we must specify the desired number of clusters, K. To determine the optimal number of clusters into which the data may be clustered, we use two heuristic methods: *Elbow Method* and *Silhouette Method* [6], [7].

The idea of elbow method is to run K-means clustering on the dataset for a range of values of K. For each value of K the sum of squared errors (SSE) is calculated. Then we plot the calculated SSE values (see Figure 1). The polygonal line should resemble an arm, and the “elbow” on the arm is the value that gives the best result. The best result is a small value of K that still has a low SSE.

The silhouette value measures the similarity of an object to objects within the same cluster compared to objects in the other clusters. The values lie in the interval $[-1, 1]$, where a low value indicates that the object is poorly assigned to its own cluster and better matched to some other clusters.

Visualization is a useful to make sense of the data. Principal component analysis (PCA) [7] is an important technique which we will use. PCA is an unsupervised feature extraction technique of dimensional reduction. By reducing the dimensions we have fewer relationships between variables to consider.

Initially we used the K-means algorithm on the original dataset. We noticed that some of the entries are too large in a few specific columns (namely, order of symmetry group, order of a vertex stabilizer, order of dart stabilizers), so we went back to the original dataset and we logarithmized those columns. After doing so, we again implemented K-means algorithm. We will show the results for both resulting datasets in the following subsections.

A. The Original dataset

In Table I we can see the first five rows of the data set. As described above, we are going to determine the number of clusters by using elbow and silhouette methods. The elbow method (see Figure 1) suggests that the number of clusters should be 2.

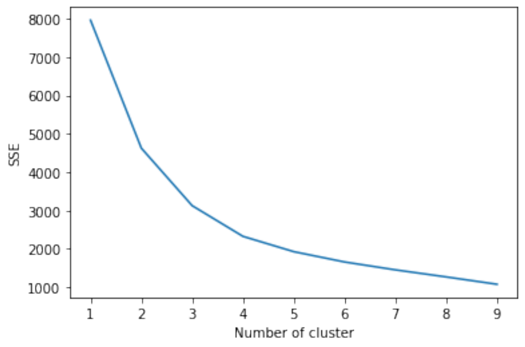

Fig. 1. The Elbow method.

Let us see what are the silhouette scores for $k = 2$ up to 4 and if both methods agree on the number of clusters. The scores are 0.4308, 0.5192 and 0.5191, respectively. Based on the reason that the closer the value is to 1 the more the objects of a cluster differ from objects of the other clusters and are more similar the the ones in their cluster, we deduce from this method that k should be 3.

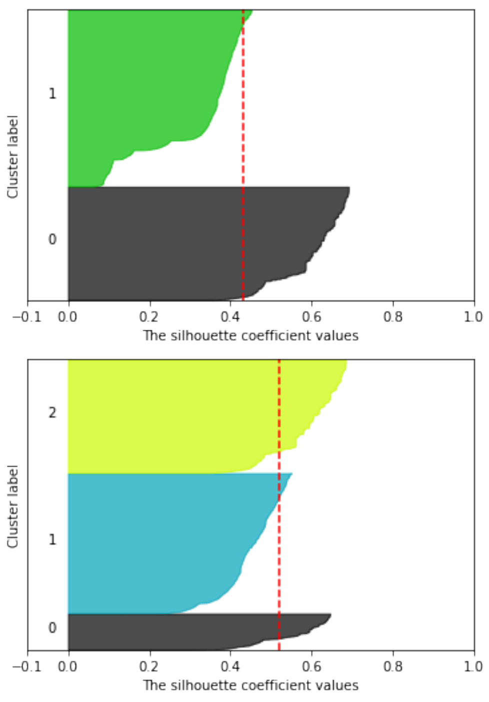

Fig. 2. The silhouette plot for $K = 2$ and 3.

In Figure 2 we can also see the visual representation of the scores. After observing the results given by both methods we decided that we will cluster the dataset for $K = 2$ and 3. For each $K = 2$ we got 4485 graphs in the first cluster and 2879 graphs in the other one. Now we use PCA to reduce dimensionality in order to plot the results in 3 dimensions.

TABLE I
ORIGINAL DATASET

	no. vertices	no. edges	worthy	bipartite	order of symmetry group	order of a vertex stabilizer	order of dart stabilizer	no. of semi-transitive orientations	girth	diameter	dart-transitive	semi-symmetric	half-arc-transitive
0	5	10	1	0	120	24	6	0	3	1	1	0	0
1	6	12	0	0	48	8	2	1	3	2	1	0	0
2	8	16	0	1	1152	144	36	2	4	2	1	0	0
3	9	18	1	0	72	8	2	1	3	2	1	0	0
4	10	20	0	0	320	32	8	1	4	2	1	0	0

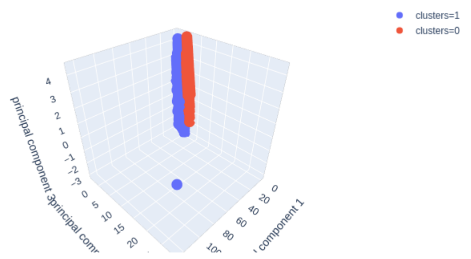

Fig. 3. Dataset and clusters ($K = 2$) after using PCA.

After clustering for $K = 3$ we get 3565 graphs in the first cluster, 2879 graphs in the second cluster and 920 graphs in the last one. As before, we use PCA to plot the results for $K = 3$ in Figure 4. Observe that there are some outliers in Figures 3 and 4. That happens because there are some very large numbers in columns “order of symmetry group”, “order of a vertex stabilizer” and “order of dart stabilizer”. After getting the cluster sizes we noticed that columns “dart-transitive”, “semi-symmetric” and “half-arc-transitive” had a huge impact on the clustering. When $K = 2$ we saw that all the semi-symmetric graphs were part of cluster 1 and all the “half-arc-transitive” and “dart-transitive” graphs were part of the other cluster.

TABLE II
CLUSTERS AND SYMMETRIES TABLE, ORIGINAL DATA ($K = 2$)

	0	1
semi-symmetric	0	2879
half-arc-transitive	920	0
dart-transitive	3565	0

When $K = 3$ each cluster had only graphs with one type of symmetry.

TABLE III
CLUSTERS AND SYMMETRIES TABLE, ORIGINAL DATA ($K = 3$)

	0	1	2
semi-symmetric	0	2879	0
half-arc-transitive	0	0	920
dart-transitive	3565	0	0

In the following subsection we will adjust the data to resolve the issue of outliers and the issue of clustering all the graphs of the same symmetry in the same cluster.

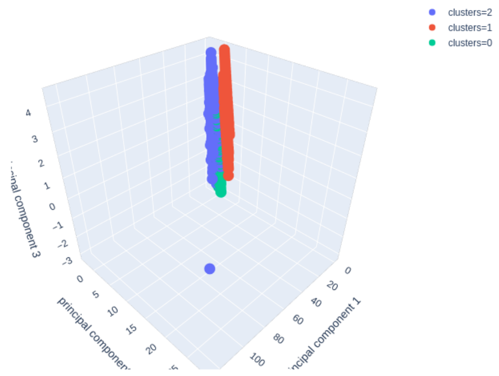

Fig. 4. Dataset and clusters ($K = 3$) after using PCA.

B. *The Logarithmized dataset*

Based on the observations in the previous subsection we decided to take the natural logarithm function of the three columns that were causing the outliers. In Table IV we can see the new version of the data. As before, we will get the optimal number of clusters from elbow and silhouette methods. From Figure 5 we see that the recommended number of clusters is 2. The silhouette scores for $K = 2, 3, 4$ are $0.601, 0.555, 0.547$, respectively. Observe that in this case the two methods agree on the number of clusters, namely $K = 2$.

TABLE IV
ORIGINAL DATASET

	no. vertices	no. edges	worthy	bipartite	order of symmetry group	order of a vertex stabilizer	order of dart stabilizer	no. of semi-transitive orientations	girth	diameter	dart-transitive	semi-symmetric	half-arc-transitive
0	5	10	1	0	4.787492	3.178054	1.791759	0	3	1	1	0	0
1	6	12	0	0	3.871201	2.079442	0.693147	1	3	2	1	0	0
2	8	16	0	1	7.049255	4.969813	3.583519	2	4	2	1	0	0
3	9	18	1	0	4.276666	2.079442	0.693147	1	3	2	1	0	0
4	10	20	0	0	5.768321	3.465736	2.079442	1	4	2	1	0	0

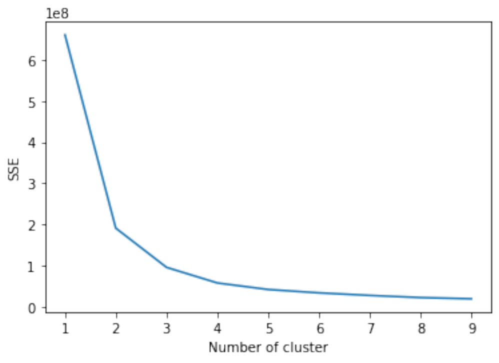

Fig. 5. The Elbow method for the log transformations.

Let's see The silhouette plot for $K = 2$ is in Figure 6.

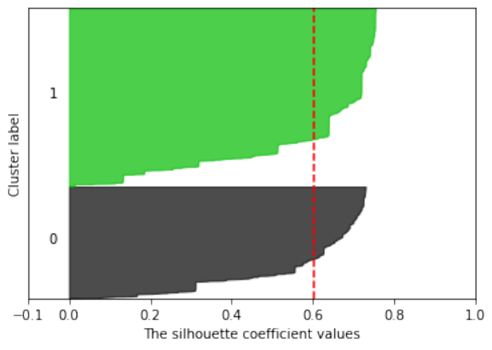

Fig. 6. The silhouette plot for $K = 2$ of the logarithmized dataset.

Since methods agreed on the number of clusters we go ahead and cluster our tweaked dataset for $K = 2$. We get 4525 graphs in the first cluster and 2839 in the second one. We use PCA to visualize the results; see Figure 7.

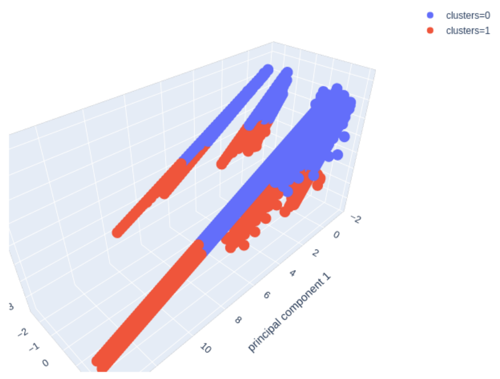

Fig. 7. Logarithmized dataset and clusters for $K = 2$ using PCA.

The logarithmization of the dataset also solved the issue of clustering the graphs of the same symmetry in one cluster as we can see in Table V.

TABLE V
CLUSTER AND SYMMETRIES TABLE, LOGARITHMIZED DATA ($K = 2$).

	0	1
semi-symmetric	944	1935
half-arc-transitive	295	625
dart-transitive	1600	1965

We will also inspect $K = 3$ even though the methods agreed on the number of clusters just to see how it looks like and how many values are there for each cluster.

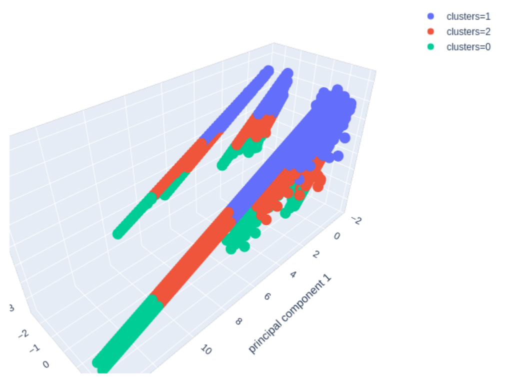

Fig. 8. Logarithmized dataset and clusters for $K = 3$ using PCA.

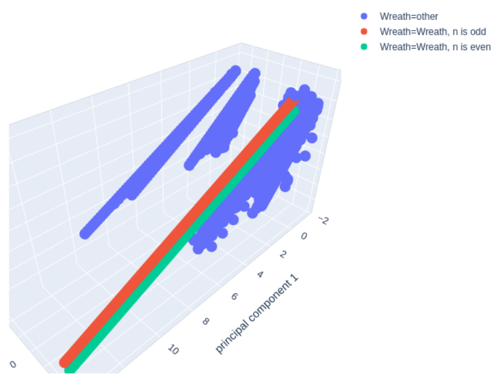

Fig. 9. The Wreath graph family.

First cluster has 2776 graphs, second cluster has 2628 and the last one has 1960 graphs.

TABLE VI
CLUSTER AND SYMMETRIES TABLE, LOGARITHMIZED DATA ($K = 3$).

	0	1	2
semi-symmetric	625	1146	1108
half-arc-transitive	198	386	336
dart-transitive	1137	1096	1332

From Table VI, we see that, similarly as in the $K = 2$ case, the two issues are not relevant anymore.

IV. Conclusion

In subsection *The Original dataset* the prediction was based only on the three symmetry columns and the clusters were exactly the three types of symmetry. Interpreting the results and the plots from *The Lograrithmized dataset* subsection it looks like somebody cut with a knife the datset in three pieces. We do not see what associates the graphs with those clusters. This dataset contains some families that can be seen in the plots, more precisely they correspond to certain lines in the plot. However K-means algorithm did not detect them. For example member of the *Wreath* family belong to the all the clusters; see Figure 9.

V. Future work

It seems that there are a lot of features but some of them are in correlation with one-another. For example the order of the vertex stabilizer is just four times the order of the dart stabilizer and also the order of the symmetry group is the product of the number of vertices and the order of the vertex stabilizer. In order to get more meaningful results, new features are required. Ideally, those features should be independent from those in the dataset. Thus we plan to continue this research work by enriching our dataset with more information. We will use *SageMath* [18] to get extra information on the graphs in this dataset. The extra information should include graph invariants such as: chromatic number, properties of their line graphs, vertex cover number, edge covers number, domination number, independence number, treewidth, matching number, genus, vertex connectivity etc. This will take some time because some of the problems are NP-hard. The aim of this continuation is to find more "meaningful" clusters. We will also make an attempt at the supervised learning by trying to predict one of the new features from the old ones. This will hopefully lead to discovery of novel connections between these properties.

References

[1] S. Wilson & P. Potočnik. Recipes for Edge-Transitive Tetravalent Graphs. arXiv: 1608.04158.2016.

[2] K. Berčič, M. Kohlhase & F. Rabe. Towards a Unified Mathematical Data Infrastructure: Database and Interface Generation. DOI: 10.1007/978-3-030-23250-4_3. In book: Intelligent Computer Mathematics (pp.28-43)

[3] B. McKay. Graph Formats. https://users.cecs.anu.edu.au/~bdm/data/formats.html

[4] V. G. Nair. Getting Started with Beautiful Soup. ISBN: 9781783289554.

[5] R. Garreta, G. Moncecchi. Learning scikit-learn: Machine Learning in Python. ISBN: 978178328193052999

[6] A. B. Downey. Think Stats. ISBN: 9781449313104.

[7] G. James, D. Witten, T. Hastie, R. Tibshirani. An Introduction to Statistical Learning. ISBN: 9781461471370.

[8] R. V. Chandra, B. S. Varanasi. Python Requests Essentials. ISBN: 9781784395414.

[9] A. Chapagain. Hands-On Web Scraping with Python. ISBN: 9781789533392.
[10] D. M. McGreggor. Mastering Matplotlib. ISBN: 9781783987542.
[11] C. Adams. Learning Python Data Visualization. ISBN: 9781783553334.
[12] D. Zinoviev. Complex Network Analysis in Python. ISBN: 9781680502695.
[13] V. David. JSON: Main principals.
[14] D. Y. Chen.Pandas for Everyone: Python Data Analysis. ISBN: 9780134546933.
[15] I. Idris. NumPy: Beginner's Guide. ISBN: 9781785281969.
[16] J. Friedl. Mastering Regular Expressions. ISBN: 9780596528126.
[17] W. Bosma, J. Cannon, and C. Playoust, The Magma algebra system. I. The user language, J. Symbolic Comput., 24 (1997), 235–265.
[18] P. Zimmermann, A. Casamayou, N. Cohen, G. Connan, T. Dumont, L. Fousse, F. Maltey, M. Meulien, M. Mezzarobba, C. Pernet, N. M. Thiéry, E. Bray, J. Cremona, M. Forets, A. Ghitza, H. Thomas. Computational Mathematics with SageMath.