USE ANT COLONY OPTIMIZATION TO SOLVE A THICKNESS PROBLEM

by

KHA, H. MAN

M.S., University of Colorado Denver, 2017

A Master project submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Computer Science

2017

Man, Kha, H. (M.S., Computer Science)

Use Ant Colony Optimization to solve a thickness problem

Thesis directed by Dr. Ellen Gethner

ABSTRACT

The thickness of a graph is the minimum number of planar graphs needed for decomposing G into planar graphs. Heawood proved that a graph of thickness M can be 6M-colored. He conjectured an inequality holds for the M-pire chromatic number of S. The purpose of this study is to find a set of graphs of thickness t with the largest chromatic number as possible by computer science tool. Particularly, because the problem is NP-hard, we applied Ant Colony Optimization (ACO) for doing the search heuristically. We have adapted the theory of ACO for doing the job and we have found some results from it. Although there are some data included in the appendix, they are still unverified and the project is unfinished.

The form and content of this abstract are approved. I recommend its publication.

Approved: Ellen Gethner

ACKNOWLEDGMENT

I would like to show my deep appreciation and gratitude to my advisor Professor Dr. Ellen Gethner for her advices and supports throughout the years that I have been in the department of Computer Science and Engineering. I also want to say thank to Jeff Flower for collaborating with me in analyzing the theory of Ant Colony Optimization and programming. Without him, I think I am not able to get the project done. I specially appreciate the helps of Dr. Thom Sulanke for sharing his past works and documents to me. I have taken a very long time for being the department and I have received many supports from the department restlessly. All of what I have gained and learned in the past 3 years are priceless.

TABLE OF CONTENTS

Tables vii

Chapter

Figures ix

1. Introduction 1

2. Background of the Study 2

2.1 Graphs 2

2.2 Ant Colony Optimization 4

2.3 Ant Colony System 5

2.3.1 Tour Construction 5

2.3.2 Global Pheromone Trail Update 6

2.3.3 Local Pheromone Trail Update 6

2.4 Adapt Ant Colony System for solving the thickness problem 7

3. Data Structures 8

3.1 Triangulation 8

3.2 Ants 8

3.3 Pheromone τ 9

4. Algorithms 10

4.1 Flowchart of the Main Program 10

4.2 Counting Cliques 10

4.3 Calculate Initial Pheromone 10

4.4 Finding a Flippable Edge 12

4.5 Randomize the Triangulation Creation 13

4.6 Coloring Algorithms 13

4.6.1 Degree of Saturation (DSATUR) 13

4.6.2 Minimum Vertex Coloring 14

4.6.3 The comparison between DSATUR and MVC 14

5. Experiments and Results 16
6. Conclusion 19
Appendix
A. Functions 20
A.1 ant_colony 20
A.1.1 ant_colony.c 20
A.1.2 ant_colony_system_initialization.c 20
A.1.3 ant_colony_system_pheromone_update.c: 20
A.1.4 ant_colony_system_stages.c 21
A.1.5 ant_structs.c 22
A.1.6 calculate_initial_pheromone.c 22
A.1.7 display_parameters.c 22
A.1.8 free_arrays 22
A.1.9 initialize_ant_parameters 22
A.1.10 initialize_arrays 22
A.1.11 Triang_lex 22
A.2 Coloring 22
A.2.1 Dharwadker_coloring 23
A.2.2 dsatur_coloring.c 23
A.2.3 get_critical_graph.c 23
A.3 includes 23
A.3.1 triang.h 23
A.3.2 triang_utils.h 23
A.4 triang 23
A.5 triang_utils 23
A.5.1 count_cliques.c 23
A.5.2 FlipEdge.c 23

A.5.3 generate_random_triang.c 23
A.5.4 get_complement_graph.c 24
A.5.5 get_flippable_edges.c 24
A.5.6 Get_int.c 24
A.5.7 load_default_triangles.c 24
A.5.8 log_function.c 24
A.5.9 merge_2_triang.c 24
A.5.10 permute_triang.c 24
A.5.11 print_neighbors2.c 25
A.5.12 print_triang.c 25
A.5.13 ran_flip_triang.c 25
A.5.14 scramble_triang.c 25
A.5.15 set_undirected_edge.c 25
B. Compiling the code 26
C. Plot graphs from the results collected from Sulanke's application 28
D. Some examples of solutions we have found in chapter 5 29
References 71

TABLES

Table

5.1 Some testing cases from the inequality 1.1 17

5.2 The results of graph thickness 2 from testing ACO algorithm 18

D.1 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 30

D.2 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 32

D.3 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 34

D.4 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 36

D.5 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 38

D.6 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 40

D.7 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 42

D.8 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 44

D.9 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 46

D.10 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 48

D.11 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 50

D.12 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 52

D.13 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 54

D.14 nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 56

D.15 nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ 58

D.16 nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ 60

D.17 nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ 61

D.18 nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ 62

D.19 nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ 63

D.20 nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$ 64

D.21 nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$ 65

D.22 nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$ 66

D.23 nv=19, genus=2, orient=0, thickness=2, K_3-free, $\chi = 9$ 67

D.24 nv=19, genus=2, orient=1, thickness=2, K_3-free, $\chi = 9$ 68
D.25 nv=21, genus=2, orient=0, thickness=2, K_3-free, $\chi = 11$ 69
D.26 nv=21, genus=2, orient=1, thickness=2, K_3-free, $\chi = 10$ 70

FIGURES

Figure

4.1 The flowchart of the program. 11

4.2 Calculate the initial pheromone. 12

D.1 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 31

D.2 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 31

D.3 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 33

D.4 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 33

D.5 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 35

D.6 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 35

D.7 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 37

D.8 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 37

D.9 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 39

D.10 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 39

D.11 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 41

D.12 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 41

D.13 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 43

D.14 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 43

D.15 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 45

D.16 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 45

D.17 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 47

D.18 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 47

D.19 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 49

D.20 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 49

D.21 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 51

D.22 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 51

D.23 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 53

D.24 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 53
D.25 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 55
D.26 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 55
D.27 Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 57
D.28 Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$ 57
D.29 Triang 1 - nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ . . . 59
D.30 Triang 2 - nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$ . . . 59

1. Introduction

Graph theory is a study of graphs which are used to model in many areas such as networks, social, computing problems and natural sciences. Bridges in Konigsberg [Kra42], Traveling Salesman Problem [ABCC07] are ones of the popular graph examples which inspired the beauty of graphs.

Graph coloring is a research field of graph theory. That is, to assign the fewest number of colors to vertices in a graph such that there are no adjacent vertices having the same color. Four Color Theorem is a famous example of graph coloring.

In the project, we need to find solutions for a graph coloring problem by computer science tools. The statement of that problem is *"What is the best upper bound for the chromatic number of any genus g thickness t orientable graph?"*. This question is generalized from Ringel's Earth Moon problem [JR83]. Heawood actually found an inequality related to M-pire chromatic number [Hea90] as follows:

$$\chi(M - pire, \varepsilon) \leq \frac{6M + 1 + \sqrt{(6M+1)^2 - 24\varepsilon}}{2} \tag{1.1}$$

It is NP-hard to determine the thickness of an input graph and NP-complete to determine whether the thickness is at most two [Man93]. Therefore, in this study, we have used Ant Colony Optimization technique to heuristically search the solutions which chromatic number are close to the upper bound of the equation above. The searches in this project are restricted for sphere and the orientable graphs. By the way, there are some conducted studies about the chromatic number of particular graph settings such as 9-chromatic critical graphs by Boutin, Gethner and Sulanke [BGS07] [GS09], thickness-2 graphs by Heawood [Hea90] and Ringel [Rin59].

2. Background of the Study

Let me introduce some definitions that we will use to present this project. Most of the terms based on the book [Wes01].

2.1 Graphs

Definition 2.1 *A graph is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called its endpoints.*

Definition 2.2 *A complete graph of n vertices, denoted as K_n, is a simple graph whose vertices are pairwise adjacent.*

By definition, the number of edges in K_n is $\frac{n(n-1)}{2}$

Definition 2.3 *A graph G is planar if there exists a plane embedding of G without edge crossings.*

Definition 2.4 *The thickness of a graph G is the minimum number of planar graphs needed for decomposing G into planar graphs.*

Definition 2.5 *A $k-$coloring of a graph G is to assign k colors to $V(G)$ vertices in G. A $k-$coloring is proper if adjacent vertices have different colors. The chromatic number $\chi(G)$ is the least number of colors k such that G is properly k-colorable. If $\chi(H) < \chi(G) = k$ for every proper subgraph H of G, then G is k-critical.*

Definition 2.6 *The complement $\overline{G}$ of a simple graph G defined by $uv \in E(\overline{G})$ if and only if $uv \notin E(G)$. A clique of a graph G is a complete subgraph of G. An independent set in a graph G, denoted as $\alpha(G)$,is a set of pairwise nonadjacent vertices.*

Proposition 2.7 *Given a graph G of n vertices, $\chi(G) \geq \frac{|V(G)|}{\alpha(G)}$.*

By the definition, an independent set contains non-adjacent vertices which can have the same color. So $\frac{|V(G)|}{\alpha(G)}$ is basically the number of different independent sets and they should have all different colors.

Proposition 2.8 *Given a graph G of n vertices, If $\overline{G}$ is $K_m - free$, then $\chi(G) \geq \lceil \frac{n}{m-1} \rceil$*

Proof: If $\overline{G}$ is $K_m - free$, so $\alpha(G) \leq m - 1$. Using the proposition 2.7, we have $\chi(G) \geq \frac{|V(G)|}{\alpha(G)} \geq \frac{n}{m-1}$

Definition 2.9 *Triangulation is a planar graph in which every face boundary is 3-cycle.*

Definition 2.10 *A handle is a tube joining two holes cut in a surface. The torus is the surface obtained by adding one handle to a sphere.*

Definition 2.11 *The genus g is the number of handles added into the sphere.*

The graphs embedded on surfaces of genus 0,1,2 are planar, toroidal and double-toroidal respectively.

Definition 2.12 *Orientable surface is a surface with two distinct sides. So, non-orientable surface is a surface with only one side*

An example of non-orientable surface is Mobius strip. That is the non-orientable surface obtained by cutting a closed band into a single strip, giving one of the two ends thus producing a half twist, and then reattaching the two ends.

An example of an orientable surface is torus that looks like a doughnut. You can draw this surface by making a hole in a sphere. Clearly, torus has genus 1 by definition.

2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) [DS04]is an Artificial Intelligence algorithm inspired by the behavior and social organization of every single ant in an ant colony. One of the surprising behaviors of ants is the capability of finding the shortest paths to food source. It was explained in a biological way that by using pheromones, ants can communicate to others and unveil the "best" paths they should follow for the next food findings. ACO has been using in computer science for finding a solution of an optimization problem.

An ACO algorithm has 3 procedures: ConstructAntsSolutions, UpdatePheromones and DaemonActions [DS04]. There are many versions of ACO algorithms but they are mostly developed on these procedures.

ConstructAntsSolutions is a procedure that constructs a tour or a solution for every single ant to evaluate. For example, in the Traveling Salesman Problem (TSP), an Ant has to visit all given cities with the shortest path. After the construction, UpdatePheromones will be called to decide the amount of pheromones to deposit or evaporate.

UpdatePheromones is the second procedure that helps ACO learn the information of the tours. There are many policies that decide to increase or decrease the pheromones all of the options. For example, after constructing 10 tours, if we think the shortest tours can be a part of an optimization solution, the pheromone will be increased significantly; on the other hand, the pheromone on the longest path will be increased just a little bit. After rewarding pheromones, pheromone evaporation will be applied for forgetting all the worst parts that won't potentially build an optimization solution.

DaemonActions procedure is a place to apply some rules for local optimization. For example, the best solution is updated after every iteration and it will be rewarded at the end of each iteration because it probably leads to an optimization solution.

2.3 Ant Colony System

Ant Colony System (ACS) was an improved version of Ant Colony Optimization (ACO) technique. It was developed by Dorigo and Gambardella in 1997. All the ACS theory that we have used in this project is from the book "Ant Colony System" [DS04] written by Marco Dorigo and Thomas Stutzle. There are three differences between ACS and ACO. First, only the best-so-far tour can get pheromone evaporation and pheromone deposit. Second, if an ant passes through a path, it will take out some amount of pheromone. Third, ACS uses randomness to sometimes ignore the pheromone impact on choosing an option in tour construction. It allows the exploration to happen after a long term.

2.3.1 Tour Construction

ACS works with pheromones and the actual cost of a tour collected/experimented after running an iteration. Let the pheromone of an edge with endpoints i, j be τ_{ij} the remaining number of triangles in the union complement after the flip be C_{ij} where i, j are the endpoints of the flipping edge. A tour is constructed by following the next equation:

$$j = \begin{cases} \max_{j \in N_i^k} \tau_{ij}[\eta_{ij}]^\beta, & \text{if } q \leq q_0 (1.1.1) \\ p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{ij}]^\beta}, & \text{otherwise} (1.1.2) \end{cases} \tag{2.1}$$

where p_{ij}^k is the probability of choosing the edge (i,j) in the graph, the heuristic information η_{ij} is inversely proportional to the C_{ij}, α and β are the powers of the pheromone and η_{ij}, N_i^k is the list of members might be added by ant k. At the beginning, an amount of pheromone will be initialized for all edges. Because our goal is to gradually decrease to $C_{ij} = 0$, the heuristic information $\eta_{ij} = \frac{1}{C_{ij}}$, not C_{ij}. For instance, flipping an edge e_1 yields $C_{e1} = 1$ while flipping an e_2 yields $C_{e1} = 10$. Then $\eta_{e_1} = 1$ and $\eta_{e_2} = \frac{1}{10}$. Clearly, the probability of choosing e_1 is 10 times than e_2. Raising it to β extends the gap larger. Therefore, tuning β is really important to help ACS get the convergence fast but to avoid the biases. If β is too high, clearly in

(1.1.1), j opts for the highest values of η_{e_i}. If β is too low, the gap between solutions is not large so that the picks will be mostly random or based on τ_{ij}.

2.3.2 Global Pheromone Trail Update

Only the best solution gets updated the pheromone after each iteration by the following equation:

$$\tau_{ij}^{bs} = (1-\rho)\tau_{ij}^{bs} + \rho\eta_{ij}^{bs} \tag{2.2}$$

where τ_{ij}^{bs} and η_{ij}^{bs} are the pheromone trail and the heuristic information of the best solution respectively. Theoretically, the equation (1.2) is where we refresh the pheromone trail of the best solution in the case that it would be preferable for the next constructions. The parameter ρ relates to pheromone evaporation. The value of ρ is fixed in the way that τ_{ij} will increase after the update.

2.3.3 Local Pheromone Trail Update

After constructing the tours, we let the program execute the flipping sequence to get the final cost. Then aside the global pheromone trail update, we apply local pheromone trail update to every tour:

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \tag{2.3}$$

where ξ is a constant and τ_0 is the initial pheromone for every arc. The parameter ξ is usually assigned as $\frac{1}{nC_m}$ where n is the length of a tour and C_m is the cost of the nearest-neighbor (NN) tour. In TSP, the NN tour is constructed by adding an unvisited city that has the shortest distance to the current entire tour. In our problem, it is built by taking FEs continuously with the minimum cost at each iteration as possible.

2.4 Adapt Ant Colony System for solving the thickness problem

Unlike in TSP the cost of a tour increases gradually when a city is added, C_{ij} might increase or decrease in this problem. Therefore, the tour should be trimmed in the way that the cost at the end of the tour is always smaller than any values in between. There would be a fixed parameter l_0 which constrains the edges to take in when constructing a tour. Although it is controversial to say that we are not able to know what part is good to be taken into account, we can extend l_0 to deal with that issue.

3. Data Structures

Because the code is written in C, not many data structures are available to use. we will introduce briefly some data structures and their usage in this section.

3.1 Triangulation

It is defined in triang.h as followed:

There are a few parameters in a triang struct: nv, ne, nf, orient, genus, the edge array and nfe. To create a triangulation, use the function *create_triang*(). After being constructed, the adjacency array includes the coordinate information of the triangulation in which an element in a row and a column are represented as a triangle formed by three adjacent vertices. For example, at the row 1 and column 2, there is a number 3; it indicates that the vertices 1, 2, 3 forms a triangle. The interesting point in the adjacency array is edge(1)(2) might be different from edge(2)(1). That helps us find FE much easier that you can read it in the latter part. There are several ways to print out a triangulation by using functions: *print_triang*(), *print_neighbors*2() and *triang_lex*().

3.2 Ants

If we use ACO, we need to work with ants. Ants are the main components of ACO to process the rule given by the programmer. Here are variables defined in an ant:

- The current length of the tour: It's necessary to know the length so that we can prune out some unnecessary parts of the tour (Read chapter 2.4 to see how to prune the tour).
- The number of triangles in the union complement: It is used for calculating the probability in the tour construction (Read chapter 2.1)
- A list to save all flippable edges at an iteration.

If you rewrite the code in an advanced programming language like Java or python, you might not need some parameters like the tour length because the languages have functions to get the length from a tour.

3.3 Pheromone τ

The pheromone for the entire graph is a 2D matrix in which a column or a row represents an endpoint of an edge. For example, $\tau[1][2] = 0.1$ means the pheromone available on edge 12 is 0.1.

4. Algorithms

4.1 Flowchart of the Main Program

The flowchart of the main program is shown in the Figure 4.1. It follows the theory that has been introduced in Chapter 2.2.2 where there are 3 main procedures that the program needs to evaluate. To begin with, a number of ants has been created to start the search. The initial amount of pheromone is also assigned to every path that an ant can reach. Theoretically, this amount should be relatively fair to the amount of an ant can add to the path after the visit because otherwise, if that gap is too big, it will take long time to see the convergence on the better set of edges for the next iterations. There should be a variable that tracks the best solution currently found by the program. The initial best solution is found by any ways you can possibly think about. ACO basically takes in the better solutions and optimizes the probability on every path. The main part of ACO is to keep building potential solutions, to calculate the cost of each solution and update the pheromones.

4.2 Counting Cliques

The main goal in this project is to count the number of cliques in the complement of the union of triangulations. First, the triangulations are generated by the function *create_triang*(). They will have an adjacency matrix for each. Then make the union of them by uniting the adjacency matrices. Because the union might not be planar, its matrix might be different from the matrices we receive from Sulanke's code. It contains only 0 or 1. Counting cliques is basically to make nested loops that check the relations between values of the matrix.

4.3 Calculate Initial Pheromone

This idea is similar to the way we do the nearest neighbors calculation in the Traveling Salesman problem. It is basically to keep flipping edges that gives us the

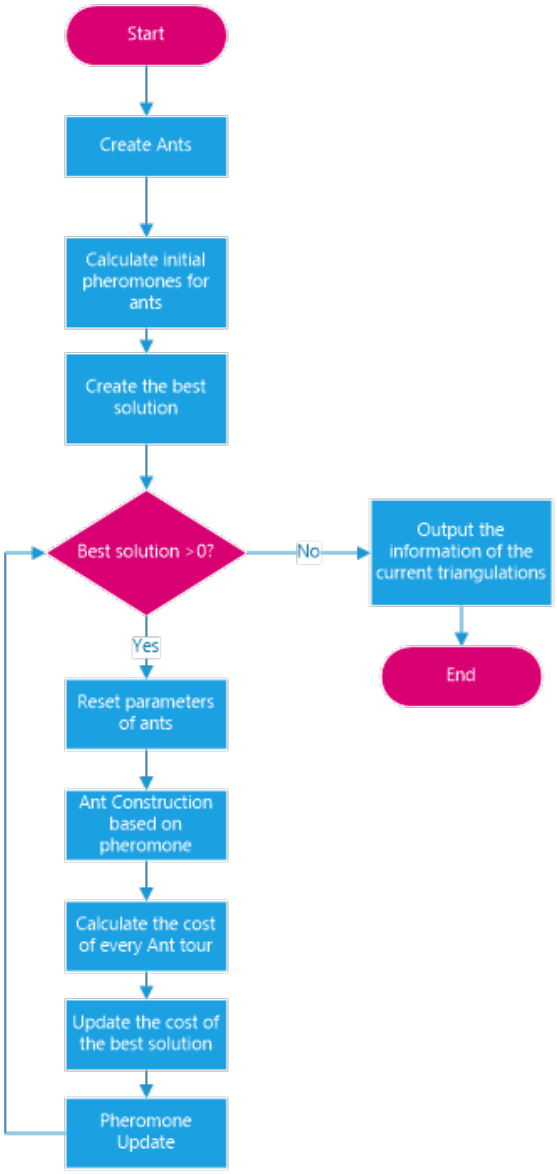


Figure 4.1: The flowchart of the program.

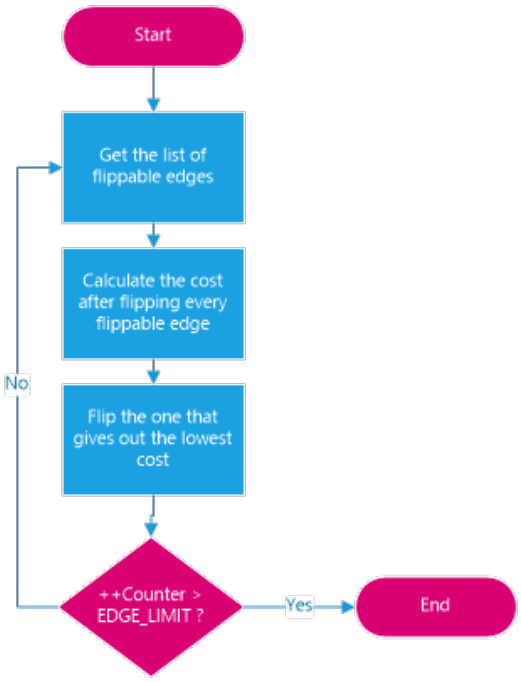


Figure 4.2: Calculate the initial pheromone.

lowest possible cost in the union. We need to retrieve the list of FE by the function $get\_flippable\_edges()$. Then calculate the cost after flipping every FE in the list by counting cliques in the union. Since there would have to be at least a FE gives out the minimum cost, flip it and keep the triangulations in the next step. We keep doing the same until we reach the number of FE. The output would be the cost of the final union. Figure 4.2 is the flowchart of this procedure.

4.4 Finding a Flippable Edge

Because Sulanke used a special triangulation adjacency matrix a, we can easily detect a flippable edge by checking two values of $a[i][j]$ and $a[j][i]$ where $0 \leq i, j < |V(G)|$. If these two values are different, flipping the edge in this area is doable. If they are the same, it is not available. Specifically, let's say $a[i][j] = m$ and $a[j][i] = n$. If $m \neq n$, it means there are 2 triangles (m, i, j) and (n, i, j) sharing ij as the diagonal. If $m = n$, there is one triangle $(m, i, j) \equiv (n, i, j)$.

4.5 Randomize the Triangulation Creation

It was recognized that the generating triangulation function written by Thom did not randomly create the triangulations. Therefore, we scramble the output from create_triang() and do a random number of flips.

4.6 Coloring Algorithms

Finding a graph G with the chromatic number $\chi(G) \geq 3$ is an NP-complete problem [GJ90]. The most intuitively simplest graph coloring algorithm that can be thought is brute-force search where all the assignments are tested with n vertices but the execution time is $O((n+1)!)$. It is not possible to use in practice, so people think about the better algorithms. Theoretically, we can categorize the algorithms into 2 types: exact coloring algorithm and approximation algorithms. An exact coloring algorithm guarantees to get a chromatic number of a given graph while an approximation algorithm finds accepted solutions, not the best. Approximation algorithms mostly use greedy approach, dynamic programming and heuristics to reduce the problem complexity. There are popular coloring algorithms that people have implemented: DSATUR [Bré79](Degree of Saturation), MCS [PP05](Register Allocation via Coloring of Chordal Graphs), lmXRLF (Efficient Coloring of a Large Spectrum of Graphs), TabuCol [HdW87](Tabu Search Techniques for Graph Coloring), MinimumVertexColoring [PS03]. In this project, we have used DSATUR and MVC for testing.

4.6.1 Degree of Saturation (DSATUR)

DSATUR [Bré79] is one of the popular greedy algorithms that used heuristic to finds an optimal coloring for a graph by greedily testing all the possible colorings. It is known as an exact coloring algorithm. In DSATUR, every vertex in a graph is ranked with a saturation degree as the number of different colors to which it is adjacent. A vertex of maximal saturation degree will be considered to color first

because it has the least colors to choose. So there would be an order of saturation degrees that DSATUR will follow and choose out a proper color but keep the number of used colors as least as possible.

Algorithm 1: DSATUR Algorithm

1 Sort the vertices to an descending order of saturation degrees.;
2 Color a vertex of maximal degree with color 1.;
3 Choose any uncolored vertex of maximal saturation degree.;
4 Color the chosen vertex with the lowest numbered color.;
5 Return to 3 if there are some remaining uncolored vertices. Otherwise, stop.

4.6.2 Minimum Vertex Coloring

Minimum Vertex Coloring (MVC) is a graph coloring function in Mathematica that uses backtracking to get a vertex coloring of a given graph with the fewest colors. Its implementation based on a set of starting some ordered configurations in which a prefix might be a part of a solution. Backtracking starts with the smallest possible configuration and add elements to it until a complete solution is found. Otherwise, replace the last element in the configuration by the next possibility. For more details, please read it at [PS03].

4.6.3 The comparison between DSATUR and MVC

By the definitions above, DSATUR looks very similar to how MVC expands out from its smallest configuration. Mathematica has implemented both algorithms as MinimumVertexColoring and BrelazColoring but their execution time when finding the chromatic number of my graphs are not good enough. Specifically, BrelazColoring (DSATUR) in Mathematica can find a good, but not necessarily minimal, edge or vertex coloring for a graph [Bre]. On the other hand, MVC takes too long to find a chromatic number of an union of triangulations with 19 vertices within 5-6 hours. So it's not possible to use MVC to prune out graphs although MVC might be trustworthy.

We need to find another reliable alternative written in C and we found one [DSA]. This is also the code that Sulanke used to prune out his graphs.

5. Experiments and Results

From the inequality 1.1, we can figure out which values we are supposed to test for the problem. Table 5.1 shows some testing cases to be considered.

Thickness	Genus	Upperbound of χ	Clique	Number of vertices for testing
1	1	7	3	7 9 11 13 15
1	1	7	4	10 13 16 19 22
1	1	7	5	13 17 21 25 29
1	2	8	3	9 11 13 15 17
1	2	8	4	13 16 19 22 25
1	2	8	5	17 21 25 29 33
1	3	9	3	11 13 15 17 19
1	3	9	4	16 19 22 25 28
1	3	9	5	21 25 29 33 37
2	0	12	3	17 19 21 23 25
2	0	12	4	25 28 31 34 37
2	0	12	5	33 37 41 45 49
2	1	13	3	19 21 23 25 27
2	1	13	4	28 31 34 37 40
2	1	13	5	37 41 45 49 53
2	2	13	3	19 21 23 25 27
2	2	13	4	28 31 34 37 40
2	2	13	5	37 41 45 49 53
2	3	14	3	21 23 25 27 29
2	3	14	4	31 34 37 40 43
2	3	14	5	41 45 49 53 57
3	0	18	3	29 31 33 35 37

3	0	18	4	43 46 49 52 55
3	0	18	5	57 61 65 69 73
3	1	19	3	31 33 35 37 39
3	1	19	4	46 49 52 55 58
3	1	19	5	61 65 69 73 77
3	2	19	3	31 33 35 37 39
3	2	19	4	46 49 52 55 58
3	2	19	5	61 65 69 73 77
3	3	20	3	33 35 37 39 41
3	3	20	4	49 52 55 58 61
3	3	20	5	65 69 73 77 81

Table 5.1: Some testing cases from the inequality 1.1

From table 5.2, there are 9 columns: NV (the number of vertices), t (thickness), g (genus), Orient? (Orientable), Km-free (the cliques to count), χ, Upper bound of χ, $|K_m|$ start (the number of cliques from the union before optimizing), $|K_m|$ end (the lowest K_m we could reach after a period of time). It was unsuccessful for me to reach any solutions with the upper bound of the chromatic number in Equation (1.1).

All the tests were ran on my personal computer. Most of them are found in seconds to minutes. The unsuccessful findings were stopped after a number of iterations.

Table 5.2: The results of graph thickness 2 from testing ACO algorithm

NV	t	g	Orient?	Km-free	χ	Upperbound of χ	$\lvert K_m \rvert$ start	$\lvert K_m \rvert$ end
17	2	0	0	3	9	12	6	0
19	2	0	0	3	10	12	48	6
19	2	1	0	3	10	13	18	0
19	2	1	1	3	10	13	11	0
21	2	1	0	3	11	13	55	10
21	2	1	1	3	11	13	49	0
23	2	1	0	3	12	13	129	47
23	2	1	1	3	12	13	124	17
19	2	2	0	3	10	13	15	0
19	2	2	1	3	10	13	7	0
21	2	2	0	3	11	13	44	0
21	2	2	1	3	11	13	26	0
23	2	2	1	3	12	13	57	0
25	2	2	1	3	13	13	157	80
21	2	3	1	3	11	14	13	0
23	2	3	1	3	12	14	34	0
25	2	3	1	3	13	14	115	4
27	2	3	1	3	14	14	464	199
25	2	4	1	3	13	15	271	0
25	2	0	0	4	9	12	61	0

6. Conclusion

We successfully adapted ACS theory for solving the thickness problem. Although we did not get any solution that reached the upper bound from inequality 1.1 given, my algorithm can seek solutions in seconds. What we have done in this project is to further clarify the steps about how to make ACS work for the problem. My implementation is all based on the original theory of ACS, so it might not be fully optimized for the problem. Although it is not developed in advanced, we do believe that this thesis will be a stepping stone as a reference. Roqyah did a fantastic work in documenting what she's done so far on this topic [Ala07]. We highly recommend her thesis before involving to this project.

APPENDIX A. Functions

Since we have already categorized the files in folders, it is easier to manipulate the code and change.

A.1 ant_colony

A.1.1 ant_colony.c

ant_colony_system() is the main function of Ant Colony which might be started directly from arguments given in the command or manually ran after the user loaded the parameters. The function keeps running until a solution is found. Then it will reset the ants and do a new search.

A.1.2 ant_colony_system_initialization.c

- reset_ant: reset an Ants parameters to the default generated triangulations t[i] and make the union by merge_triangs2()
- reset_tau: reset the pheromone matrix to the default values
- initialize_best_solution: create/reset a best solution ever found by ACO
- ant_colony_system_save_selection: Save the selection. Basically, when an index in the FE array is chosen, it will flip the triangulation containing that FE, remake the union again and save the FE
- print_the_tour_cost
- print_tau

A.1.3 ant_colony_system_pheromone_update.c:

- local_pheromone_update

- glocal_pheromone_update

A.1.4 ant_colony_system_stages.c

- RouletteWheelSelection: Apply Roulette Wheel Selection algorithm (RWS) for choosing out an FE. There is a parameter dupTimes that eliminates loops. For example, FE with the endpoints 12 and 34 keeps being picked by RWS but actually they are in the same quadrilateral. It means there is not much change in our triangulations fundamentally. dupTimes will remove the option that duplicates many times in an Ant tour.

- ant_colony_system_tour_construction: This function is actually the main core of ACO. First, it will get the list of available FE in all layers. The list has a variable to remember what layer a FE belongs to, so it's pretty much convenient for the calculation after. Then it calculates the cost of each FE by getting the inverse of the remaining triangles in the union complement. The inverse is a must because the lower triangles we have, the higher possibility we can reach the goal. It's also applied for Traveling Salesman problem in which the lower distance for a tour is preferable.

- ant_colony_system_selection: there is a percentage to choose an option without pheromone impact. That avoids biased choices created by pheromone after a long time. Read ACS for further information.

- prune_tours: We have an idea that the tour should be pruned out to get the part of it in which the cost should be gradually increased or decreased in order. Since we only care about the last cost of the tour, pruning might not affect too much about the way we choose a FE but it is good in many ways if we don't count unnecessary paths into a tour.

A.1.5 ant_structs.c

- init_ant: allocate memory to an Ant
- free_ant: free the memory
- free_solution: free the memory

A.1.6 calculate_initial_pheromone.c

We calculate the initial pheromone in the way that we keep blindly choosing the FE that yields the lowest possible remaning triangles in the union complement. It's similar to Nearest Neighbors (NN) in TSP; that is to keep traveling to the nearest cities. Apparently, it's not the best in theory and in experiments we've done. On the other hand, NN is the case when pheromone weight isn't counted in ACS.

A.1.7 display_parameters.c

Display the arguments inserted in the command.

A.1.8 free_arrays

Free memory from some specific array pointers.

A.1.9 initialize_ant_parameters

Parses the arguments to initialize ant parameters.

A.1.10 initialize_arrays

This function helps to initialize ants and calculate initial pheromone.

A.1.11 Triang_lex

Lex format was defined by Thom. It is used to load a triangulation in his format.

A.2 Coloring

A.2.1 Dharwadker_coloring

This is the open library for vertex coloring that roqyah [Ala07] used for her testing. It's available at the link `http://www.dharwadker.org/vertex_coloring/` . We change it a little bit to integrate it into our code.

A.2.2 dsatur_coloring.c

A.2.3 get_critical_graph.c

The main function in this file is get_critical_graph() in which it will prune out vertices and edges until the resulting graph is critical.

A.3 includes

A.3.1 triang.h

A.3.2 triang_utils.h

A.4 triang

A.5 triang_utils

A.5.1 count_cliques.c

This function is used for counting the cliques in a graph. It is only able to count K_3, K_4 and K_5.

A.5.2 FlipEdge.c

It helps to flip an edge on a specific triangulation. Because of the advantage of the triangulation creation, by checking 2 elements in edge[i][j], if they are different, so there is an available FE; otherwise, if they are the same or not defined, it means they both present the relations in a triangle so that it's not possible to flip.

A.5.3 generate_random_triang.c

The outcome from create_triang() is not random. In other words, you will get the same list of triangulation after calling the function a few times. That's why we have to scramble that outcome and randomly flip 50 Fes. All of the idea is adapted from Thom.

A.5.4 get_complement_graph.c

It outputs the complement of the adjacency matrix by toggling the values of the elements.

A.5.5 get_flippable_edges.c

All the Fes will be found by find_flip() and stored into the Ant struct.

A.5.6 Get_int.c

Get the integer from the console.

A.5.7 load_default_triangles.c

Sometimes it needs to take very long time to possibly get a solution, we need to save out the entire running database in the program so that we can continue the finding from the save points.

A.5.8 log_function.c

initialize_log_function(): The log file will be created after the main function is called. It will be stored in the folder Log. The name of the file will contain the date information so that solutions in a day will be in the same file.

A.5.9 merge_2_triang.c

- merge_triangs(): The input of this function is a struct array of t[] which is defined globally.
- merge_triangs2(): The input of this function is the struct array of an Ant after being loaded from the global.
- merge_triangs3():

A.5.10 permute_triang.c

interchange two vertices of a triangulation.

A.5.11 print_neighbors2.c

print out the neighbor list of a vertex.

A.5.12 print_triang.c

print the adjacency matrix under a table.

A.5.13 ran_flip_triang.c

flip random Fes a number of times

A.5.14 scramble_triang.c

randomly permute all the vertices of a triangulation

A.5.15 set_undirected_edge.c

change the values of an adjacency matrix. It might not work on the adjacency matrix of the triangulation generated by create_triang().

APPENDIX B. Compiling the code

We have written a README file to guide you how to compile and execute the code. Basically, because the code is written on C, we use gcc to compile the code. A Makefile was created in the root folder to simplify the compiling process. You typically need to go to the root folder and run 'make'. Then, execute the ant_colony by the following command: ./ant_colony [load_default_triangs?] [File Path] [nv] [genus] [orientable?] [thickness] [vertices_in_cliques] [alpha] [beta] [rho] [number of Ants] [EDGE_LIMIT] [Qo] [Xi] [want to print every tour] [want to print tau] [number of iterations]

Here are the details for the parameters:

- [load_default_triangs?] : This is to recognize that you want to reload the some previous data or not. VALUE should be 0 or 1. If you choose 0, you don't have to define path (or leave it any value you want) but [nv], [genus] and [orientable?] must be specified. EXAMPLE VALUE: 0

- [File Path]: the path to the file you want to load in the Data folder at root directory. The file consists of 2 triangulations in lex format that is what Sulanke's loading function needs. EXAMPLE VALUE: 17_93 - Load the file 17_93.txt in the Data folder.

- [nv]: number of vertices to generate in case that [load_default_triangs?] = 0. If [load_default_triangs?] = 1, leave it as any value. EXAMPLE VALUE: 17

- [genus]: number of genus in a triangulation you want to generate in case that [load_default_triangs?] = 0. If [load_default_triangs?] = 1, leave it as any value. EXAMPLE VALUE: 0

- [orientable?]: the surface is orientable or not in case that [load_default_triangs?] = 0. If [load_default_triangs?] = 1, leave it as any value. EXAMPLE VALUE: 0

- [thickness]: the number of planar graphs you try to get from decomposing the original graph.

- [vertices_in_cliques]: number of vertices in the cliques you want to count in the complement of the union. EXAMPLE VALUE: 3

- [alpha], [beta], [rho]: these are 3 parameters alpha, beta and pheromone evaporation needed for ant_colony. EXAMPLE VALUE: 1 30 0.01

- [number of Ants]: Number of ants to run. EXAMPLE VALUE: 50

- [EDGE_LIMIT]: The maximum number of edges you can flip for an ant. DEFAULT VALUE is nv*(nv-1)*50. If you leave 0, the application will load the DEFAULT VALUE.

- [Qo], [Xi]: 2 parameters for Ant Colony System. EXAMPLE VALUE: 0.1 0.1

- [want to print every tour]: The value is 0 or 1. Default value is 0. If you set 1, the tour of every ant will be printed out in every iteration.

- [want to print tau]: Its value is 0 or 1. If you set 1, the tau or pheromone of all edges will be printed out at the end of every iteration.

- [number of iterations]: Number of iterations for the execution.

My MATLAB code has a bug which generates lower or equal to the chromatic number found in my ACO application. This bug might happen when I prune out the graph or from the coloring algorithm DSATUR that I used from the other.

APPENDIX C. Plot graphs from the results collected from Sulanke's application

The outcome from the modified Sulanke's application is the neighbors list of each triangulation found for a solution. The vertices in each list starts at 0 so that we might need to increase the sequence numbers by 1 because some visualization tools like Mathematica requires the first vertex as 1. We programmed a MATLAB code to read all the output and generate some codes to paste into Mathematica. For the details, please find the comments in the code. On the other hand, we used 2 algorithms to find the chromatic number of graph: minimum vertex coloring (mvc) in Mathematica and DSATUR algorithm. It seems like mvc takes too long to get a result so that DSATUR algorithm is the one we used the most. We actually write a verification code in the MATLAB code to verify the color list got from DSATUR. The DSATUR code is written in C so that it is already implemented in Sulanke's code.

APPENDIX D. Some examples of solutions we have found in chapter 5

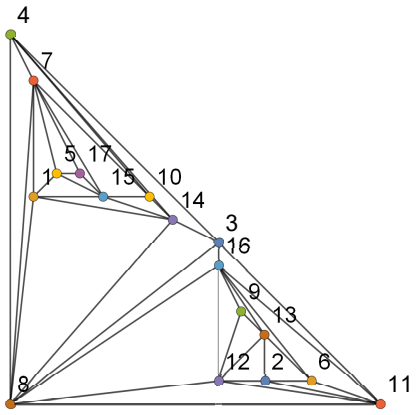
The following data is not verified, therefore please examine it by hand when you intend to use them as reference. Those data without any embedding are considered as conjectures.

Table D.1: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 2,1,1,3,8,2,4,6,3,8,4,5,6,5,7,7,9

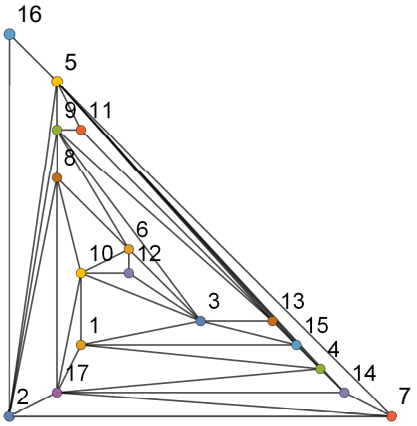
1st Embedding

Triang 1	Triang 2
1: 5 7 8 14 15	1: 3 4 10 15 17
2: 6 11 12 13	2: 5 7 8 9 16 17
3: 4 8 11 14 16	3: 1 6 9 10 12 13 15
4: 3 7 8 10 14	4: 1 5 14 15 17
5: 1 7 15 17	5: 2 4 7 9 11 13 14 15 16
6: 2 11 13 16	6: 3 8 9 10 12
7: 1 4 5 8 10 15 17	7: 2 5 14 17
8: 1 3 4 7 11 12 14 16	8: 2 6 9 10 17
9: 12 13 16	9: 2 3 5 6 8 11 13
10: 4 7 14 15	10: 1 3 6 8 12 17
11: 2 3 6 8 12 16	11: 5 9 13
12: 2 8 9 11 13 16	12: 3 6 10
13: 2 6 9 12 16	13: 3 5 9 11 15
14: 1 3 4 8 10 15	14: 4 5 7 17
15: 1 5 7 10 14 17	15: 1 3 4 5 13
16: 3 6 8 9 11 12 13	16: 2 5
17: 5 7 15	17: 1 2 4 7 8 10 14



Color List: 2,1,1,3,8,2,4,6,3,8,4,5,6,5,7,7,9

Figure D.1: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 2,1,1,3,8,2,4,6,3,8,4,5,6,5,7,7,9

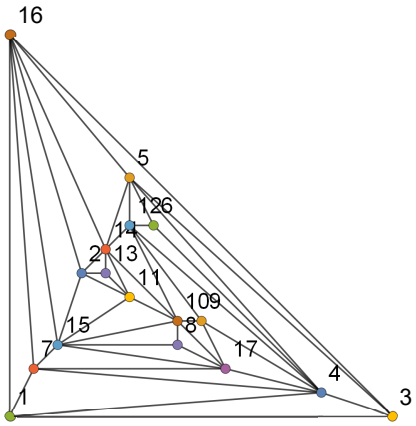
Figure D.2: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.2: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 3,1,8,1,2,3,4,5,2,6,8,7,5,4,7,6,9

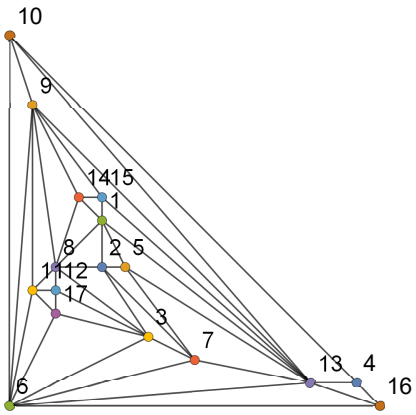
2^{nd} Embedding

Triang 1	Triang 2
1: 3 4 7 16	1: 2 5 8 13 14 15
2: 11 13 14 15 16	2: 1 3 5 7 8
3: 1 4 5 16	3: 2 6 7 8 12 17
4: 1 3 5 6 7 9 12 17	4: 10 13 16
5: 3 4 6 12 14 16	5: 1 2 7 13
6: 4 5 12	6: 3 7 9 10 11 13 16 17
7: 1 4 15 16 17	7: 2 3 5 6 13
8: 10 15 17	8: 1 2 3 9 11 12 14
9: 4 10 12 17	9: 6 8 10 11 13 14 15
10: 8 9 11 12 14 15 17	10: 4 6 9 13
11: 2 10 13 14 15	11: 6 8 9 12 17
12: 4 5 6 9 10 14	12: 3 8 11 17
13: 2 11 14	13: 1 4 5 6 7 9 10 15 16
14: 2 5 10 11 12 13 16	14: 1 8 9 15
15: 2 7 8 10 11 16 17	15: 1 9 13 14
16: 1 2 3 5 7 14 15	16: 4 6 13
17: 4 7 8 9 10 15	17: 3 6 11 12



Color List: 3,1,8,1,2,3,4,5,2,6,8,7,5,4,7,6,9

Figure D.3: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 3,1,8,1,2,3,4,5,2,6,8,7,5,4,7,6,9

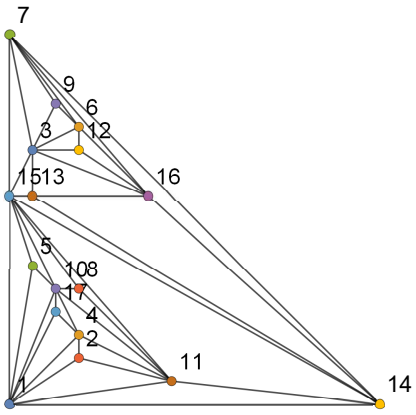
Figure D.4: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.3: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,4,1,2,3,2,3,4,5,5,6,8,6,8,7,9,7

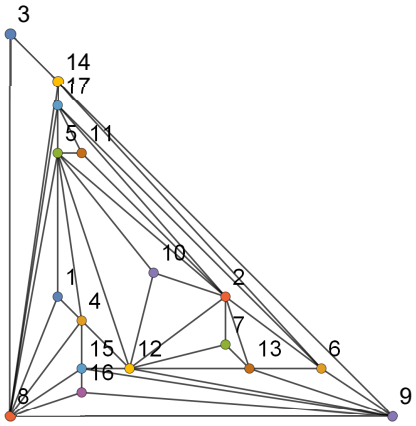
3^{rd} Embedding

Triang 1	Triang 2
1: 2 4 5 10 11 14 15 17	1: 4 5 8
2: 1 4 11	2: 5 6 7 10 11 12 13 17
3: 6 7 9 12 13 15 16	3: 8 14
4: 1 2 10 11 17	4: 1 5 8 12 15
5: 1 10 15	5: 1 2 4 8 10 11 12 17
6: 3 7 9 12 16	6: 2 9 13 14 17
7: 3 6 9 14 15 16	7: 2 12 13
8: 10 11 15	8: 1 3 4 5 9 14 15 16 17
9: 3 6 7	9: 6 8 12 13 14 15 16
10: 1 4 5 8 11 15 17	10: 2 5 12
11: 1 2 4 8 10 14 15	11: 2 5 17
12: 3 6 16	12: 2 4 5 7 9 10 13 15
13: 3 14 15 16	13: 2 6 7 9 12
14: 1 7 11 13 15 16	14: 3 6 8 9 17
15: 1 3 5 7 8 10 11 13 14	15: 4 8 9 12 16
16: 3 6 7 12 13 14	16: 8 9 15
17: 1 4 10	17: 2 5 6 8 11 14



Color List: 1,4,1,2,3,2,3,4,5,5,6,8,6,8,7,9,7

Figure D.5: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,4,1,2,3,2,3,4,5,5,6,8,6,8,7,9,7

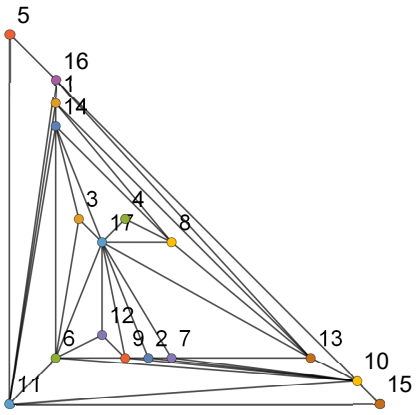
Figure D.6: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.4: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 2,1,2,3,4,3,5,8,4,8,7,5,6,1,6,9,7

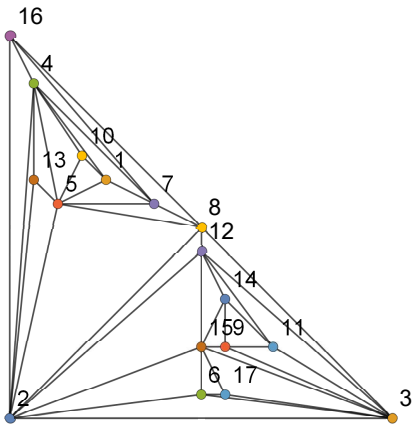
4^{th} Embedding

Triang 1	Triang 2
1: 8 11 13 14 16	1: 4 5 7 10
2: 7 9 10 17	2: 3 4 5 6 8 12 13 15 16
3: 6 14 17	3: 2 6 8 9 11 12 15 17
4: 8 17	4: 1 2 5 7 10 13 16
5: 11 16	5: 1 2 4 7 8 10 13
6: 3 9 10 11 12 14 17	6: 2 3 15 17
7: 2 10 13 17	7: 1 4 5 8 16
8: 1 4 13 14 17	8: 2 3 5 7 12 16
9: 2 6 10 12 17	9: 3 11 14 15
10: 2 6 7 9 11 13 15 16	10: 1 4 5
11: 1 5 6 10 14 15 16	11: 3 9 12 14
12: 6 9 17	12: 2 3 8 11 14 15
13: 1 7 8 10 16 17	13: 2 4 5
14: 1 3 6 8 11 17	14: 9 11 12 15
15: 10 11	15: 2 3 6 9 12 14 17
16: 1 5 10 11 13	16: 2 4 7 8
17: 2 3 4 6 7 8 9 12 13 14	17: 3 6 15



Color List: 2,1,2,3,4,3,5,8,4,8,7,5,6,1,6,9,7

Figure D.7: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 2,1,2,3,4,3,5,8,4,8,7,5,6,1,6,9,7

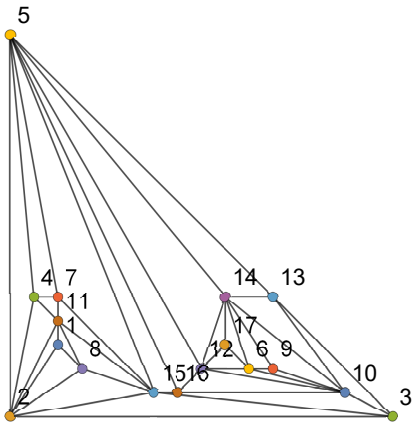
Figure D.8: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.5: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,2,3,3,8,8,4,5,4,1,6,5,7,9,7,6,2

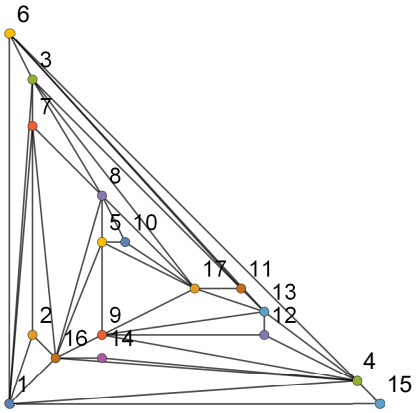
5^{th} Embedding

Triang 1	Triang 2
1: 2 8 11	1: 2 3 4 6 7 15 16
2: 1 3 4 5 8 11 15	2: 1 7 16
3: 2 10 13 15	3: 1 6 7 8 11 17
4: 2 5 7 11	4: 1 6 9 12 13 14 15 16
5: 2 4 7 12 13 14 15 16	5: 8 9 10 16 17
6: 9 10 12 14 17	6: 1 3 4 11 13
7: 4 5 11 15	7: 1 2 3 8 16
8: 1 2 11 15	8: 3 5 7 10 16 17
9: 6 10 14	9: 4 5 12 13 16 17
10: 3 6 9 12 13 14 16	10: 5 8 17
11: 1 2 4 7 8 15	11: 3 6 13 17
12: 5 6 10 14 16 17	12: 4 9 13
13: 3 5 10 14	13: 4 6 9 11 12 17
14: 5 6 9 10 12 13 17	14: 4 16
15: 2 3 5 7 8 11 16	15: 1 4
16: 5 10 12 15	16: 1 2 4 5 7 8 9 14
17: 6 12 14	17: 3 5 8 9 10 11 13



Color List: 1,2,3,3,8,8,4,5,4,1,6,5,7,9,7,6,2

Figure D.9: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,2,3,3,8,8,4,5,4,1,6,5,7,9,7,6,2

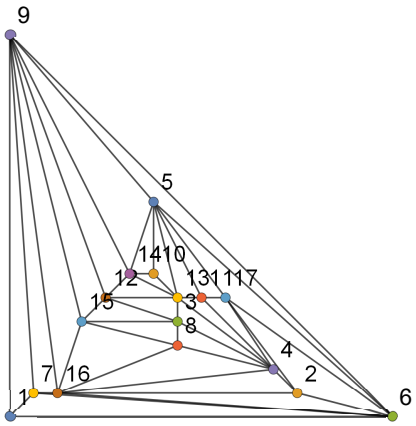
Figure D.10: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.6: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,2,3,5,1,3,8,4,5,2,4,6,8,9,7,6,7

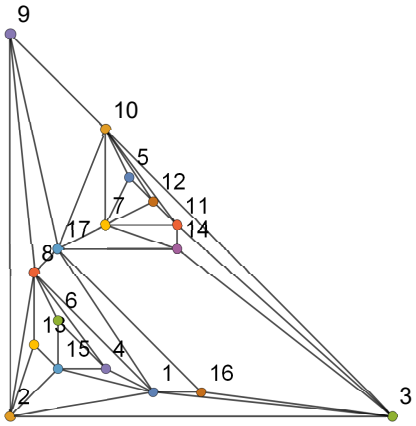
6^{th} Embedding

Triang 1	Triang 2
1: 6 7 9	1: 2 3 4 8 15 16 17
2: 4 6 16 17	2: 1 3 8 9 13 15
3: 4 8 12 13 15	3: 1 2 10 11 14 16
4: 2 3 8 11 13 16 17	4: 1 6 8 15
5: 6 9 10 11 13 14 17	5: 7 10 12
6: 1 2 5 7 9 16 17	6: 4 8 15
7: 1 6 9 16	7: 5 10 11 12 14 17
8: 3 4 15 16	8: 1 2 4 6 9 13 17
9: 1 5 6 7 12 14 15 16	9: 2 8 10 17
10: 5 13 14	10: 3 5 7 9 11 12 17
11: 4 5 13 17	11: 3 7 10 12 14
12: 3 9 13 14 15	12: 5 7 10 11
13: 3 4 5 10 11 12 14	13: 2 8 15
14: 5 9 10 12 13	14: 3 7 11 17
15: 3 8 9 12 16	15: 1 2 4 6 13
16: 2 4 6 7 8 9 15	16: 1 3 17
17: 2 4 5 6 11	17: 1 7 8 9 10 14 16



Color List: 1,2,3,5,1,3,8,4,5,2,4,6,8,9,7,6,7

Figure D.11: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,2,3,5,1,3,8,4,5,2,4,6,8,9,7,6,7

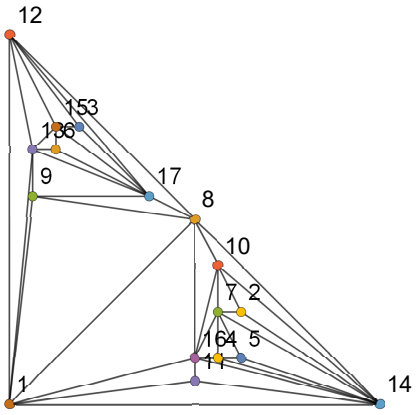
Figure D.12: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.7: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 6,8,1,8,1,2,3,2,3,4,5,4,5,7,6,9,7

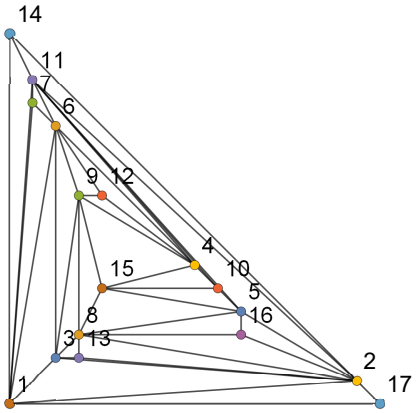
7^{th} Embedding

Triang 1	Triang 2
1: 8 9 11 12 13 14 16	1: 2 3 6 7 11 14 17
2: 7 10 14	2: 1 3 5 8 11 13 14 16 17
3: 12 15 17	3: 1 2 6 8 9 13
4: 5 7 14 16	4: 6 9 10 11 12 15
5: 4 7 14	5: 2 8 10 11 15 16
6: 13 15 17	6: 1 3 4 7 9 11 12
7: 2 4 5 10 14 16	7: 1 6 11
8: 1 9 10 12 14 16 17	8: 2 3 5 9 13 15 16
9: 1 8 13 17	9: 3 4 6 8 12 15
10: 2 7 8 14 16	10: 4 5 11 15
11: 1 14 16	11: 1 2 4 5 6 7 10 14
12: 1 3 8 13 15 17	12: 4 6 9
13: 1 6 9 12 15 17	13: 2 3 8
14: 1 2 4 5 7 8 10 11 16	14: 1 2 11
15: 3 6 12 13 17	15: 4 5 8 9 10
16: 1 4 7 8 10 11 14	16: 2 5 8
17: 3 6 8 9 12 13 15	17: 1 2



Color List: 6,8,1,8,1,2,3,2,3,4,5,4,5,7,6,9,7

Figure D.13: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 6,8,1,8,1,2,3,2,3,4,5,4,5,7,6,9,7

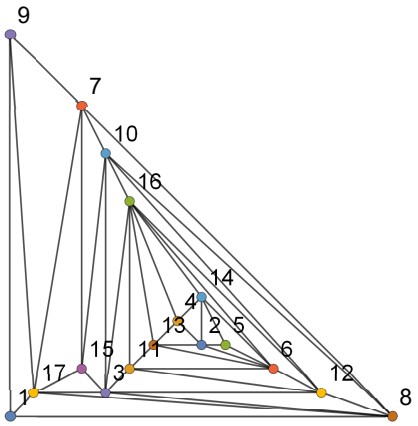
Figure D.14: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.8: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,1,5,2,3,4,4,6,5,7,2,8,6,7,9,3,8

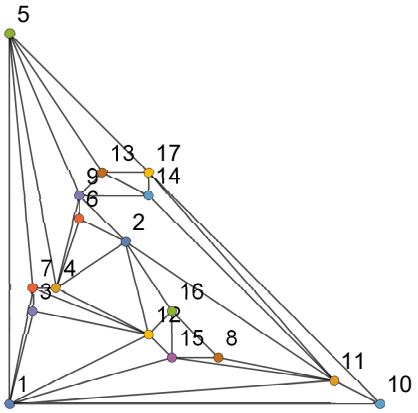
8^{th} Embedding

Triang 1	Triang 2
1: 8 9 17	1: 3 5 7 10 11 12 15
2: 4 5 6 13 14	2: 4 6 9 11 12 16
3: 8 10 11 12 15 16 17	3: 1 7 12
4: 2 13 14 16	4: 2 5 6 7 9 12
5: 2 6 14	5: 1 4 7 9 13 17
6: 2 5 11 12 13 14 16	6: 2 4 9
7: 8 9 10 15 17	7: 1 3 4 5 12
8: 1 3 7 10 12 17	8: 11 15 16
9: 1 7 17	9: 2 4 5 6 13 14
10: 3 7 8 12 15 16	10: 1 11 17
11: 3 6 12 13 16	11: 1 2 8 10 14 15 17
12: 3 6 8 10 11 16	12: 1 2 3 4 7 15 16
13: 2 4 6 11 16	13: 5 9 14 17
14: 2 4 5 6 16	14: 9 11 13 17
15: 3 7 10 17	15: 1 8 11 12 16
16: 3 4 6 10 11 12 13 14	16: 2 8 12 15
17: 1 3 7 8 9 15	17: 5 10 11 13 14



Color List: 1,1,5,2,3,4,4,6,5,7,2,8,6,7,9,3,8

Figure D.15: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,1,5,2,3,4,4,6,5,7,2,8,6,7,9,3,8

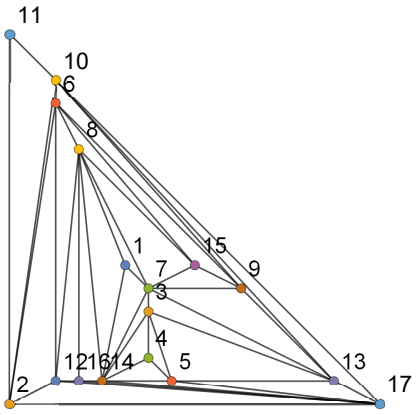
Figure D.16: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.9: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,2,2,3,4,4,3,8,6,8,7,1,5,6,9,5,7

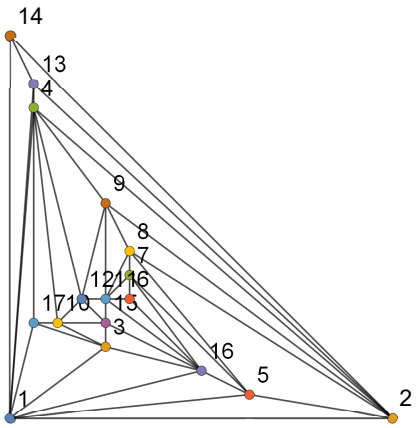
9^{th} Embedding

Triang 1	Triang 2
1: 7 8 14	1: 2 3 4 5 13 14 16 17
2: 6 10 11 12 17	2: 1 4 5 8 9 13 14
3: 4 5 7 13 14	3: 1 10 15 16 17
4: 3 5 14	4: 1 2 9 10 12 13 17
5: 3 4 13 14 17	5: 1 2 7 8 16
6: 2 8 9 10 12 15	6: 7 11 16
7: 1 3 8 9 13 14 15	7: 5 6 8 11 16
8: 1 6 7 12 14 15 16	8: 2 5 7 9 11
9: 6 7 10 13 15	9: 2 4 8 11 12
10: 2 6 9 11 13 17	10: 3 4 12 15 17
11: 2 10	11: 6 7 8 9 12 15 16
12: 2 6 8 16 17	12: 4 9 10 11 15
13: 3 5 7 9 10 17	13: 1 2 4 14
14: 1 3 4 5 7 8 16 17	14: 1 2 13
15: 6 7 8 9	15: 3 10 11 12 16
16: 8 12 14 17	16: 1 3 5 6 7 11 15
17: 2 5 10 12 13 14 16	17: 1 3 4 10



Color List: 1,2,2,3,4,4,3,8,6,8,7,1,5,6,9,5,7

Figure D.17: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,2,2,3,4,4,3,8,6,8,7,1,5,6,9,5,7

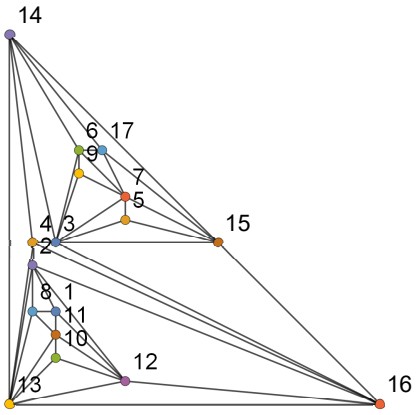
Figure D.18: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.10: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,5,1,2,2,3,4,7,8,3,6,9,8,5,6,4,7

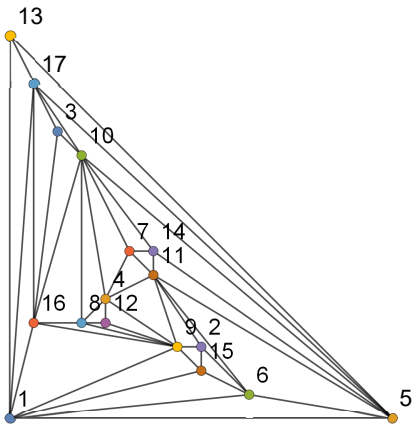
10^{th} Embedding

Triang 1	Triang 2
1: 2 8 11 12	1: 5 6 9 13 15 16 17
2: 1 4 8 12 13 16	2: 6 9 11 15
3: 4 5 6 7 9 14 15 16	3: 10 16 17
4: 2 3 13 14 16	4: 7 8 9 10 11 12
5: 3 7 15	5: 1 6 10 11 13 14 17
6: 3 7 9 14 17	6: 1 2 5 11 15
7: 3 5 6 9 15 17	7: 4 10 11 14
8: 1 2 11 13	8: 4 9 10 12 16
9: 3 6 7	9: 1 2 4 8 11 12 15 16
10: 11 12 13	10: 3 4 5 7 8 14 16 17
11: 1 8 10 12 13	11: 2 4 5 6 7 9 14
12: 1 2 10 11 13 16	12: 4 8 9
13: 2 4 8 10 11 12 14 16	13: 1 5 17
14: 3 4 6 13 15 17	14: 5 7 10 11
15: 3 5 7 14 16 17	15: 1 2 6 9
16: 2 3 4 12 13 15	16: 1 3 8 9 10 17
17: 6 7 14 15	17: 1 3 5 10 13 16



Color List: 1,5,1,2,2,3,4,7,8,3,6,9,8,5,6,4,7

Figure D.19: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,5,1,2,2,3,4,7,8,3,6,9,8,5,6,4,7

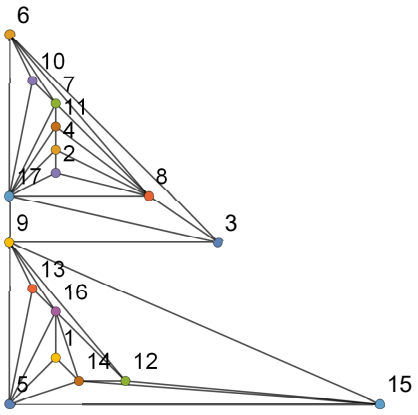
Figure D.20: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.11: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 8,5,1,2,1,2,3,4,8,5,6,3,4,6,7,9,7

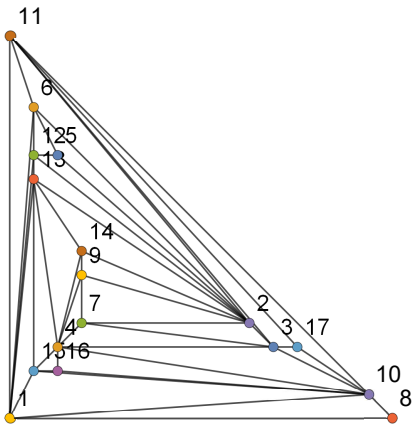
11^{th} Embedding

Triang 1	Triang 2
1: 5 14 16	1: 6 8 10 11 12 13 15
2: 4 8 17	2: 3 5 6 7 9 11 12 13 14
3: 6 8 9 17	3: 2 4 7 10 11 17
4: 2 8 11 17	4: 3 7 9 10 13 14 15 16
5: 1 9 13 14 15 16	5: 2 6 12
6: 3 7 8 10 17	6: 1 2 5 11 12
7: 6 8 10 11 17	7: 2 3 4 9
8: 2 3 4 6 7 11 17	8: 1 10
9: 3 5 12 13 15 16 17	9: 2 4 7 14
10: 6 7 17	10: 1 3 4 8 11 15 16 17
11: 4 7 8 17	11: 1 2 3 6 10 17
12: 9 14 15 16	12: 1 2 5 6 13
13: 5 9 16	13: 1 2 4 12 14 15
14: 1 5 12 15 16	14: 2 4 9 13
15: 5 9 12 14	15: 1 4 10 13 16
16: 1 5 9 12 13 14	16: 4 10 15
17: 2 3 4 6 7 8 9 10 11	17: 3 10 11



Color List: 8,5,1,2,1,2,3,4,8,5,6,3,4,6,7,9,7

Figure D.21: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 8,5,1,2,1,2,3,4,8,5,6,3,4,6,7,9,7

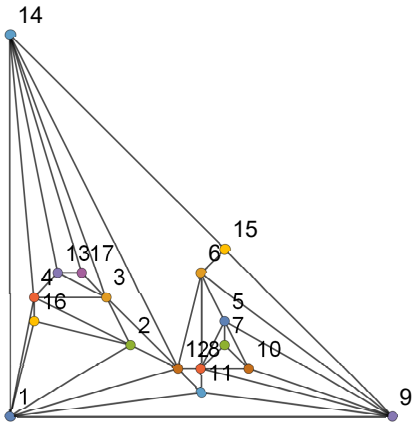
Figure D.22: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.12: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,3,2,4,1,2,3,4,5,6,7,6,5,7,8,8,9

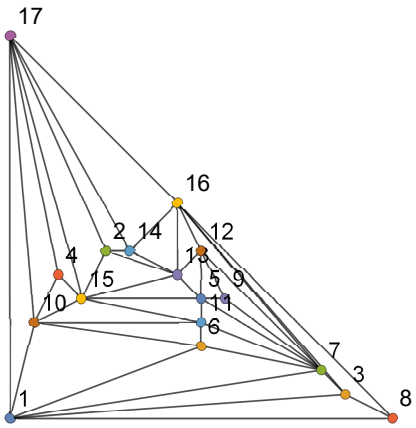
12^{th} Embedding

Triang 1	Triang 2
1: 2 4 9 11 12 14 16	1: 3 6 7 8 10 17
2: 1 3 4 12 16	2: 13 14 15 17
3: 2 4 12 13 14 17	3: 1 7 8 16
4: 1 2 3 13 14 16	4: 10 15 17
5: 6 7 8 9 10	5: 7 9 11 12 13 15
6: 5 8 9 12 15	6: 1 7 10 11
7: 5 8 10	7: 1 3 5 6 9 11 12 16
8: 5 6 7 9 10 11 12	8: 1 3 16
9: 1 5 6 8 10 11 15	9: 5 7 12
10: 5 7 8 9	10: 1 4 6 11 15 17
11: 1 8 9 12	11: 5 6 7 10 15
12: 1 2 3 6 8 11 14	12: 5 7 9 13 16
13: 3 4 14 17	13: 2 5 12 14 15 16
14: 1 3 4 12 13 15 17	14: 2 13 16 17
15: 6 9 14	15: 2 4 5 10 11 13 17
16: 1 2 4	16: 3 7 8 12 13 14 17
17: 3 13 14	17: 1 2 4 10 14 15 16



Color List: 1,3,2,4,1,2,3,4,5,6,7,6,5,7,8,8,9

Figure D.23: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,3,2,4,1,2,3,4,5,6,7,6,5,7,8,8,9

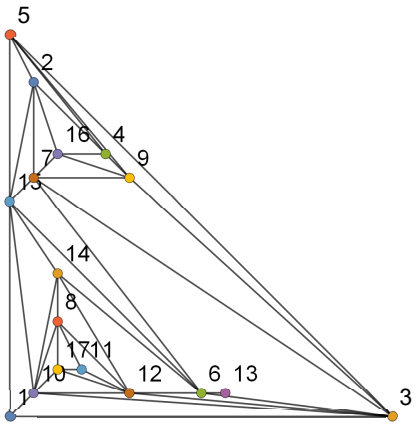
Figure D.24: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.13: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,1,2,3,4,3,6,4,8,5,7,6,9,2,7,5,8

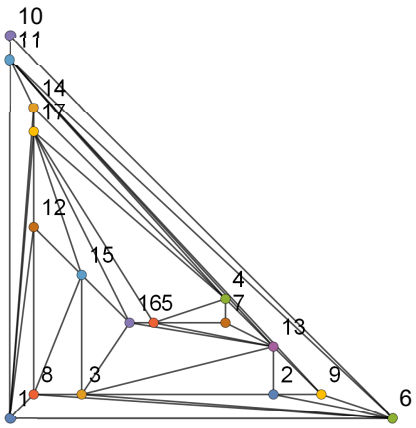
13^{th} Embedding

Triang 1	Triang 2
1: 3 10 15	1: 6 8 11 12 14 17
2: 4 5 7 15 16	2: 3 6 9 13
3: 1 5 6 7 9 10 12	3: 2 6 8 13 15 16
4: 2 5 9 16	4: 5 7 11 13 14 17
5: 2 3 4 9 15	5: 4 7 13 16 17
6: 3 7 12 13 14 15	6: 1 2 3 8 9 10 11
7: 2 3 6 9 15 16	7: 4 5 13
8: 10 11 12 14 17	8: 1 3 6 12 15
9: 3 4 5 7 16	9: 2 6 11 13
10: 1 3 8 12 14 15 17	10: 6 11
11: 8 12 17	11: 1 4 6 9 10 13 14
12: 3 6 8 10 11 14 17	12: 1 8 15 17
13: 6	13: 2 3 4 5 7 9 11 16
14: 6 8 10 12 15	14: 1 4 11 17
15: 1 2 5 6 7 10 14	15: 3 8 12 16 17
16: 2 4 7 9	16: 3 5 13 15 17
17: 8 10 11 12	17: 1 4 5 12 14 15 16



Color List: 1,1,2,3,4,3,6,4,8,5,7,6,9,2,7,5,8

Figure D.25: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



Color List: 1,1,2,3,4,3,6,4,8,5,7,6,9,2,7,5,8

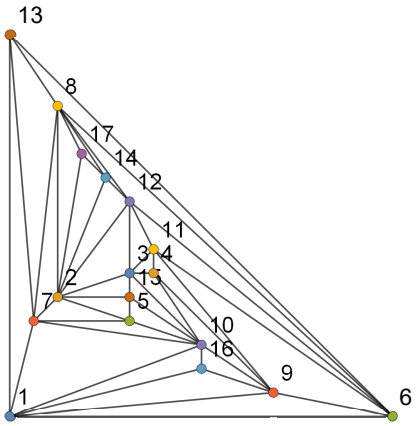
Figure D.26: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.14: nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Color List: 1,2,1,2,3,3,4,8,4,5,8,5,6,7,6,7,9

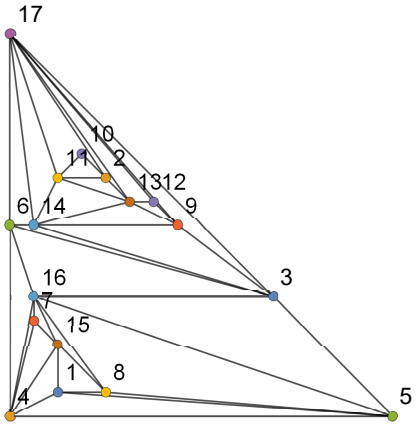
14^{th} Embedding

Triang 1	Triang 2
1: 6 7 9 10 13 16	1: 4 5 8 15
2: 3 5 7 8 12 14 15 17	2: 10 11 13 17
3: 2 4 10 11 12 15	3: 5 6 9 14 16 17
4: 3 9 10 11	4: 1 5 6 7 15 16
5: 2 7 10 15	5: 1 3 4 8 16
6: 1 8 9 11 12 13	6: 3 4 14 16 17
7: 1 2 5 8 10 13	7: 4 15 16
8: 2 6 7 12 13 14 17	8: 1 5 15 16
9: 1 4 6 10 11 16	9: 3 12 13 14 17
10: 1 3 4 5 7 9 15 16	10: 2 11
11: 3 4 6 9 12	11: 2 10 13 14 17
12: 2 3 6 8 11 14	12: 9 13 17
13: 1 6 7 8	13: 2 9 11 12 14 17
14: 2 8 12 17	14: 3 6 9 11 13 17
15: 2 3 5 10	15: 1 4 7 8 16
16: 1 9 10	16: 3 4 5 6 7 8 15
17: 2 8 14	17: 2 3 6 9 11 12 13 14



Color List: 1,2,1,2,3,3,4,8,4,5,8,5,6,7,6,7,9

Figure D.27: Triang 1 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$



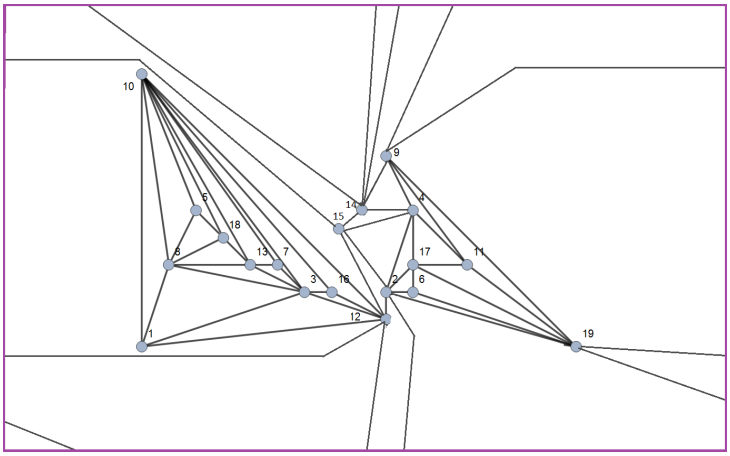
Color List: 1,2,1,2,3,3,4,8,4,5,8,5,6,7,6,7,9

Figure D.28: Triang 2 - nv=17, genus=0, orient=0, thickness=2, K_3-free, $\chi = 9$

Table D.15: nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

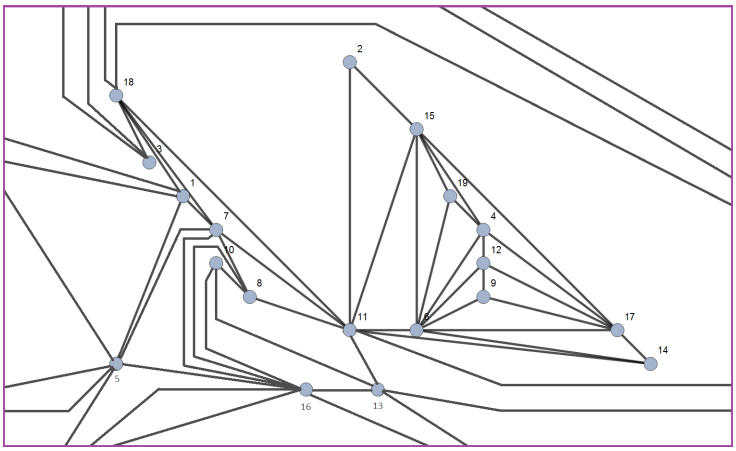
1st Embedding

Triang 1	Triang 2
1: 3 8 10 12	1: 5 7 13 16 18
2: 4 6 9 12 14 15 17 19	2: 11 15
3: 1 7 8 10 12 13 16	3: 5 16 18
4: 2 9 11 14 15 17	4: 6 12 15 17 19
5: 8 10 18	5: 1 3 7 11 13 16 18
6: 2 17 19	6: 4 9 11 12 14 15 17 19
7: 3 10 13	7: 1 5 8 11 16 18
8: 1 3 5 10 13 18	8: 7 10 11 16
9: 2 4 11 14 15 19	9: 6 12 17
10: 1 3 5 7 8 12 13 16 18	10: 8 13 16
11: 4 9 17 19	11: 2 5 6 7 8 13 14 15 18
12: 1 2 3 10 14 15 16 19	12: 4 6 9 17
13: 3 7 8 10 18	13: 1 5 10 11 16
14: 2 4 9 12 15 19	14: 6 11 17
15: 2 4 9 12 14	15: 2 4 6 11 17 19
16: 3 10 12	16: 1 3 5 7 8 10 13 18
17: 2 4 6 11 19	17: 4 6 9 12 14 15
18: 5 8 10 13	18: 1 3 5 7 11 16
19: 2 6 9 11 12 14 17	19: 4 6 15



Color list = 1,1,2,2,3,3,4,6,4,7,5,5,8,6,7,9,8,10,9

Figure D.29: Triang 1 - nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$



Color list = 1,1,2,2,3,3,4,6,4,7,5,5,8,6,7,9,8,10,9

Figure D.30: Triang 2 - nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

Table D.16: nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

Color List: 1,2,1,5,4,2,3,6,7,8,3,4,5,6,7,9,8,9,10

2^{nd} Embedding

Triang 1	Triang 2
1: 2 4 5 7 8 9 14 19	1: 2 4 10 14 16
2: 1 7	2: 1 4 5 8 9 10 14 16 19
3: 7 13 17	3: 6 11 12 14 15 17 18
4: 1 5 8 9 10 11 16 19	4: 1 2
5: 1 4 7 10 14 19	5: 2 8 9 16
6: 7 17 18	6: 3 11 12 13 14 15 17
7: 1 2 3 5 6 13 14 17	7: 12 15 18
8: 1 4 10 11 16 19	8: 2 5 9
9: 1 4 11 19	9: 2 5 8 10 16
10: 4 5 8 11	10: 1 2 9 16 19
11: 4 8 9 10 13 15 16 17 18 19	11: 3 6 12
12: 13	12: 3 6 7 11 14 15 17 18
13: 3 7 11 12 17	13: 6 14 15 18
14: 1 5 7	14: 1 2 3 6 12 13 15 17 18
15: 11	15: 3 6 7 12 13 14 17 18
16: 4 8 11 19	16: 1 2 5 9 10
17: 3 6 7 11 13 18	17: 3 6 12 14 15 18
18: 6 11 17	18: 3 7 12 13 14 15 17
19: 1 4 5 8 9 11 16	19: 2 10

Table D.17: nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

Color List: 1,1,3,2,2,3,4,4,5,5,6,7,7,8,8,9,9,10,6

3rd Embedding

Triang 1	Triang 2
1: 3 4 8 9 15 16 18	1: 4 11 12 13
2: 7 11 14 17	2: 5 6 10 12 19
3: 1 4 8 9 13 15 19	3: 16 18 19
4: 1 3 8 9 12 15 18	4: 1 11 13 16
5: 7 10 11	5: 2 6 12 14 17 19
6: 7 10 11 12 14 17	6: 2 5 19
7: 2 5 6 10 11 12 17	7: 12 14 19
8: 1 3 4 9 15 16 18	8: 9 13 15 18 19
9: 1 3 4 8 13 16 18 19	9: 8 13 15
10: 5 6 7 11	10: 2 12 14 17 19
11: 2 5 6 7 10 14	11: 1 4 12 14 17
12: 4 6 7	12: 1 2 5 7 10 11 14 17
13: 3 9 19	13: 1 4 8 9 15 16 18
14: 2 6 11 17	14: 5 7 10 11 12 17 19
15: 1 3 4 8 16 18 19	15: 8 9 13
16: 1 8 9 15 19	16: 3 4 13 18
17: 2 6 7 14	17: 5 10 11 12 14 19
18: 1 4 8 9 15	18: 3 8 13 16 19
19: 3 9 13 15 16	19: 2 3 5 6 7 8 10 14 17 18

Table D.18: nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

Color List: 1,1,2,3,4,5,6,2,7,3,4,5,6,8,7,9,9,8,10

4^{th} Embedding

Triang 1	Triang 2
1: 4 7 17	1: 3 5 6 8 9 15 18
2: 3 10 11 12 14 15 16 19	2: 13 17
3: 2 4 5 14 15	3: 1 6 7 10 11 12 16 18 19
4: 1 3 5 6 7 8 15 17 18	4: 9 17
5: 3 4 6 7 8 9 17 18	5: 1 15
6: 4 5 8 18	6: 1 3 7 9 15 17
7: 1 4 5 8 9 17 18	7: 3 6 15
8: 4 5 6 7 9 18	8: 1 9 13 15 17
9: 5 7 8 16 18	9: 1 4 6 8 11 13 14 17 19
10: 2 13 16 19	10: 3 11 12 14 15 18
11: 2 12 13 19	11: 3 9 10 14 16 18 19
12: 2 11 13 15 16 18	12: 3 10 14 19
13: 10 11 12 16 19	13: 2 8 9 14 17
14: 2 3 16	14: 9 10 11 12 13 17 19
15: 2 3 4 12 18	15: 1 5 6 7 8 10 17
16: 2 9 10 12 13 14 18	16: 3 11 19
17: 1 4 5 7	17: 2 4 6 8 9 13 14 15 19
18: 4 5 6 7 8 9 12 15 16	18: 1 3 10 11
19: 2 10 11 13	19: 3 9 11 12 14 16 17

Table D.19: nv=19, genus=1, orient=1, thickness=2, K_3-free, $\chi = 10$

Color List: 2,1,2,3,3,5,4,8,4,5,6,7,8,9,7,1,10,6,9

5^{th} Embedding

Triang 1	Triang 2
1: 8 15 17 18	1: 4 5 6 9 10 14 16 19
2: 4 6 7 12 13 15	2: 3 9 10 11 19
3: 4 5 9 10 13 16 18 19	3: 2 7 9 11 12
4: 2 3 13 15 16 19	4: 1 6 9 10 11 12 18
5: 3 9 10 16 18	5: 1 7 8 12 14 17 19
6: 2 7 8 13 15 17	6: 1 4 11 14 15
7: 2 6 13	7: 3 5 8 11 12 14 15 17
8: 1 6 14 15 16 18	8: 5 7 17
9: 3 5 18	9: 1 2 3 4 10 11 12 13 16 19
10: 3 5 11 12 13 16	10: 1 2 4 9 16 18 19
11: 10 12 13	11: 2 3 4 6 7 9 13 15 19
12: 2 10 11 13 16	12: 3 4 5 7 9 18 19
13: 2 3 4 6 7 10 11 12 15	13: 9 11 19
14: 8 16 18	14: 1 5 6 7 15 17
15: 1 2 4 6 8 13 17	15: 6 7 11 14
16: 3 4 5 8 10 12 14 17 18 19	16: 1 9 10
17: 1 6 15 16 18	17: 5 7 8 14
18: 1 3 5 8 9 14 16 17	18: 4 10 12 19
19: 3 4 16	19: 1 2 5 9 10 11 12 13 18

Table D.20: nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$

Color List: 1,2,6,4,8,10,3,3,4,5,6,2,7,7,5,8,1,9,10,11,9

1st Embedding

Triang 1	Triang 2
1: 9 10 14 20	1: 2 8 11 16 18 19
2: 9 19 20	2: 1 8 10 11 14 16 18
3: 4 5 7 15 16 17 21	3: 6 12 13
4: 3 5 6 8 15 16 17 21	4: 7 12 13 14
5: 3 4 8 12 15 17 21	5: 6 7 13
6: 4 15 17	6: 3 5 7 12 13 15 21
7: 3 16 17	7: 4 5 6 12 13 14 15 16 21
8: 4 5 17	8: 1 2 9 10 11 14 16 18 19 20
9: 1 2 10 11 15 16 19 20	9: 8 14 17 18
10: 1 9 11 14	10: 2 8 16 18 19 20
11: 9 10 14 16 18 20	11: 1 2 8 19
12: 5 15 17 18 19	12: 3 4 6 7 13 14 16 19 21
13: 15 16 17 21	13: 3 4 5 6 7 12
14: 1 10 11 20	14: 2 4 7 8 9 12 16 17 18 19
15: 3 4 5 6 9 12 13 16 17 19	15: 6 7 21
16: 3 4 7 9 11 13 15 17 18 21	16: 1 2 7 8 10 12 14 19 20
17: 3 4 5 6 7 8 12 13 15 16 18 21	17: 9 14 18
18: 11 12 16 17 19 20	18: 1 2 8 9 10 14 17 19
19: 2 9 12 15 18 20	19: 1 8 10 11 12 14 16 18
20: 1 2 9 11 14 18 19	20: 8 10 16
21: 3 4 5 13 16 17	21: 6 7 12 15

Table D.21: nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$

Color List: 1,2,3,4,4,8,5,1,6,7,8,9,10,9,5,11,10,7,2,6,3

2^{nd} Embedding

Triang 1	Triang 2
1: 2 4 7 11 12 13	1: 3 9 10 16
2: 1 7 9 10 11 12 13	2: 3 4 16
3: 7 11 12	3: 1 2 4 9 10 13 15 16
4: 1 7 9 10 12 19	4: 2 3 11 13 15 16
5: 6 14 15	5: 6 8 12 14 17 18 19 20 21
6: 5 14 15 19 21	6: 5 8 17 18 19 20
7: 1 2 3 4 10 11 12	7: 8 9 12 13 16 19 20
8: 14 15 17 20 21	8: 5 6 7 11 12 13 16 18 19
9: 2 4 11 12 15 18 19 21	9: 1 3 7 10 13 16
10: 2 4 7 11 12 13 16 19	10: 1 3 9
11: 1 2 3 7 9 10 12 19 21	11: 4 8 13 16
12: 1 2 3 4 7 9 10 11 13 16	12: 5 7 8 16 20
13: 1 2 10 12 16	13: 3 4 7 8 9 11 15
14: 5 6 8 15 17 21	14: 5 18 19 20
15: 5 6 8 9 14 17 18 19 20 21	15: 3 4 13
16: 10 12 13	16: 1 2 3 4 7 8 9 11 12
17: 8 14 15	17: 5 6 18 19 20 21
18: 9 15 21	18: 5 6 8 14 17 19 20 21
19: 4 6 9 10 11 15 21	19: 5 6 7 8 14 17 18 20
20: 8 15 21	20: 5 6 7 12 14 17 18 19
21: 6 8 9 11 14 15 18 19 20	21: 5 17 18

Table D.22: nv=21, genus=1, orient=1, thickness=2, K_3-free, $\chi = 11$

Color List: 1,2,6,2,3,7,4,5,8,9,1,6,5,7,8,9,10,10,3,11,4

3^{rd} Embedding

Triang 1	Triang 2
1: 5 7 12 15 18 20 21	1: 2 4 8 14 16 21
2: 10 11 12 14 19 20	2: 1 3 6 7 9 13 14 17 19 21
3: 10 11 13 17 19	3: 2 6 9 10 21
4: 5 7 15 16	4: 1 8 12 14 18 20
5: 1 4 7 8 12 16 20 21	5: 14 15 18
6: 9 10 11	6: 2 3 10 13 17 19 21
7: 1 4 5 8 12 15 18 20	7: 2 13 14 16
8: 5 7 12 14 15 20 21	8: 1 4 16 18
9: 6 10 11 17	9: 2 3 13 19 21
10: 2 3 6 9 11 13 17 19	10: 3 6 17 21
11: 2 3 6 9 10 12 17 19	11: 13 16 21
12: 1 2 5 7 8 11 14 15 16 19 20	12: 4 15 18 20
13: 3 10 19	13: 2 6 7 9 11 16 17 19 21
14: 2 8 12 20	14: 1 2 4 5 7 15 16 18 20
15: 1 4 7 8 12 16 21	15: 5 12 14 18 20
16: 4 5 12 15 20	16: 1 7 8 11 13 14 18 21
17: 3 9 10 11	17: 2 6 10 13 19 21
18: 1 7 20	18: 4 5 8 12 14 15 16
19: 2 3 10 11 12 13 20	19: 2 6 9 13 17 21
20: 1 2 5 7 8 12 14 16 18 19	20: 4 12 14 15
21: 1 5 8 15	21: 1 2 3 6 9 10 11 13 16 17 19

Table D.23: nv=19, genus=2, orient=0, thickness=2, K_3-free, $\chi = 9$

Triang 1	Triang 2
1: 3 4 13 15 16 18 19	1: 2 7 11 14
2: 11 14	2: 1 3 4 7 11 13 14 15 16 18 19
3: 1 12 13 18 19	3: 2 4 13 15 16 19
4: 1 12 13 19	4: 2 3 15 16 18
5: 6 8 9 10 11 12 14 17	5: 7
6: 5 8 11	6: 7 9 10 12 14 17
7: 9 11	7: 1 2 5 6 8 10 14 17
8: 5 6 11 12	8: 7 9 10 14 17
9: 5 7 11 17	9: 6 8 10 12 14 17
10: 5 11 14 17	10: 6 7 8 9 12
11: 2 5 6 7 8 9 10 17	11: 1 2 14
12: 3 4 5 8 13 15 16 18 19	12: 6 9 10 17 18
13: 1 3 4 12 16 18 19	13: 2 3 15
14: 2 5 10	14: 1 2 6 7 8 9 11 17
15: 1 12 16 18 19	15: 2 3 4 13
16: 1 12 13 15 18	16: 2 3 4 19
17: 5 9 10 11	17: 6 7 8 9 12 14
18: 1 3 12 13 15 16 19	18: 2 4 12
19: 1 3 4 12 13 15 18	19: 2 3 16

Table D.24: nv=19, genus=2, orient=1, thickness=2, K_3-free, $\chi = 9$

Triang 1	Triang 2
1: 2 3 18	1: 4 5 6 7 8 10 11 13 15 19
2: 1 3 6 9 11 18	2: 5 7 14 19
3: 1 2 5 6 8 11 18	3: 9 14 16 19
4: 7 8 9 10 11 12 15 16 17	4: 1 6 10 13
5: 3 8 12 13 14 18	5: 1 2 10 16 19
6: 2 3 7 8 9 13	6: 1 4 10 11 15 17
7: 4 6 10 12 15 18	7: 1 2 9 11 14 17
8: 3 4 5 6 11 12 13 15 16	8: 1 9 10 13 17
9: 2 4 6 11 12 15 16	9: 3 7 8 14 17 19
10: 4 7 11 15 17	10: 1 4 5 6 8 12 13 16 18
11: 2 3 4 8 9 10 15 17	11: 1 6 7 13 17 18
12: 4 5 7 8 9 14 15 19	12: 10 13 16 17
13: 5 6 8 15 16	13: 1 4 8 10 11 12 17 18
14: 5 12 18 19	14: 2 3 7 9 16 17
15: 4 7 8 9 10 11 12 13 16	15: 1 6 17
16: 4 8 9 13 15	16: 3 5 10 12 14 17 19
17: 4 10 11	17: 6 7 8 9 11 12 13 14 15 16
18: 1 2 3 5 7 14 19	18: 10 11 13
19: 12 14 18	19: 1 2 3 5 9 16

Table D.25: nv=21, genus=2, orient=0, thickness=2, K_3-free, $\chi = 11$

Triang 1	Triang 2
1: 3 5 6 10 13 17	1: 3 7 8 18 21
2: 8 12 16 19 20	2: 4 9 11 14 15
3: 1 5 6 13 17 20 21	3: 1 7 10 18
4: 14 16 20	4: 2 8 9 11 12 15 16 19
5: 1 3 6 10 13 18 20 21	5: 7 17 18
6: 1 3 5 7 13 21	6: 10 17 18 20
7: 6 13 21	7: 1 3 5 8 10 17 18
8: 2 16 19	8: 1 4 7 9 11 12 14 15 16 21
9: 12 15 19 20	9: 2 4 8 11 14 16
10: 1 5 17 21	10: 3 6 7 13 17 18 20
11: 12 14 15 20	11: 2 4 8 9 14 16 19
12: 2 9 11 14 15 19 20	12: 4 8 16
13: 1 3 5 6 7 18 21	13: 10 17 20
14: 4 11 12 16 19 20	14: 2 8 9 11 15 16
15: 9 11 12 20	15: 2 4 8 14 16 19
16: 2 4 8 14 19 20	16: 4 8 9 11 12 14 15 19
17: 1 3 10 21	17: 5 6 7 10 13 18 20
18: 5 13 20	18: 1 3 5 6 7 10 17 21
19: 2 8 9 12 14 16 20	19: 4 11 15 16
20: 2 3 4 5 9 11 12 14 15 16 18 19	20: 6 10 13 17
21: 3 5 6 7 10 13 17	21: 1 8 18

Table D.26: nv=21, genus=2, orient=1, thickness=2, K_3-free, $\chi = 10$

Triang 1	Triang 2
1: 6 11 14 15 16 20	1: 5 10 13 17 19 20
2: 3 4 7 8 9 11 18 21	2: 12 17
3: 2 8 11 21	3: 4 7 9 11 12 17 18 21
4: 2 7 18 21	4: 3 7 8 9 11 12 17
5: 6 8	5: 1 6 10 13 14 15 16 19 20
6: 1 5 7 8 10 13 14 16 19	6: 5 15 20
7: 2 4 6 8 12 14 18 21	7: 3 4 9 11
8: 2 3 5 6 7 12 14 19 21	8: 4 9 12 18
9: 2 11 21	9: 3 4 7 8 12 17 18
10: 6 11 13 15 16 17 19	10: 1 5 14 20
11: 1 2 3 9 10 15 16 21	11: 3 4 7 12 13 18 20
12: 7 8 21	12: 2 3 4 8 9 11 17 18
13: 6 10 16	13: 1 5 11 14 15 16 17 19 20
14: 1 6 7 8 15 19	14: 5 10 13 16 20
15: 1 10 11 14 17 19 20	15: 5 6 13 16 20
16: 1 6 10 11 13 17 19 20	16: 5 13 14 15
17: 10 15 16 19 20	17: 1 2 3 4 9 12 13 18 21
18: 2 4 7	18: 3 8 9 11 12 17 21
19: 6 8 10 14 15 16 17 20	19: 1 5 13 20
20: 1 15 16 17 19	20: 1 5 6 10 11 13 14 15 19
21: 2 3 4 7 8 9 11 12	21: 3 17 18

REFERENCES

[ABCC07] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[Ala07] Roqyah R. Alalqam. Heuristic methods applied to difficult graph theory problems. Master's thesis, University of Colorado Denver, 2007.

[BGS07] Debra L. Boutin, Ellen Gethner, and Thom Sulanke. Thickness-two graphs part one: New nine-critical graphs, permuted layer graphs, and catlins graphs, 2007.

[Bre] "Brelaz's Heuristic Algorithm." From MathWorld–A Wolfram Web Resource. `http://mathworld.wolfram.com/BrelazsHeuristicAlgorithm.html`. Accessed: 2010-09-30.

[Bré79] Daniel Brélaz. New methods to color the vertices of a graph. *Commun. ACM*, 22(4):251–256, April 1979.

[DS04] Marco Dorigo and Thomas Stutzle. *Ant Colony Optimization*. A Bradford Book; First Edition, 2004.

[DSA] Michael Trick's DSATUR Algorithm. `http://mat.gsia.cmu.edu/COLOR/solvers/trick.c`. Accessed: 2010-09-30.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[GS09] Ellen Gethner and Thom Sulanke. Thickness-two graphs part two: More new nine-critical graphs, independence ratio, cloned planar graphs, and singly and doubly outerplanar graphs. *Graph. Comb.*, 25(2):197–217, May 2009.

[HdW87] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[Hea90] P. J. Heawood. Map colour theorem. pages 332–338, 1890.

[JR83] Brad Jackson and Gerhard Ringel. Maps of m-pires on the projective plane. *Discrete Math.*, 46(1):15–20, January 1983.

[Kra42] M. Kraitchik. *Mathematical Recreations*. W.W. Norton, New York, 1942.

[Man93] A. Mansfield. Determining the thickness of graphs is np-hard. In *Mathematical Proceedings of the Cambridge Philosophical Society*, pages 9–23, 1993.

[PP05] Fernando Magno Quintão Pereira and Jens Palsberg. *Register Allocation Via Coloring of Chordal Graphs*, pages 315–329. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[PS03] Sriram Pemmaraju and Steven Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica ®*. Cambridge University Press, New York, NY, USA, 2003.

[Rin59] Gerhard Ringel. *Färbungsprobleme auf Flächen und Graphen*, volume 2 of *Mathematische Monographien*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1959.

[Wes01] D. B. West. *Introduction to Graph Theory 2nd Edition*. Prentice Hall, 2001.