A meta-heuristic approach for graph genus problem

Syed Mujtaba Hassan* Sudais Yasin †

May 8, 2024

Abstract

Graph genus problem can be dated back to the map coloring problem proposed in 1890 by Heawood. The problem is to draw a graph on an orientable surface with n holes without any edges crossing. The minimum number of holes any such surface can have is known as the genus of the graph. The genus problem is known to be **NP-hard**. In this project we introduce a meta-heuristic approach for the graph genus problem. We apply an evolutionary algorithm (EA) to approximate a solution to the graph genus problem.

Keywords: graph theory, graph genus, algebraic topology, evolutionary algorithms

1 Introduction

Graph genus problem can be dated back to the map coloring problem proposed in 1890 by Heawood [14]. Intuitively the problem is to draw a graph on an orientable surface with n holes without any edges crossing. The minimum number of holes any such surface can have is known as the genus of the graph. The genus of a graph is the primary criterion of non-planarity. The genus problem is known to be **NP-hard** [12]. The problem is known to be **NP-hard** even for cubic graphs [13]. The problem is one of the hardest problems in graph theory and algebraic topology. Furthermore, the problem has a higher complexity even for smaller graphs. The brute force approach run in $O(|V|^{\Delta(G)})$ time. Despite the hardness of the problem, there are several real-world applications of the genus problem, such as utility problem, and circuit embedding, among others.

We introduce a meta-heuristic approach for the graph genus problem. We apply an evolutionary algorithm (EA) to approximate a solution to the graph genus problem. This is the meta-heuristic approach to the genus problem. In section 2 we shall introduce our definitions and the problem more formally. In section 4 we give an EA formulation of our graph genus problem. Our formulation and algorithm are then experimentally evaluated in section 5.

2 Topologic and graph theoretic preliminaries

We first need to go over some graph theoretic and topological prerequisites. Throughout this document, we will denote a graph as $G = (V, E)$ where G is a graph with vertex set V and edge set E. An edge is represented as a set $\{v, u\}$ for some v and u belonging to V such that $v \neq u$. Also, for a graph G we use $V(G)$ to denote the vertex set of G and $E(G)$ to denote the edge set of G. Unless specified, all graphs considered here are undirected and simple meaning they contain no loops, multi-edges or directed edges. For each vertex $v \in V$ we used $d_G(v)$ to denote the degree of v in G. We also use $\Delta(G)$ to denote the maximum degree of G and $\delta(G)$ to denote the minimum degree of G. An isomorphism between graphs $G = (V, E)$ and $G' = (V', E')$ is a bijection $f : V \to V'$ such that $\forall v, u \in V$, $\{v, u\} \in E \iff \{f(v), f(u)\} \in E'$ [4, 15]. For a vertex $v \in V$, $N(v)$ denotes the set of neighbors of v [4]. For a graph $G = (V, E)$, $D(G)$ denotes the directed graph obtained from G such that $V(D(G))$ and $E(D(G)) = \{(u, v), (v, u) | \{u, v\} \in E\}$. So we replace every undirected edge in G with a directed edge in $D(G)$. A rotation at a vertex v is a cyclic order

*ms06948@st.habib.edu.pk, Computer Science Department, Habib University, Karachi, Pakistan
†sy06541@st.habib.edu.pk, Computer Science Department, Habib University, Karachi, Pakistan

(counter-clockwise) of the neighbors of v. A rotation system R of a graph G is a set of rotations for each vertex $v \in V$ [1]. A directed graph is a graph $G = (V, E)$ where edges have direction, so an edge from v to u is represented as (v, u), for $v, u \in V$. In a directed graph for a vertex $v \in V$, $d^+(v)$ denotes the out going edges of v and $d^-(v)$ incoming edges on v.

We shall denote a surface X, by topological space (X, τ), where X is a the set topology is defined on and τ is the topology on X. A homeomorphism between two topological spaces is a continuous bijective map whose inverse is also continuous [10]. An n-manifolds is a Hausdorff space such that each point has an open neighborhood homeomorphic to the open n-dimensional disc $U^n = \{x \in R^n \mid |x| < 1\}$ [9]. A connected 2-manifold is said to be orientable if every closed path is orientation preserving (traversing the path does not change the orientation) [9]. Now we shall restrict the surfaces we deal with further. Here the surfaces we deal with are orientable, compact and connected 2-manifolds. Also, we only consider R^3 under the usual topology, so we are dealing only with subspaces of R^3 under the usual topology. Intuitively a Torus is a surface homeomorphic to a donut or a coffee mug. Formally, a torus is any topological space homeomorphic to the product space of two circles in R^2, $S^1 \times S^1$ [9]. In other words, a torus is any topological space homeomorphic to the subset $T = \{(x, y, z) \in R^3 \mid (\sqrt{x^2 + y^2} - 2)^2 + z^2 = 1\}$ of R^3 [9]. There are many equivalent formal definitions of a torus we presented two of the easiest ones here. Figure 1 shows an example of a torus. For two surfaces S_1 and S_2, their connected sum is a surface S, denoted by $S_1 \# S_2$, which is

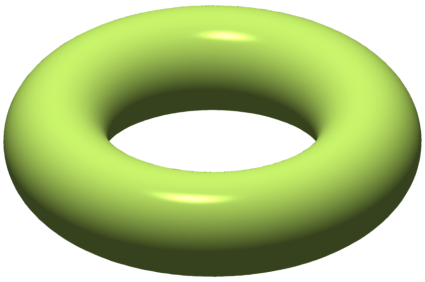

Figure 1: Torus

obtained by cutting a small circular hole in each surface, and then gluing the two surfaces together along the boundaries of the holes [9]. A surface that is the connected sum of n tori has a genus n. Intuitively this means the surface has n holes [9]. Figure 2 shows two examples of surfaces with genera 2 and 3. Genus of a sphere is defined to be 0.

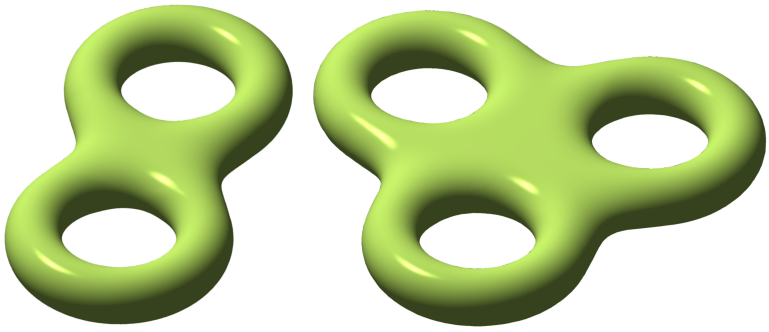

Figure 2: surfaces of genus two and three

With these graph theoretic and topological prerequisites we shall define the graph genus problem in section 2.

Genus of a graph

Before we can go into the genus of the graph we shall first define what we mean by embedding a graph $G = (V, E)$ on a surface X. Embedding G onto X is drawing G on X, such that no two edges of G cross each other. More formally our drawing is obtaining a subset Σ of X such that, for every vertex v in V we have a point x in Σ corresponding to v, and for every edge $\{v, u\}$, if x and y are points in Σ corresponding to v and u respectively then, there is a polygonal arc p with endpoints x and y, such that all $p \subseteq \Sigma$, and for each such p, $p \setminus \{x, y\}$ contains any point from any other polygonal arc or any point corresponding to a vertex. And finally, G is isomorphic to the graph Σ. The regions of $X \setminus \Sigma$ are called the faces of Σ, intuitively these are the regions enclosed by the vertices and edges of Σ.

Now we define what we mean by the genus of a graph. For a graph G the genus n of the surface X with the smallest genus such that G can be embedded on X is called the genus of G. It is denoted by $\gamma(G)$. For planar graphs the genus is 0, they can be embedded on a sphere, which minus its north pole is homeomorphic to a plane in R^2. For non-planar graphs, we will need surfaces of high genera. Figure 3 shows the embedding of K_6 (the complete graph with 6 vertices) on a torus.

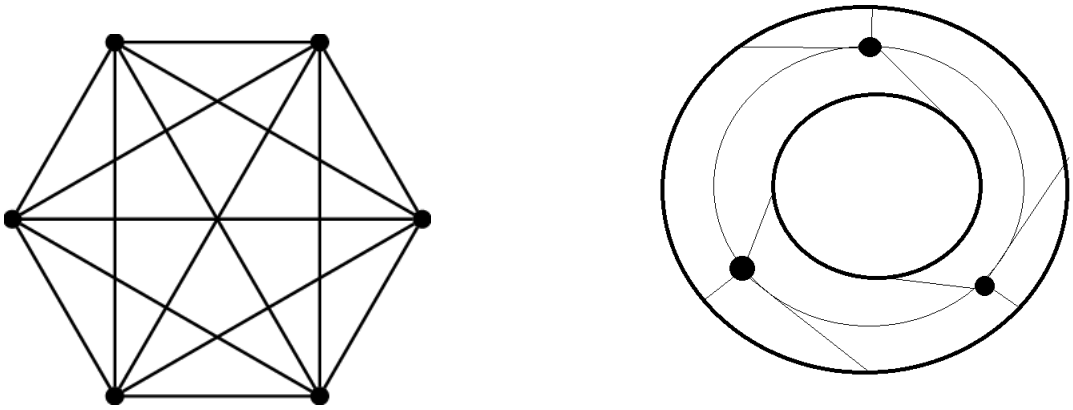

Figure 3: K_6 and its embedding on a torus, 3 vertices are placed on the top side of the torus and 3 are placed on the bottom of the torus.

Now the decision version of the graph genus problem is the following; GENUS $= \{\langle G = (V, E), k\rangle \mid \gamma(G) \leq k\}$. While the optimization version of the problem is; MIN-GENUS $= \{\langle G = (V, E), k\rangle \mid \gamma(G) = k\}$. So given a graph G we need to find the genus of G.

For a long time, the complexity of GENUS was an open problem, in their book Garey and Johnson wrote it as one of the open problems in computational complexity theory [7]. However, in 1989 Carsten Thomassen showed that GENUS was **NP-complete** [12]. He later in 1997 showed that it was **NP-complete** even when the input is restricted to cubic graphs [13]. While the optimization version of the problem MIN-GENUS is not even known to be in **NP** and if **NP-hard**. However, the problem has been shown to be fixed-parameter tractable as for any fixed genus there are only a finite number of forbidden graph minors [1], but for every surface, these are not known and finding the set of forbidden minors for a surface is not an easy task. So the problem remains one of the hardest problems in graph theory and algebraic topology. Furthermore, the problem has a higher complexity even for smaller graphs. The brute force approach run in $O(|V|^{\Delta(G)})$ time.

3 Application of Genus problem

We now provide some application motivation for the graph genus problem. The utility problem is an infamous mathematical puzzle with numerous real-world applications. Kullman in his paper gives a comprehensive survey of the problem [8]. According to Kullman, the problem dates back to ancient times, while the first

published instance of the problem was given in 1917 by Henry Ernest Dudeney in his book Amusements in Mathematics [5, 8]. The basic version of the problem is as follows, given three utilities, water, gas and electricity, and three houses A, B and C, on a street. Now the task is to place the three utility connections to the three houses such that no connection crosses each other. For the three houses and utilities given in our example, the connections cannot be drawn on a plane. Figure 4 illustrates our example. Now we do however want to draw our connection, so maybe if we were given access to a tunnel we could pass some of our connections through it in order to solve the problem. For our example of three houses and three utilities only one such tunnel would be required. Now imagine we have some n number of houses and m number of utilities and now some houses might have some sort of connection to be built between them, or some utilities might need some connections between them. Now we need to know how many tunnels would be required to make our connections successfully So the problem becomes increasingly difficult. Another application of this problem comes in circuit embeddings. When designing circuits we wish to embed on silicon chips we don't want different connections between our circuit components to cross. So our aim is to embed our circuit in such a that no connection crosses each other, it is of interest to us to find our how many holes do we need in order to successful embed the circuit. These real-world applications give us the motivation to define the graph genus problem.

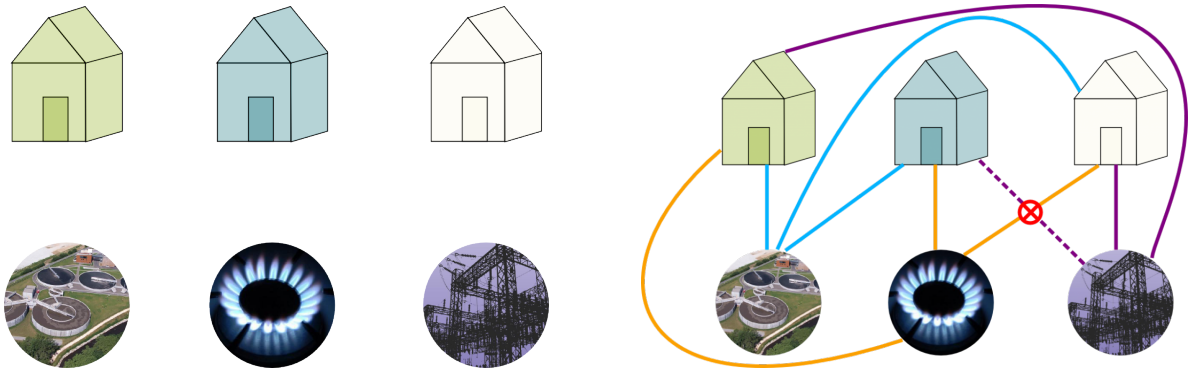

Figure 4: An illustration of utility problem

4 EA formulation

As we saw the genus problem is quite a difficult problem to solve, and we have that no exact polynomial time solution exists for the problem unless $\mathbf{NP} = \mathbf{P}$. We would apply a meta-heuristic approach to solve the graph genus problem. More specifically we want to use an evolutionary algorithm to solve the graph genus problem.

For this approach, it is of interest to us to formulate the problem as a combinatorial optimization problem. Throughout history, many such approaches have been made most famously the formulation given by Filotti in 1979 [6]. However Wendy Myrvold and William Kocay showed that there is a flaw in the formulation by Filotti, they also identified errors in many famous algorithms which try to solve the graph genus problem [11]. However many other valid formulation exists, for example, the formulation given by G. Brinkmann [2], Marston Conder and Klara Stokes [3], Stephan Beyer and Markus Chimani and Ivo Hedtke and Michal Kotrbcik [1], among others.

For our approach, we solely consider combinatorial embeddings. For a graph $G = (V, E)$ A combinatorial embedding is given by a rotation system R defined on $D(G)$. Let P be the set of all rotation systems on $D(G)$, in other words, P is the set of all permutations on $N(v)$ for each $v \in V$. Let $F : P \to \mathbb{N}$ be a computable function such that given a rotation system R maps the number of faces in the embedding given by R. Listing 1 gives an algorithm that computes F. The algorithm runs in $O(|E|)$ time. So now given a rotation system (which corresponds to an embedding) we can compute the the number of faces in that embedding in polynomial time. Now the number of faces we compute through this formulation connects beautifully the genus of the graph, by the following equation $2 - 2\gamma(G) = |V| - |E| + f$ [9]. Through this,

once we have to maximize the number of faces we can compute the minimum genus of the graph. So our problem is now to find a rotation system that maximizes the number of faces in the corresponding embedding.

```
def faceTracing(Graph, rotation_system):
    D = list(get_directed_edges(Graph))
    faces = 0
    unused = D
    arc = unused[0]
    start = arc
    while len(unused) > 0:
        unused.remove(arc)
        arc = (arc[1], rotation_system[arc[1]][
            (rotation_system[arc[1]].index(arc[0]) + 1)%len(rotation_system[arc[1]])
            ])
        if arc == start:
            faces+=1
            if len(unused) <= 0:
                break
            arc = unused[0]
            start = arc
    return faces
```

Listing 1: Facetracing Algorithm

We apply an evolutionary algorithm to approximate the optimal rotation system. Our chromosomes which are our candidate solutions are tuples corresponding to a rotation system. So each chromosome C is a nested tuple, such that $C[v]$ corresponds to the permutation of outgoing edges around $v \in D(G)$. The crossover method used in our EA approach is the classic two-point cross-over. For parents P_1 and P_2, we construct child C by taking permutation around half of the vertices of P_1 and the other half of P_2. The mutation scheme is as follows, we randomly select a vertex v and then select two random edges going from that vertex, we then swap the position of those two edges creating a new permutation about v. The selection schemes implemented are *Fitness propositional selection*, *Ranked selection*, *Tournament selection*, *Truncation*, *Random selection*. In section 5 we test multiple selection schemes and evaluate our results. So this gives us an EA formulation of the graph genus problem in the next section we will go over the experimental testing evaluation of our algorithm.

5 Experimental Evaluation

We now test our algorithm on seven graphs, k_4 with $\gamma(k_4) = 0$, k_5 with $\gamma(k_5) = 1$, k_7 with $\gamma(k_7) = 1$, k_8 with $\gamma(k_8) = 2$, k_9 with $\gamma(k_9) = 3$, k_{10} with $\gamma(k_{10}) = 4$, *petersen graph* with $\gamma(petersen\ graph) = 1$, and $k_{14,12}$ with $\gamma(k_{14,12}) = 30$.

The best results were obtained with fitness promotional parent selection scheme and truncation survivor selection. Our mutation rate was 0.7. Our experimental results are shown in figures 5, figures 6, figures 7, figures 8, figures 9, figures 10, figures 11, and figures 12 shows the plots obtained from our experiments. We see that on smaller graphs we are reaching the minimum genus while on larger graphs we see a convergence pattern.

6 Conclusion

In this project, we provided the first meta-heuristic approach to the graph genus problem. We applied an evolutionary approach to the problem and tested our algorithm experimentally. As the problem is computationally expensive even for small graphs, we see that our formulation gives a practical algorithm to solve the graph genus problem. In future work, we can tune our hyperparameters and test our algorithm for much larger graphs.

All the code for our algorithm can be found on our git repository: `https://github.com/nitrodragonoid/Meta-Heuristic-Approach-for-Graph-Genus.git`

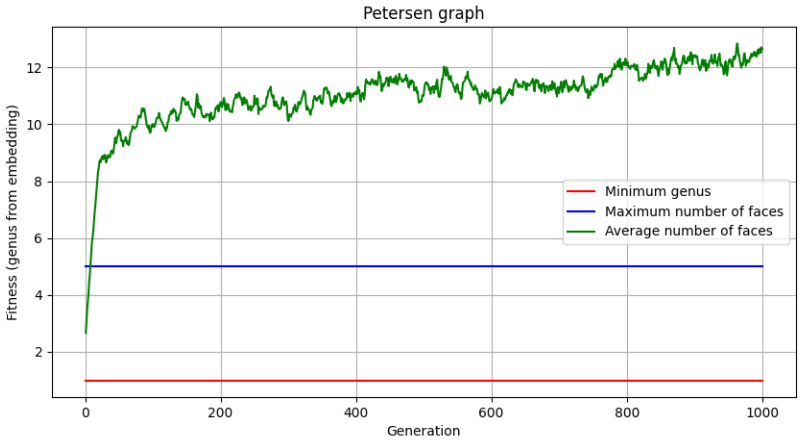

Figure 5: Plots obtained from our algorithm tested on *petersen graph*.

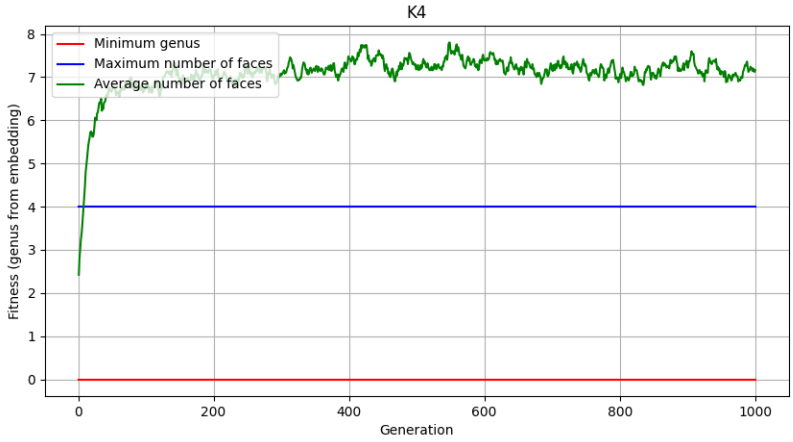

Figure 6: Plots obtained from our algorithm tested on k_4 graph.

References

[1] Stephan Beyer, Markus Chimani, Ivo Hedtke, and Michal Kotrbcík. A practical method for the minimum genus of a graph: Models and experiments. In *The Sea*, 2016.

[2] G. Brinkmann. A practical algorithm for the computation of the genus, 2020.

[3] Marston Conder and Klara Stokes. New methods for finding minimum genus embeddings of graphs on orientable and non-orientable surfaces. *Ars Mathematica Contemporanea*, 17:1–35, 06 2019.

[4] Reinhard Diestel. *Reinhard Diestel Graph Theory Electronic Edition 2017.* Springer-Verlag Heidelberg, 1997.

[5] H.E. Dudeney. *Amusements in Mathematics.* Thomas Nelson, London, 1917.

[6] I. S. Filotti, Gary L. Miller, and John H. Reif. On determining the genus of a graph in o(v o(g)) steps(preliminary report). *Proceedings of the eleventh annual ACM symposium on Theory of computing*, 1979.

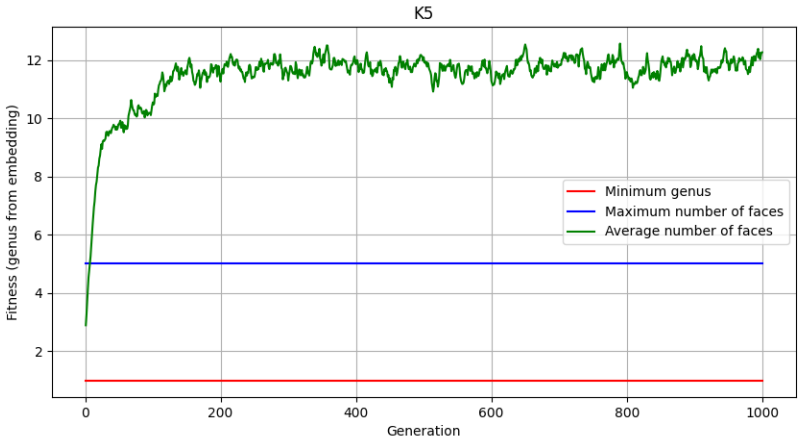

Figure 7: Plots obtained from our algorithm tested on k_5 graph.

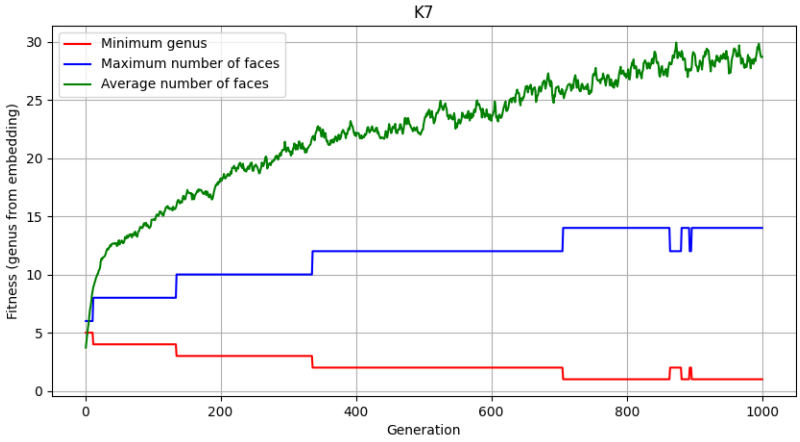

Figure 8: Plots obtained from our algorithm tested on k_7 graph.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[8] David E. Kullman. The utilities problem. *Mathematics Magazine*, 52(5):299–302, 1979.

[9] W.S. Massey. *A Basic Course in Algebraic Topology*. Graduate Texts in Mathematics. Springer New York, 1991.

[10] J.R. Munkres. *Topology*. Featured Titles for Topology. Prentice Hall, Incorporated, 2000.

[11] Wendy Myrvold and William Kocay. Errors in graph embedding algorithms. *Journal of Computer and System Sciences*, 77(2):430–438, 2011. Adaptivity in Heterogeneous Environments.

[12] Carsten Thomassen. The graph genus problem is np-complete. *Journal of Algorithms*, 10(4):568–576, 1989.

[13] Carsten Thomassen. The genus problem for cubic graphs. *Journal of Combinatorial Theory, Series B*, 69(1):52–58, 1997.

[14] Liangxia Wan. The genus of a graph: A survey. *Symmetry*, 15(2), 2023.

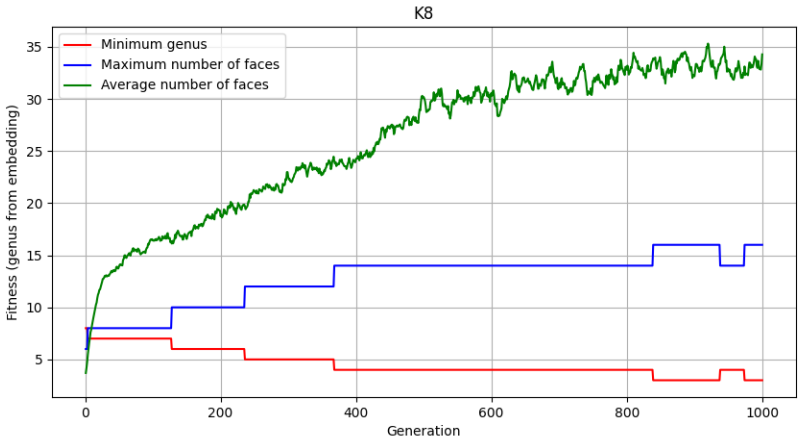

Figure 9: Plots obtained from our algorithm tested on k_8 graph.

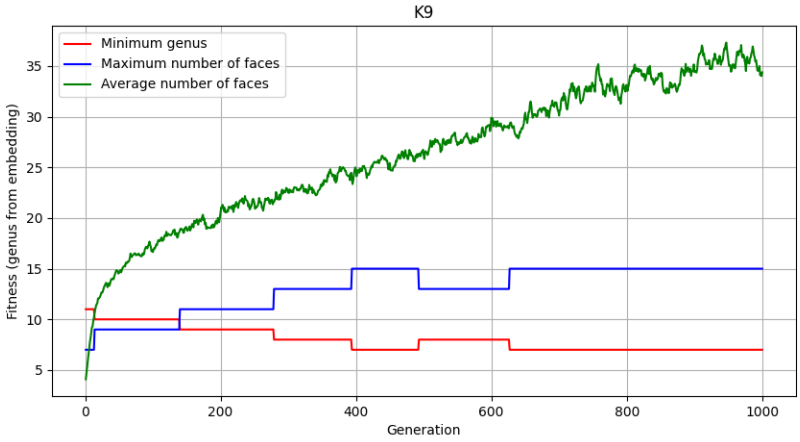

Figure 10: Plots obtained from our algorithm tested on k_9 graph.

[15] Douglas Brent West. *Introduction to Graph Theory.* Prentice Hall, 2001.

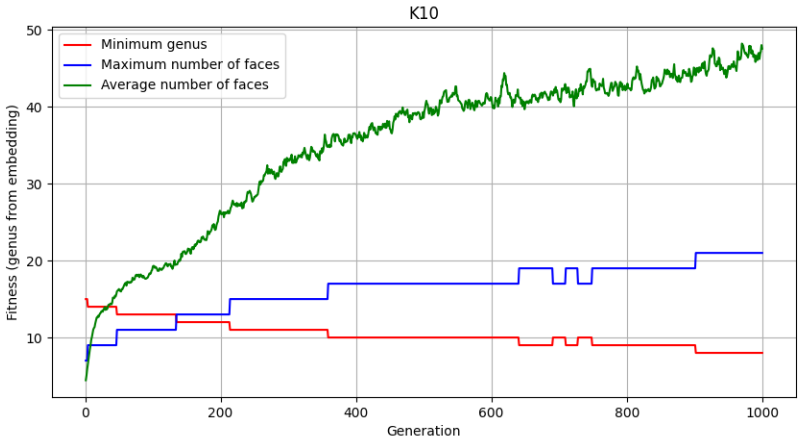

Figure 11: Plots obtained from our algorithm tested on k_10 graph.

Figure 12: Plots obtained from our algorithm tested on $k_{14,12}$ graph.