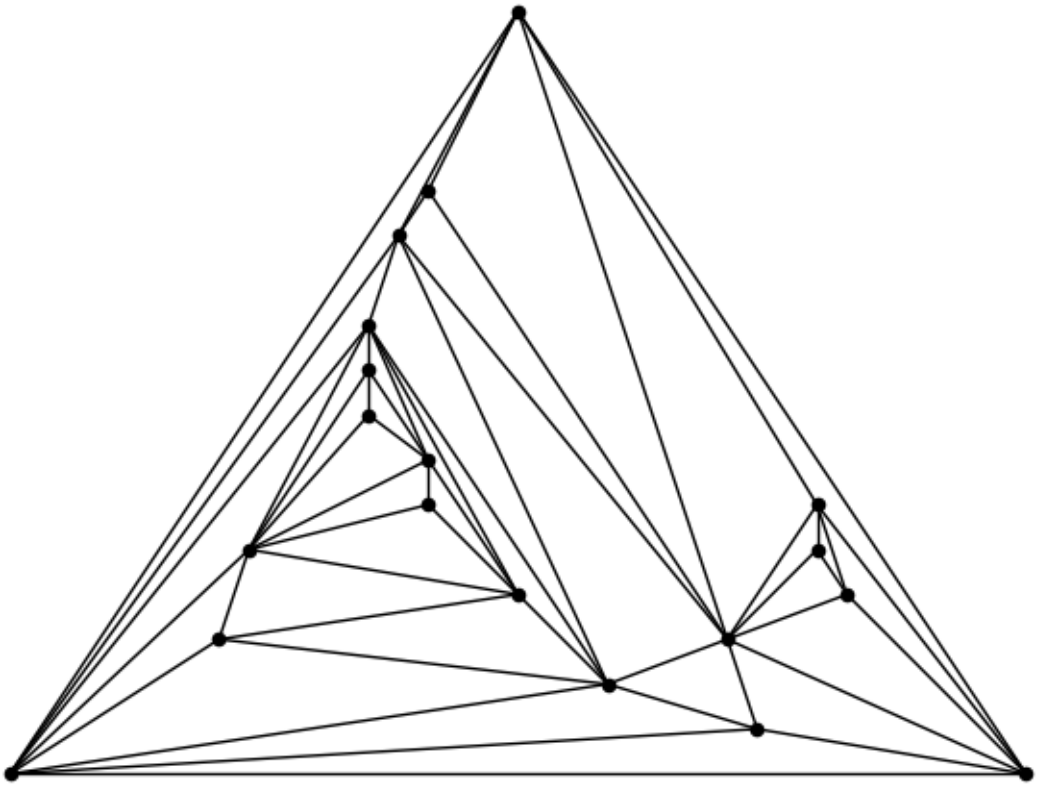
Generating Graphs for the Earth-Moon Problem with Edge Flips

Revisiting Sulanke's Algorithm

Master Thesis Applied Mathematics

Bob Vermeulen



Supervisor: Prof.dr. F.M. Spieksma

September 21, 2025

Universiteit Leiden

Contents

1 Introduction

We start out with some basic definitions and notations of graph theory, such that we can properly define the Earth-Moon Problem: What is the minimum number of colours required to colour any map that can be split into two planar graphs? Currently the Earth-Moon problem is unsolved, but we do have a lower and upper bound on the minimum number of required colours. We show both of these bounds and provide proofs for them later in this section.

Section 2 is all about Sulanke's Algorithm. This was the best known algorithm for generating Earth-Moon graphs, until it was lost for over a decade. Luckily, the algorithm was recovered in 2022 [1], but the information about edge cases and other details about about implementation are still lost. We fill in the missing gaps ourself to create an altered version of Sulanke's Algorithm and provide theorems to support the changes we made. In particular, we made a constructive proof that every edge-maximal planar graph can be obtained from any other edge-maximal planar graph with the same number of vertices, by applying so called *edge flips*. This proof is a new result, which shows that Sulanke's Algorithm can generate all graphs we are interested in. We also put our adapted algorithm to the test, and the most interesting graphs we found are shown in Appendix B.

Two other leads for the Earth-Moon problem are discussed in Section 3. First, we give a brief summary of Tilley's new view on the 4-colour theorem and theorise whether that information is also applicable to the Earth-Moon Problem. Second, we delve deeper into 2-pire graphs. These graphs are currently the key to the best upper bound that we have of the Earth-Moon Problem and we explore an idea that might lead to an even tighter upper bound with the help of 2-pire graphs.

Our research did not lead to a new bound for the Earth-Moon problem, but that does not stop further research on the topic. We discuss possible continuations of our research and mention some other ideas on the topic in Section 4. From our research it seems unlikely that Sulanke's Algorithm can improve the current lower bound on the Earth-Moon problem, but we do think that the bound can be improved by approaching the problem with different algorithms.

1.1 Defining the Earth-Moon problem

This subsection contains some graph theory basics and other commonly used definitions in this thesis, starting with simple graphs.

Definition 1.1 (Simple graph) A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, where each edge is an unordered pair of two vertices. G is simple, if all edges are unique and each edge is associated with two different vertices.

Given a graph $G = (V, E)$, we will call the number of vertices $|V| = n$ and the number edges $|E| = m$. Furthermore, we call duplicate edges parallel edges, and we call edges that connect a vertex to itself self-loops. We are mostly interested in simple graphs for the Earth-Moon problem, as the existence of non-simple graphs will complicate things, as seen later on. Another relevant property for graphs is planarity. A graph is called planar, if the graph can be drawn on a plane, such that edges are connected to their corresponding vertices and the edges do not intersect. We call such a representation a planar embedding of the graph.

The Earth-Moon problem is all about colouring graphs, which leads us to the next definition.

Definition 1.2 (Graph colouring) A (proper) colouring of a graph G is a labelling of the vertices where we assign each vertex a colour such that adjacent vertices have different colours. The chromatic number $\chi(G)$ of a graph is the minimal number of colours required to colour the vertices of graph G.

Graphs with self-loops cannot be properly coloured and parallel edges are redundant when colouring a graph. Therefore restricting ourselves to simple graphs when colouring graphs is preferable.

Arguably, the most famous theorem about chromatic numbers and planar graphs is the 4-colour theorem, proven by Appel and Haken in 1976 [2]. This theorem can be seen as the predecessor of the Earth-Moon problem.

Theorem 1.3 (4-colour theorem) *Let G be a simple planar graph. Then $\chi(G) \leqslant 4$ holds.*

There exist simple planar graphs with chromatic number 4. For example, $\chi(K_4) = 4$. It follows, that the bound of the 4-colour theorem is tight.

This theorem is famous for being one of the first theorems that heavily relies on computer technology for its proof, therefore it is not realistic to show a proof here. We will however cover a conjecture made by Tilley in 2018 [3] about the 4-colour theorem. If his conjecture is true, it would provide a more intuitive proof for the 4-colour theorem than the current computer assisted proofs. Tilley's conjecture is covered in Subsection 3.1. There, we examine whether concepts used in the conjecture could help with the Earth-moon problem as well. Colouring planar graphs with as few colours as possible can also be seen as a map colouring problem, in which we want to colour all countries on a map with the fewest number of colours, such that countries with a shared border have different colours. We will mostly look at the graph theoretic approach of colouring problems though.

We are especially interested in graphs with a high chromatic number. Adding an edge to a graph never decreases its chromatic number. Therefore, adding as many edges as possible to a graph is desired if we want to maximize its chromatic number. An important subset of simple planar graphs for colouring problems is the set of all edge-maximal planar graphs. This set consists of all simple planar graphs, that are no longer simple and planar if any edge is added to them. The next simple yet convenient theorem shows two additional identifying properties for edge-maximal planar graphs.

Theorem 1.4 (Triangulated graph) *Let G be a simple planar graph with $n \geqslant 3$ vertices, m edges and a planar embedding. Then the following is equivalent:*

(1) If any edge is added to G, it is no longer a simple planar graph.

(2) All faces of the planar embedding of G are triangles.

(3) $m = 3n - 6$ holds.

A simple planar graph that satisfies any of these properties is called an edge-maximal graph or triangulated graph. These two names can be used interchangeably.

Proof. (1) $\implies$ (2) Since G is simple, each face consists of at least 3 vertices. Assume there is a face with 4 or more vertices. We can add an edge between two non-adjacent vertices of that face and G will still be simple and planar. This is in contradiction with the edge-maximal property, thus there are no faces with 4 or more vertices. We conclude that all faces have exactly 3 vertices. Faces with three vertices are known as triangles.

(2) $\implies$ (3) Denote f as the number of faces of G. Euler's Polyhedron Formula gives us a relation between the number of vertices, edges and faces of a graph with a planar embedding. It tells us that $n - m + f = 2$ must hold. Since all faces are triangles, each face is adjacent to exactly 3 edges and each edge is adjacent to exactly 2 faces. Thus the equality $2m = 3f$ must also hold. We use this equality to eliminate f from Euler's Formula and get that $m = 3n - 6$ must be satisfied by G.

(3) $\implies$ (1) Since G is simple, all faces have at least 3 edges. Therefore, each face is adjacent to at least 3 edges and each edge is adjacent to exactly 2 faces. This gives us the inequality $2m \geqslant 3f$. By substituting this inequality into Euler's Formula we get $m \leqslant 3n - 6$, which holds for all simple planar graphs. If we were to add an edge to G we have $3n - 5$ edges in total, which means the new graph can not be a simple planar graph. □

It is important to note, that in context of this thesis, triangulated graphs are simple and planar by definition.

Figure 1 shows two graphs where the vertices v_1 and v_2 are swapped. These two graphs are similar, but not equal, because vertex v_1 has a different degree in each graph. We call graphs that are the same, up to the order of their vertices, isomorphic graphs. Two isomorphic graphs always have the

Figure 1: Two different graphs that are isomorphic.

same chromatic number. Graph isomorphism will play an important role in some of the upcoming sections.

Definition 1.5 (Graph Isomorphism) Let $G = (V_1, E_1)$ and $H = (V_2, E_2)$ be two graphs. G and H are isomorphic if there exists a bijection $f : V_1 \to V_2$, such that any two vertices $v, w \in V_1$ are adjacent in G if and only if $f(v)$ and $f(w)$ are adjacent in H. We denote this property as $G \simeq H$.

We require one more definition before we can state the Earth-Moon Problem, namely the so-called thickness of a graph. The thickness can be seen as a way of measuring how "planar" a graph is.

Definition 1.6 (Thickness) For a graph $G = (V, E)$ we define the thickness $\theta(G)$ as the minimal number of partitions of E, such that each partition with vertex set V is a planar graph.

Graphs on subsets of the edge set are called subgraphs, which are also used more generally. Graphs with a thickness of 1 are planar graphs and graphs with a higher thickness are not. We define the Earth-Moon problem by extending the 4-colour problem to graphs with a thickness of 2.

Problem 1.7 (Earth-Moon problem) What is the maximum chromatic number a thickness 2 graph can have?

This problem was first introduced by Ringel [4] in 1959, when the 4-colour theorem was still a conjecture. He predicted that 8 colours are sufficient to colour any graph with thickness 2. The problem can also be formulated as follows: Let be given a map of countries, but each country has one enclave on the moon. We want to assign a single colour to each country and enclave pair, such that countries and enclaves that share a border have different colours. This formulation is the origin of the Earth-Moon Problem's name.

1.2 Sulanke's Earth-Moon graph

In 1974, 15 years after Ringel stated the Earth-Moon problem, it turned out that Ringel's prediction was false and that there do exist Earth-Moon graphs with a chromatic number of 9. Inspired by a problem proposed by the popular mathematician Gardner, Sulanke discovered the assumedly first known Earth-Moon graph with a chromatic number of 9, which is shown in Figure 2 [5]. That makes this graph a counterexample to Ringel's prediction.

Because Sulanke's graph can be split into two planar graphs, we know it has thickness 2. Furthermore, we see that the vertices 6 through 11 form the complete graph K_6, which requires 6 different colours. Vertices 1 through 5 are all directly connected to each vertex of the K_6, so that vertices 1 through 5 must use different colours than the 6 already used. These 5 vertices form a cycle of length 5, which requires 3 different colours. In total we need at least 6+3=9 colours to properly colour Sulanke's graph, giving it chromatic number 9.

As of today, this is the best known lower bound of the Earth-Moon problem. Many other examples of thickness 2 graphs with chromatic number 9 have been found by Sulanke and Gethner. They created an algorithm to find these graphs, which is known as Sulanke's Algorithm nowadays [6]. So far, this is the best known algorithm for generating thickness 2 graphs with chromatic number 9. In Section 2 we discuss this algorithm.

1.3 Empire graphs

Apart from a lower bound for the Earth-Moon Problem, an upper bound is known. To prove this upper bound we use a set of graphs, where vertices are grouped in a way that is reminiscent of graph thickness.

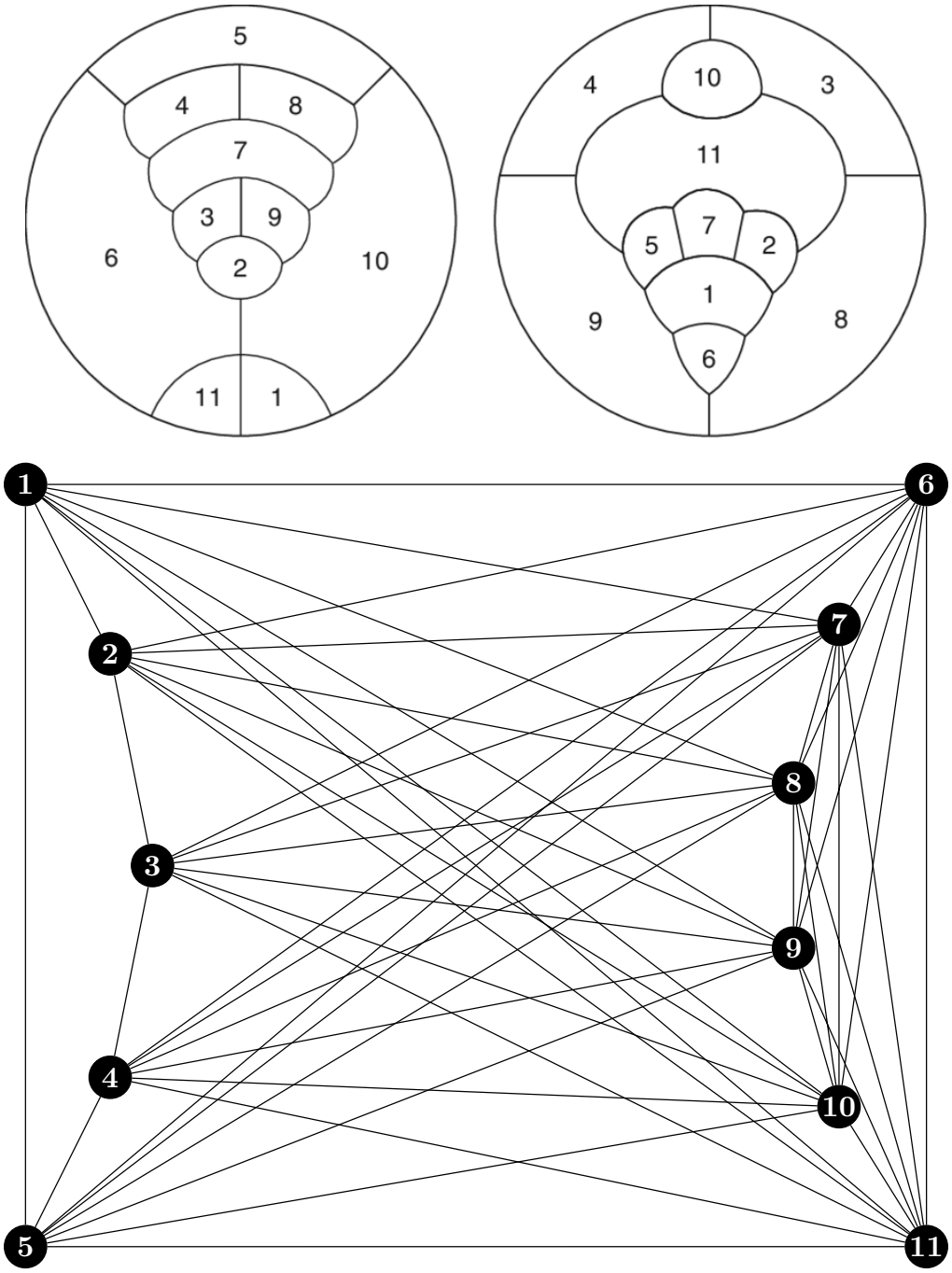


Figure 2: Earth-Moon map and its corresponding graph with a chromatic number of 9 by Sulanke [5].

Definition 1.8 (M-pire graph) For $M \geqslant 1$, an M-pire graph is a planar graph, where the set of vertices is partitioned into sets we call empires. Empires contain at most M vertices and no pair of vertices in an empire share an edge. We call such a set of vertices that do not share edges an independent set. A proper colouring of an M-pire graph follows the same rules as stated in Definition 1.2, but has the extra restriction that all vertices in an empire must be coloured with the same colour.

We are interested in the highest chromatic number an M-pire graph can have. Heawood found an upper bound for this problem in 1890 [7].

Theorem 1.9 *Let G be an M-pire graph for $M \geqslant 1$. Then $\chi(G) \leqslant 6M$ [7].*

Proof. We prove this theorem by contradiction. Assume that there exists an M-pire graph with a chromatic number of at least $6M+1$. Let $G = (V, E)$ be an M-pire graph with the smallest number of vertices, such that $\chi(G) \geqslant 6M + 1$ holds. With this graph, we construct a new graph $G^* = (V^*, E^*)$

by merging the vertices of each empire into one vertex. Each empire in G is a vertex in V^* and two empires are connected in G^* if there is a pair of connected vertices in the two associated empires. Since the structure of the graph is preserved, G and G^* have the same chromatic number.

For these two graphs $|V| \leqslant M|V^*|$ holds, because empires in G contain at most M vertices. Furthermore $|E^*| \leqslant |E|$ is also true, because there could be multiple edges between two empires in G which we reduce to a single edge in E^*.

To get to a contradiction, we look at the average vertex degree $D = \frac{2|E^*|}{|V^*|}$ of graph G^*. Note, $2|E^*|$ is the sum of all the vertex degrees. Euler's polyhedron formula gives us the upper bound $|E| \leqslant 3|V| - 6$, which we use to give an upper bound on D. We also use the two aforementioned upper bounds.

$$D = \frac{2|E^*|}{|V^*|} \leqslant \frac{2|E|}{|V^*|} \leqslant \frac{2\big(3|V| - 6\big)}{|V^*|} = \frac{6|V| - 12}{|V^*|} \leqslant \frac{6M|V^*| - 12}{|V^*|} = 6M - \frac{12}{|V^*|}$$

As $\chi(G^*) \geqslant 6M + 1 \geqslant 2$ holds, G^* must contain at least one vertex. It follows that $\frac{12}{|V^*|} > 0$ holds, so that $D < 6M$ is true. For the average degree to be this small, there must exist a vertex in V^* with a degree of at most $6M - 1$. This corresponds to an empire in G that is adjacent to at most $6M - 1$ other empires. We remove this empire and its corresponding edges from G to construct a new graph $\tilde{G}$. Since G is the smallest M-pire graph with $\chi(G) \geqslant 6M + 1$, we have $\chi(\tilde{G}) \leqslant 6M$.

Let $\tilde{G}$ be coloured with $6M$ colours and apply the same colouring to G. One empire is still uncoloured in G. By construction, this empire is adjacent to at most $6M - 1$ empires. At least one of the $6M$ colours is still available to colour this empire, which results in a proper colouring of G with only $6M$ colours. This is a contradiction with $\chi(G) \geqslant 6M + 1$ as there should not exist a proper colouring with $6M$ colours of G. Therefore, there does not exist a smallest M-pire graph G with a chromatic number of at least $6M + 1$ and we conclude that there cannot exist an M-pire graph with chromatic number $6M + 1$ or greater. □

Heawood also showed that the bound of Theorem 1.9 is tight for $M = 2$ [7]. About a century later, Ringel and Young proved that the bound is tight for all $M > 1$ [8]. We only provide an example that shows the bound is tight for $M = 2$, because the $M = 2$ case is relevant for the Earth-Moon problem.

Figure 3 depicts a 2-pire map with a chromatic number of 12. In this example all 12 empires are directly connected to all other empires, which makes it a 2-pire version of the K_{12}. Therefore, the chromatic number of this example is equal to $12 = 6M$.

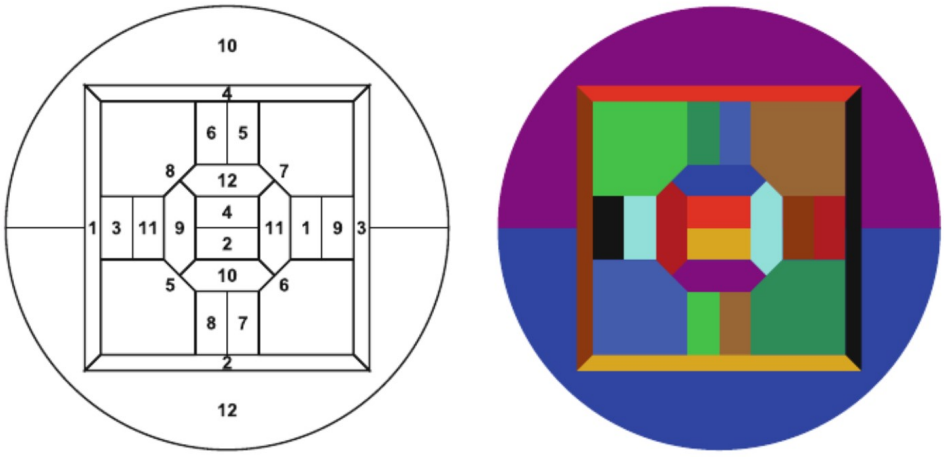


Figure 3: 2-pire graph with chromatic number 12 found by Kim, image made by Gethner[5].

We are especially interested in 2-pire graphs, because they are similar to thickness 2 graphs. If we split a thickness 2 graph into two planar graphs and make the split vertices 2-pires, we have created a

2-pire graph. A colouring of such a 2-pire graph is proper if and only if that same colouring is a proper colouring of the thickness 2 graph. When viewing these families of graphs this way, thickness 2 graphs are a subset of 2-pire graphs. Therefore we can apply Theorem 1.9 to the Earth-Moon problem.

Corollary 1.10 *Let G be a graph with $\theta(G) = 2$. Then $\chi(G) \leqslant 12$.*

Despite the fact that 2-pire and thickness 2 graphs are very similar, 2-pire graphs often have a thickness greater than 2. For example, the 2-pire version of the K_{12} from Figure 3 has a thickness of 3, because it has too many edges to be a thickness 2 graphs. Theorem 1.4 tells us that planar graphs on 12 vertices have at most $3 \cdot 12 - 6 = 30$ edges and thickness 2 graphs have at most double that amount with 60 edges. The K_{12} has 66 edges and thus $\chi(K_{12}) > 2$. All 2-pire graphs with a thickness greater than 2 are irrelevant for the Earth-Moon problem.

Currently, this is the best known upper bound on the Earth-Moon problem. It raises the question how tight this bound is for thickness 2 graphs. A thickness 2 graph is more restricted than a 2-pire graph, so it seems plausible that the bound could be lowered by using properties unique to Earth-Moon graphs. Ideas for improving this upper bound are covered in Chapter 3. Thanks to Sulanke's graph, we know that there exist thickness 2 graphs that require 9 colours to colour and the previous corollary shows that 12 colours are always enough to colour a thickness 2 graph. With these two facts combined, we know that the solution of the Earth-Moon problem is 9, 10, 11 or 12.

2 Sulanke's Algorithm

Sulanke is the creator behind the allegedly fastest known algorithm for generating thickness 2 graphs with chromatic number 9 [6]. In this chapter, we cover Sulanke's Algorithm and investigate if it is realistic to find a thickness 2 graph with chromatic number 10 with the algorithm. The discovery of such a graph would increase the lower bound of the Earth-Moon problem to 10.

2.1 Origin and definition

The most difficult part of constructing an algorithm that finds thickness 2 graphs with a high chromatic number is the fact, that computing the chromatic number of a graph is an NP-hard problem [9]. There are faster algorithms to compute chromatic numbers for specific families of graphs, but to our current knowledge these families of graphs are not helpful for the Earth-Moon Problem. In practice, it does not seem realistic to compute the exact chromatic number of the graphs we consider.

For our research, we bravely try to approximate the chromatic number of promising thickness 2 graphs in an alternative way. The Lovász number gives a lower bound on chromatic number of a graph, which can be computed in polynomial time [10]. To get a lower bound, we apply 10 different greedy colouring algorithms, which takes linear time to compute. We apply these bounds to promising thickness 2 graphs for the Earth-Moon problem. For graphs with 17 vertices, the upper and lower bounds are equal most of the time, but our method falls short for graphs with more vertices. Interestingly, this is a very fast algorithm to compute the exact chromatic number of smaller graphs. Unfortunately, our algorithm performs worse on graphs with high chromatic numbers, which are exactly the graphs we are interested in. This ultimately led us to drop this idea.

Sulanke's Algorithm is an algorithm made to find promising graphs for the Earth-Moon problem. It does so by generating pairs of planar graphs that (hopefully) have a high chromatic number when combined into a single thickness 2 graph. The algorithm also has to deal with the problem that computing chromatic numbers of graphs is NP-hard. It uses the second best option to overcome this problem: A lower bound on chromatic numbers obtained from of the Complement Theorem.

Theorem 2.1 (Complement Theorem) *Let $G = (V, E)$ be a graph with $|V| = n$ vertices, such that the complement G^c does not have the K_a as subgraph. Then,*

$$\chi(G) \geqslant \left\lceil \frac{n}{a-1} \right\rceil.$$

Proof. The colouring of a graph with x colours can be seen as partitioning the vertices of G into x sets of vertices with the same colour. Each subgraph induced by one of these partitions is a complete graph in G^c, because between vertices of the same colour, there cannot be any edge. By assumption, these subgraphs contain at most $a-1$ vertices, implying that G^c does not contain K_a as a subgraph. This yields the bound $\chi(G) \geqslant \left\lceil \frac{n}{a-1} \right\rceil$ [6]. □

Naturally we are interested in the tightest cases for which graphs satisfy the given requirements. The six most useful cases for the Earth-Moon problem are given below.

Corollary 2.2 *Let $G = (V, E)$ be a graph with n vertices. Then the following is true:*

If $n = 17$ and $K_3 \nsubseteq G^c$, then $\chi(G) \geqslant 9$. *If $n = 19$ and $K_3 \nsubseteq G^c$, then $\chi(G) \geqslant 10$.*
If $n = 25$ and $K_4 \nsubseteq G^c$, then $\chi(G) \geqslant 9$. *If $n = 28$ and $K_4 \nsubseteq G^c$, then $\chi(G) \geqslant 10$.*
If $n = 33$ and $K_5 \nsubseteq G^c$, then $\chi(G) \geqslant 9$. *If $n = 37$ and $K_5 \nsubseteq G^c$, then $\chi(G) \geqslant 10$.*

Corollary 2.2 implies that graphs with few complete subgraphs in their complement are good candidates for having a high chromatic number. Sulanke's Algorithm makes use of this fact. First pick a number of vertices n and complete graph K_a from one of the statements in Corollary 2.2. Then, generate a graph on n vertices for which the algorithm minimizes the number of complete graphs K_a in the complement. Sulanke's Algorithm is very good at generating graphs with chromatic number 9 and it seems plausible that it could also generate a graph with a chromatic number of 10. This information about Sulanke's Algorithm is from Sulanke and Gethner's paper about the Earth-Moon problem, which was published in 2009 [6].

Unfortunately, Sulanke and Gethner's paper that shows the results made with Sulanke's Algorithm does not include the algorithm itself. They planned to publish the algorithm in a paper that has never been released. Van der Beek worked on the Earth-Moon Problem in 2022, found Sulanke and Gethner's work and hit a dead end due to the missing algorithm. By contacting Sulanke, van der Beek was able to recover and publish Sulanke's Algorithm with the help of Sulanke himself [1]. In 2024, van Valkengoed has successfully implemented the algorithm again as a proof of concept [11]. This thesis can be seen as a continuation of their work on the Earth-Moon problem.

With the algorithm recovered, we can explain in detail how the algorithm works. We consider the basic version of the algorithm that minimizes the number of K_3s (or triangles) in the complement of a thickness 2 graph. To initiate the algorithm, we need two triangulated graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ on the same n vertices. These two planar graphs together form the thickness 2 graph $G = (V, E_1 \cup E_2)$. It is possible that G_1 and G_2 both contain the same edge e. When this is the case, we only add edge e to G once. This prevents parallel edges from occurring in G. There are no self loops in G_1, G_2 and G, because G_1 and G_2 are simple by definition. It follows that G is also a simple graph. We only consider triangulated planar graphs for G_1 and G_2, because they are edge-maximal, see Theorem 1.4. The more edges in the graph, the less edges in the complement G^c.

Now that we have a starting point with two triangulated graphs, we want to turn G into a graph with few triangles in its complement. This is done by moving edges around in one of these planar graphs with so called flips. Figure 4 shows the configuration we require to do a flip: two faces that share an edge in a planar representation of the graph. The figure also shows, which edge e gets replaced by its complementary edge $\tilde{e}$. There are three reasons why flips are helpful. Firstly, flips always preserve planarity, because they only alter the graph locally. Secondly, the configuration required to do a flip is very common in triangulated graphs. Thirdly, flips mostly preserves the triangulated property, since the number of edges stays the same. We have to be careful here, because flipping an edge in a

triangulated (and thus simple) graph can create a non-simple graph. How Sulanke's Algorithm exactly deals with this problem is unknown, but the most practical solution is to keep the graph triangulated at all times. If a graph before and after a flip is triangulated, we call that flip a simple flip. We restrict ourselves to simple flips for Sulanke's Algorithm.

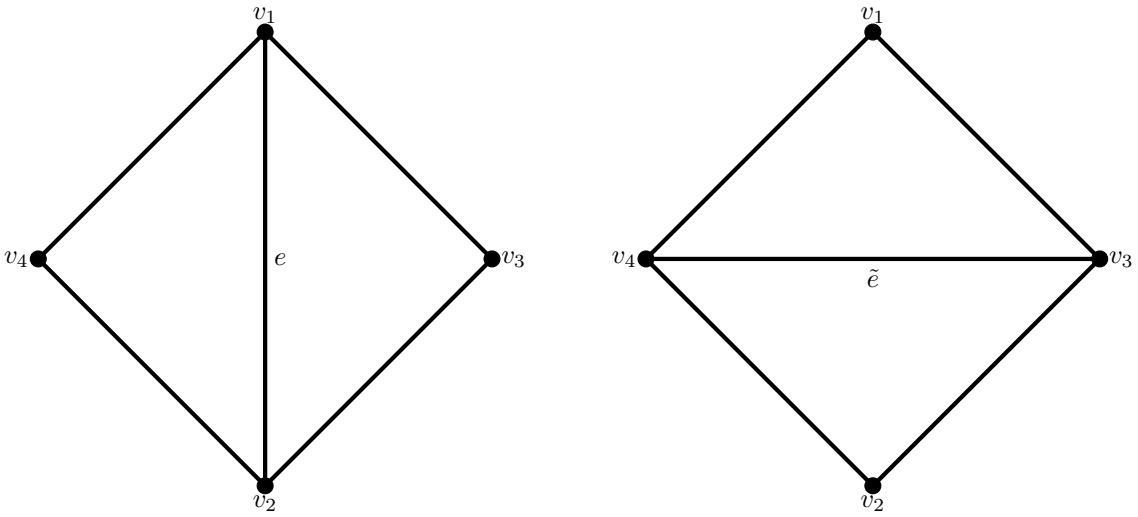


Figure 4: Example of an edge e flipped into its complementary edge $\tilde{e}$, given a planar representation.

By repeatedly doing flips that reduce the number of triangles in the complement of G, or at least keep this amount the same, we already have a working algorithm. Unfortunately this strategy tends to get the algorithm stuck in a local optimum, which gives poor results. To solve this, Sulanke's Algorithm uses simulated annealing. This is a form of probabilistic optimization, that repeatedly applies small changes to an instance of a given problem. A change is accepted, if it improves the instance and is sometimes rejected if it makes the instance worse. The probability of rejecting a bad change increases over time. The idea behind simulated annealing is to accept bad changes at first to prevent the solution getting stuck in a local optimum. When the algorithm is close to the end, we want to find the best solution that is close to the current state. Rejecting more bad changes near the end helps with finding the best nearby solution.

In the case of Sulanke's Algorithm, the small changes we do are the simple flips. The algorithm repeatedly does flips and checks whether that reduces the number of triangles in the complement. If the number of triangles has increased, we undo the flip with a probability dependent on the number of added triangles and the number of iterations already done.

After a fixed number of iterations or once the complement of G is triangle free, we end the algorithm. If we do all the mentioned steps consecutively, we get Sulanke's Algorithm as presented in Algorithm 1.

Note, that this is just the basic version of the algorithm. We can also expand it by minimizing the number of K_a for arbitrary $a \geqslant 3$. According to the Complement Theorem 2.1, we can consider larger graphs for larger choices of a.

Sulanke's Algorithm fully relies on the Complement Theorem 2.1, which is a double-edged sword for the Earth-Moon problem. On one hand, the theorem helps Sulanke's Algorithm allegedly to be the best algorithm for generating thickness 2 graphs with a high chromatic number. On the other hand it does not identify all thickness 2 graphs with a chromatic number of 10, if they do exist. For example, consider a hypothetical thickness 2 graph G with a chromatic number of 10, 17 vertices and no triangles in its complement. Applying the Complement Theorem to graph G, tells us it has a chromatic number of at least 9. Therefore, Sulanke's Algorithm would never recognize G as a graph with a chromatic number of 10. In the worst case, it might even be possible that thickness 2 graphs with a chromatic number of 10 exist, but none of them can be identified by the Complement Theorem. In this worst case, Sulanke's Algorithm will never identify a chromatic number 10 graph, despite the existence of such a graph.

Algorithm 1 Sulanke's Algorithm [1]

```
1: Let G_1 = (V, E_1), G_2 = (V, E_2) be two triangulated graphs on n given vertices.
2: Define G = (V, E_1 ∪ E_2)
3: Let k_max be the maximum number of flips.
4:
5: if G^c is triangle free then
6:     return G
7: for k = 0, ..., k_max do
8:     Randomly select a simple flip with edge e in G_1 or G_2 and its complementary edge ẽ
9:     if flipping e to ẽ increases the number of triangles in G^c then
10:        Flip the edge e to ẽ with a probability of P(G, G_ẽ, k).
11:    else
12:        Flip the edge e to ẽ
13:    if G^c triangle free then
14:        return G
15: return G
```

2.2 Randomizing flips

As of today, Sulanke's Algorithm has been unsuccessful in finding a thickness-2 graph with chromatic number 10, just as any other algorithm. For our research we want to maximize the efficiency of the algorithm to get a clearer view, whether the algorithm might be able to find a chromatic number 10 graph. Algorithm 1 can be split in three parts:

Line 1: Determine which two triangulated graphs are used to initialise the algorithm.

Lines 8-12: Select a random flip and do a flip.

Lines 5 and 9: Compute the number of triangles in the complement of the graph.

Unfortunately, we do not know the best way to implement these three parts, because that information is lost to time. We will discuss each part individually and construct an efficient implementation for each of them. Selecting random flips and doing flips is covered in this subsection, computing the number of triangles is covered in Subsection 2.3 and choosing the initial graphs is covered in Subsection 2.4.

Choosing a flip at random seems difficult and time consuming. Finding and selecting a random flip configuration as shown in Figure 4 is indeed no easy feat. Since finding these configurations is complex, we came up with a more efficient method of choosing a random flip. To do so, we start out with determining which flips are available. Remember that the flipping operation is made on a planar representation of a planar graph. Planar graphs have many planar representations, which makes listing all flips complicated. The list of possible flips can differ between planar representations. An example is shown in Figure 5 where edge (v_1, v_2) either flips into (v_3, v_4) or (v_3, v_5) depending on the planar representation of the graph.

Since Sulanke's Algorithm is restricted to triangulated graphs, we do not have to consider all planar graphs. Given a planar representation of a triangulated graph, there is an easy way of listing all flips of that representation. Since all faces are triangles, all edges are adjacent to exactly two triangular faces. Two triangular faces that share an edge is precisely the configuration that is required for a flip. I recommend taking a second look at Figure 4 to convince yourself of this fact. It follows, that each edge corresponds to one flip that uniquely depends on its two adjacent faces. Each edge is adjacent to two faces and those faces cover four different vertices. Two of those vertices are adjacent to the original edge and the remaining two edges are adjacent to the complementary edge, which is the edge we get by doing the flip. When we list all possible flips of a planar representation of a triangulated graph, we get a list of $|E| = m$ edges.

Flips are reminiscent of dual graphs. The dual of a planar representation is constructed by replacing

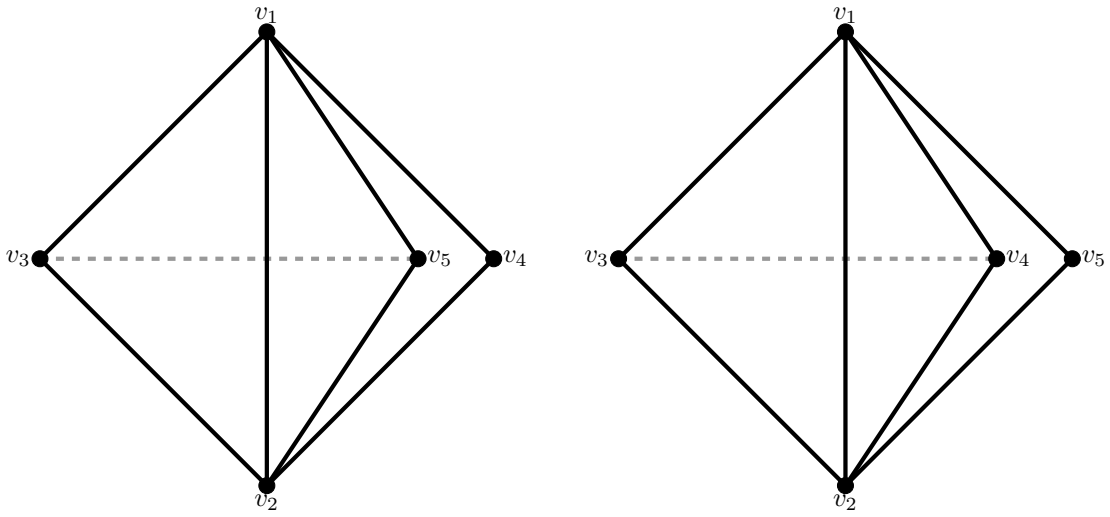


Figure 5: A planar graph which has different flips depending on the planar representation. The dashed line depicts the complementary edge of (v_1, v_2).

all faces with vertices and vice versa. Vertices are adjacent in the dual if their corresponding faces are adjacent in the original planar representation. Note, that graphs can have multiple different duals, because the dual depends on the planar representation. A relevant property of dual graphs is that they have exactly the same amount of edges as the original graph. Each edge in the original graph becomes an edge in the dual, which uniquely depends on the two adjacent faces in the original graph. The list of edges and their complementary edges and the list of edges and their dual edges is practically the same in triangulated graphs. The only difference is that the first list maps an edge to two vertices and the latter maps an edge to two faces. If we go back to Figure 4 we see that the flip maps edge e to the vertices v_3 and v_4. The dual maps e to the faces a and b, but from this pair of faces we can also derive that e has (v_3, v_4) as complementary edge, as v_3 and v_4 are the only two vertices in a and b that are not adjacent to e.

It follows that our list of flips and the list of dual edges is the same, besides a different notation. This also implies that all planar representations of a triangulated graph support the same set of flips if their dual does not depend on the planar representation. A dual does not depend on the planar representation, if the corresponding graph has a unique dual. A single planar representation of a graph would be sufficient for deriving all flips if the graph has a unique dual and it turns out that all triangulated graphs have this special property.

Theorem 2.3 *All planar representations of a triangulated graph G have a unique dual.*

Proof. There are two cases we have to consider. First, we examine all triangulated graphs with at most 3 vertices. These are the K_0, K_1, K_2 and K_3. For each of these graphs we manually check whether their duals are isomorphic. The K_0 and K_1 have themselves as the unique dual. The K_2 has a vertex with a self loop as unique dual and the K_3 has two vertices with a triple parallel edge as unique dual. All their duals are unique and thus the theorem is true for up to 3 vertices.

Second, we consider all triangulated graphs with 4 or more vertices. Hakimi and Schmeichel have proven in their article *On the connectivity of maximal planar graphs* [12] that triangulated graphs with 4 or more vertices are 3-connected. A graph is k-connected if the graph has more than k vertices and at least k vertices need to be removed before the graph becomes disconnected.

Hakimi and Schmeichel's theorem is helpful, because we can concatenate it with Whitney's theorem. He has proven that 3-connected graphs have a unique dual in his article *Congruent Graphs and the Connectivity of Graphs* [13]. These two theorems combined prove that all triangulated graphs with 4 or more vertices have a unique dual. Since we already proved the theorem for up to 3 vertices, the theorem is true for any triangulated graph. □

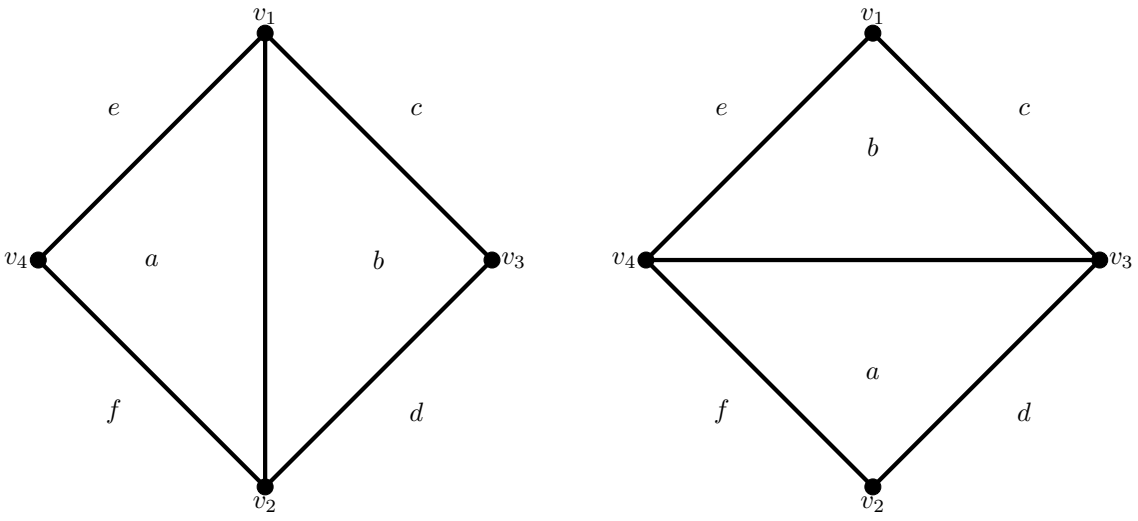
Together with our previous arguments, Theorem 2.3 implies the following corollary:

Corollary 2.4 *There are exactly $|E| = m$ possible flips in a triangulated graph $G = (V, E)$, one flip for each edge.*

This information is very useful for implementing Sulanke's Algorithm. Instead of searching for every single flip configuration and then choosing one at random, we can simply pick a random edge and derive its corresponding flip. With this strategy we came up with an algorithm that selects a random flip and performs it in constant time. Do note, that the setup at the start of the algorithm takes linear time. That is no problem for Sulanke's Algorithm though, because speeding up the repeating part of the algorithm is way more relevant. In our algorithm we represent the planarity of the graph with a list of faces. It is also possible to use the dual or a planar representation instead, but we do not provide an algorithm for that. Figure 6 provides a visual example of the algorithm.

Algorithm 2 Edge flip in constant time

1: Let $G = (V, E)$ be a triangulated graph.
2: Let F_V be a list of all faces and their adjacent vertices for a planar representation of G.
3: Let E_F be a list of all edges and their adjacent faces on the same planar representation.
4:
5: Randomly pick an edge $(v_1, v_2) = e$ from E_F.
6: Read the two faces a, b that are adjacent to e from E_F.
7: Read the missing two vertices v_3, v_4 surrounding edge e from F_V.
8: Flip $e = (v_1, v_2)$ to $\tilde{e} = (v_3, v_4)$ in G and E_F.
9: Update face a in F_V by replacing v_1 with v_3 for this face.
10: Update face b in F_V by replacing v_2 with v_4 for this face.
11: Update E_F by swapping a and b for the edges with a new adjacent face in F_V.



E_F	
edge	faces
(v_1, v_2)	a b
(v_1, v_3)	b c
(v_2, v_3)	b d
(v_1, v_4)	a e
(v_2, v_4)	a f

F_V	
face	vertices
a	v_1 v_2 v_4
b	v_1 v_2 v_3

E_F	
edge	faces
(v_3, v_4)	a b
(v_1, v_3)	b c
(v_2, v_3)	a d
(v_1, v_4)	b e
(v_2, v_4)	a f

F_V	
face	vertices
a	v_3 v_2 v_4
b	v_1 v_4 v_3

Figure 6: Example of an edge (v_1, v_2) flipped into edge (v_3, v_4) with Algorithm 2.

We have omitted one important problem that arises when selecting a random flip: not all flips are simple flips and we require flips to be simple for Sulanke's Algorithm. All Corollary 2.4 tells us, is that there are at most m possible simple flips in a triangulated graph. It is a recurring theme in this thesis that non-simple flips complicate things by a lot.

The best we can do, is to work around non-simple flips. To do so, we need to know what leads to non-simple flips. We can split these flips into two categories: flips that create a parallel edge and flips that create a self-loop. Self-loops are the easiest case, because they require a pair of triangles that share all their three vertices, a parallel face so to say. There is only one triangulated graph which has a parallel face, namely the K_3. Doing a flip in any other triangulated graph never creates a self-loop. There exist planar graphs besides the K_3 with flips that form self-loops, but those graphs are non-simple, making them non-triangulated too. Figure 7 shows two examples of how flips can create self-loops due to a parallel face being present in the graph.

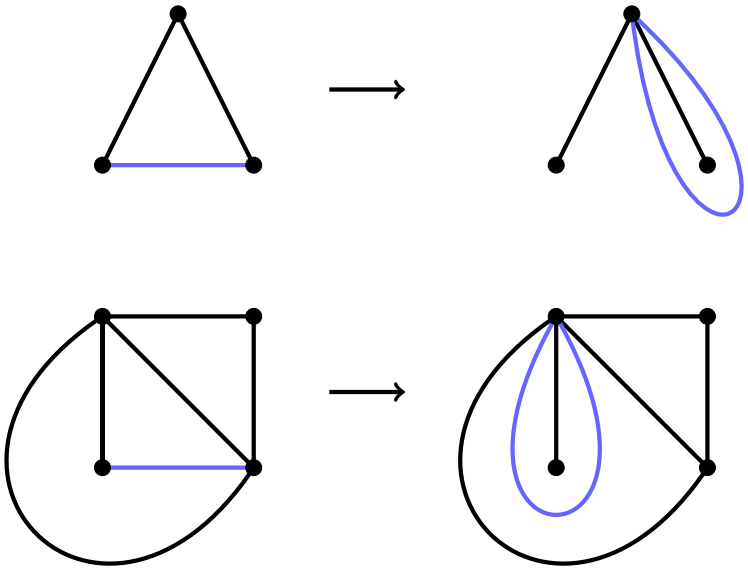


Figure 7: A triangulated graph (above) and a non-triangulated graph (below) that create self-loops when certain edges are flipped. The bottom left graph is not triangulated, because the graph is not simple.

If we restrict ourselves to triangulated graphs with 4 or more vertices, we never encounter a flip that creates a self-loop. This is barely a restriction, because small graphs are irrelevant for the Earth-Moon problem. This also helps with proofs later on, as we do not have to consider any situation where an edge has a self-loop as complementary edge.

The case where a flip results in a parallel edge is a lot more problematic. When doing many flips in a triangulated graph, one is bound to encounter a flip that creates a parallel edge and there is no elegant way of preventing this. Our solution is to simply select a different random flip, if the random flip we found creates a parallel edge. This compromise does not have a notable impact on the speed of the algorithm, because finding a random flip and its complementary edge can be done in constant time. It also helps that most flips in a triangulated graph are simple.

The configuration to do a simple flip is the K_4, but with a single edge missing. The configuration of a non-simple flip is precisely the K_4. We can approximate the probability of a flip being non-simple as follows: assume that we have a triangulated graph $G = (V, E)$ with n vertices and $3n - 6$ edges. We pick 5 edges that are in the right configuration for a (possible non-simple) flip, which is a subgraph in the form of the K_4 with 1 edge missing. This flip is non-simple if the 6th edge that would complete the K_4 is present in G. We assume that each vertex pair that does not contain one of these 5 edges, has an equal probability of containing one of the $3n - 1$ remaining edges. There are $\frac{1}{2}n(n-1) - 5$ of these vertex pairs. Under these assumptions, we get the this approximated probability for the occurrence of a non-simple flip:

$$\frac{\text{number of edges outside a flip}}{\text{number of vertex pairs outside a flip}} = \frac{3n-6-5}{\frac{1}{2}n(n-1)-5} \approx \text{probability of non-simple flip.}$$

We are not interested in running Sulanke's Algorithm on graphs with fewer than $n = 17$ vertices. At 17 vertices the probability of a flip being non-simple is 0.38 according to our rough approximation. This number gets even smaller when we increase the number of vertices. In our version of Sulanke's Algorithm, we repeatedly generate random flips with Algorithm 2, until we find a simple flip. Each time a non-simple flip is chosen, we waste time finding a different flip. This is not a problem in practice, because Algorithm 2 only takes constant time. The amount of time lost due to non-simple flips is negligible compared to the runtime of the entire algorithm.

2.3 Counting cliques

The second part of Sulanke's Algorithm that requires more clarification, is how we count the number of cliques with a fixed size a in the complement of the graph. A simple yet effective solution is to use the algorithm of Chiba and Nishizeki for finding cliques of fixed size $a \geqslant 3$. The algorithm and the details about the time complexity can be found in their article *Arboricity and Subgraph Listing Algorithms* [14]. Different algorithms with a similar time complexity can also be used instead, but we do not cover that here.

Algorithm 3 List all cliques of size $a \geqslant 3$ in a graph [14]

```
1: Let G = (V, E) be a graph.
2: Let C = ∅ be an empty set of edges.
3: Let S = ∅ be an empty set of cliques.
4: Label all vertices v ∈ V with a.
5: do K(a, V).
6: return S

7: procedure K(k, U)
8:     if k = 2 then
9:         for edge (x, y) in the subgraph induced by U do
10:            S := S ∪ ({x, y} ∪ C).
11:    else
12:        Sort all vertices v ∈ U from smallest to largest degree in the subgraph induced by U.
13:        for vertex v ∈ U in the order made in line 12 do
14:            Let U' ⊆ U be all vertices adjacent to v that have label k.
15:            Relabel all vertices u ∈ U' with k − 1.
16:            C := C ∪ {v}.
17:            do K(k − 1, U').
18:            C := C\{v}.
19:            Relabel all vertices u ∈ U' with k.
20:            Relabel vertex v with k + 1.
```

We denote $\gamma(G)$ as the arboricity of a graph G. This is the minimum number of forests required to cover the edges of G. Note that a forest is a set of disjoint trees. For example, all trees have an arboricity of 1. The K_4 has an arboricity of 2, because one tree cannot span the K_4, but two trees can, see Figure 8.

Algorithm 3 finds all cliques of size a in $\mathcal{O}\big(a\gamma(G)^{a-2}m\big)$ time in a graph with m edges. Thickness 2 graphs are sparse, which leads to a low arboricity. Conversely, the complement of thickness 2 graphs are dense and have a high arboricity.

Due to the arboricity being so high, we simply use the worst case bound $\gamma(G) \leqslant \left\lceil \frac{1}{2}\sqrt{2m+n} \right\rceil$ from Chiba and Nishizeki's paper for complements of thickness 2 graphs. We know the exact number of

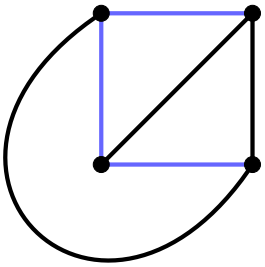


Figure 8: The K_4 split into a black and a blue tree, which shows that its arboricity satisfies $\gamma(K_4) \leqslant 2$.

edges m in the complement, because triangulated graphs with at least 4 vertices have $3n-6$ edges by Theorem 1.4. It follows that the complement has $m = \frac{1}{2}n^2 - \frac{1}{2}n - (3n-6)$ edges, which we can substitute in our worst case bound of the arboricity as shown in the next equation. This leads to a time complexity of $\mathcal{O}(an^a)$, when we want to find cliques of size a in the complement of a triangulated graph on n vertices.

$$\mathcal{O}(m) = \mathcal{O}\left(\frac{1}{2}n^2 - \frac{1}{2}n - (3n-6)\right) = \mathcal{O}\big(n^2\big)$$

$$\gamma(G) \leqslant \left\lceil \frac{1}{2}\sqrt{2\Big(\frac{1}{2}n^2 - \frac{1}{2}n - (3n-6)\Big) + n} \right\rceil, \text{ thus } \mathcal{O}\big(\gamma(G)\big) = \mathcal{O}(n)$$

$$\mathcal{O}\big(a\gamma(G)^{a-2}m\big) = \mathcal{O}(an^{a-2}n^2) = \mathcal{O}(an^a)$$

It turns out, that we can do better and count all cliques of size a even faster. Since we alter only a single edge in each iteration of Sulanke's Algorithm, the list of cliques in the complement stays mostly the same. Instead of counting the number of cliques from scratch in each iteration, we can use information from previous iterations. When we have a list of all cliques of size a and then change a single edge, we only have to count how many old cliques we lose and how many new cliques we gain. This idea is demonstrated in Algorithm 4.

Algorithm 4 List all cliques of size $a \geqslant 3$ in the complement after a flip

1: Let $G = (V, E)$ be a planar triangulated graph.
2: Let L be a list containing all cliques of size a in G^c.
3: Flip a vertex e into $\tilde{e}$ in G.
4:
5: Make a list L_- of all cliques containing $\tilde{e}$ in L.
6: Make a list L_+ of all cliques containing e in G^c.
7: $L = (L \backslash L_-) \cup L_+$
8: **return** L

We only have to list every single clique at the start, if we use Algorithm 4 in Sulanke's Algorithm. After the initialisation, we repeatedly update the list while the algorithm progresses. To compute the time complexity, we assume, that the size of the clique list is approximately constant. This is not necessarily true in practice, but this assumption keeps the time complexity simple. Line 5 of Algorithm 4 takes $a|L|$ time, because there are this many vertices in the list to check. The runtime of line 6 depends on the chosen algorithm.

Since we know that exactly one edge $e = (v, w)$ gets added to the complement, all newly formed cliques in that step contain edge e. This edge is adjacent to two vertices, which means we only need $a-2$ more vertices to complete a clique. For $a = 3$ this leads to a very fast algorithm to list all new

cliques. We only have to check, whether any vertices form a triangle with edge e in the complement. All these vertices form a clique of size three with the vertices v and w. To achieve this, we need to check all $2(n-2)$ possible edges that are adjacent to v or w, which has a time complexity of $\mathcal{O}(n)$.

For $a = 4$ we list all vertices that form a triangle with edge $e = (v, w)$ in the complement again. Then for each edge (x, y) in the subgraph induced by this list we get the newly formed clique $\{v, w, x, y\}$. Checking all the edges in the subgraph takes $\mathcal{O}(n^2)$ time.

We can generalize all cases for which $a \geqslant 5$ holds. We need to find all cliques of size $a-2$ that are fully connected to the vertices of e. We have $a - 2 \geqslant 3$, thus we can use Algorithm 3 by Chiba and Nishizeki to list all these cliques. We already computed that this algorithm has a time complexity of $\mathcal{O}(an^a)$ on the graphs we consider, which is the same time complexity as the other two cases $a = 3$ and $a = 4$.

All in all, the time complexity of Algorithm 4 depends on whether line 5 or 6 takes the longest. We simply take the maximum to get a final time complexity of:

$$\mathcal{O}\Big(\max\big\{an^a, a|L|\big\}\Big).$$

Computing the number of cliques is the most time consuming part of Sulanke's Algorithm, since this has to be done every single iteration. Finding and doing a flip is also done every iteration, but that takes constant time, which has a negligible impact on the runtime.

We now have all the information required to compute the time complexity of Sulanke's Algorithm. Denote the number of vertices by n, the size of cliques to eliminate in the complement by a and the total number of attempted flips by b. Let $|L|$ be the number of cliques in the complement, which we assume to be mostly constant during the algorithm. This leads to the follwing time complexity of Sulanke's Algorithm:

$$\mathcal{O}\Big(\max\big\{ban^a, ba|L|\big\}\Big).$$

Note that, we are only interested in running the algorithm on graphs, for which it is realistic to eliminate all cliques of size a from the complement. This leads to very small values of $|L|$, while running the algorithm. As long as the algorithm is used for its intended purpose, the time complexity can be simplified to $\mathcal{O}\big(ban^a\big)$.

2.4 Initial triangulated graphs

We need two triangulated graphs to initiate Sulanke's Algorithm, which raises the question which graphs we should pick. Most importantly, we want to know whether our choice for the two initial graphs locks us out of possible outcomes. All triangulated graphs on the same number of vertices look very similar to each other, but they do differ nonetheless. Figure 9 on page 18 depicts the two triangulated graphs G_0 and H_0, which are not isomorphic. The easiest way to see that they are not isomorphic, is by counting the degrees of the vertices. All vertices have a degree of 4 in G_0, while only vertices v_1 and v_2 have a degree of 4 in H_0 Therefore these two graphs are not isomorphic. Do note that 6 is the smallest number of vertices n, that supports non-isomorphic triangulated graphs. For $n \leqslant 5$ fixed, all triangulated graphs on n vertices are isomorphic. Furthermore, for $n \leqslant 4$ fixed, all triangulated graphs on n vertices are equal.

Now that we know, that triangulated graphs can be distinctively different, we want to know whether we can obtain all triangulated graphs, given a finite number of flips. It turns out that, if we start with a triangulated graph on n vertices, we can obtain any other triangulated graph on n vertices using simple flips. We first prove, that we can obtain a triangulated graph isomorphic to any other triangulated graph given a finite number of simple simple flips. Second, we prove the stronger version of the theorem in which we obtain equal graphs, rather than isomorphic graphs.

Theorem 2.5 *Let be given two triangulated graphs G and H on the same vertex set V and a planar representation of both graphs. We can transform graph G into graph G' via a finite sequence of simple flips on G, such that $G' \simeq H$ holds.*

Proof. Given that the vertex set V has 4 or less vertices, the problem becomes trivial. There is only one triangulated graph on such a vertex set V. Thus, $G = H$ holds by default and no flips are required. For the remainder of the proof we assume, that $|V| \geqslant 5$.

We will give a proof by construction. Our goal is to construct a graph G_T, by flipping vertices in G, such that $G_T \simeq H$. We create G_T step by step, starting from graph $G = G_0$. We track the graphs we make in the sequence $G_0, G_1, G_2, ..., G_T$. We also make use of a second graph sequence that starts with graph $H_0 = H$. Furthermore, we create a sequence of subgraphs and a sequence of cycles. All four sequences are detailed here:

- $G_0, G_1, G_2, ..., G_T$
  - G_{i+1} is either equal to G_i or G_{i+1} is obtained from G_i by simple flips.
- $H_0, H_1, H_2, ..., H_T$
  - H_{i+1} is obtained from H_i by a permutation of the vertices, thus $H_i \simeq H_{i+1}$ holds.
- $G_0^S, G_1^S, G_2^S, ..., G_T^S$
  - This sequence of connected subgraphs starts with $G_0^S = \varnothing$.
  - The subgraph G_i^S contains i triangular faces and is present in both G_i and H_i.
  - The subgraph G_{i+1}^S is obtained by adding an adjacent triangular face to G_i^S, thus $G_i^S \subseteq G_{i+1}^S$ holds.
- $C_0, C_1, C_2, ...C_T$
  - C_i is the cycle along the boundary of G_i^S.

All graphs G_i and H_i are triangulated and have $|V|$ vertices. Therefore, all these graphs have exactly $T = 2n - 4$ faces. (This follows from Theorem 1.4 and Eulers Formula.) If we manage to construct the four sequences with the given properties, we end up with the graph G_T^S. This graph is a subgraph of G_T and H_T. By construction, it has T triangular faces, thus G_T^S consists of all faces in G_T and all faces in H_T. It follows, that $G_T^S = G_T = H_T$. Since the graphs G_i are created by doing simple flips in G_0 and the graphs H_i are created by permutations of the vertices in H_0, we get the following relation between the first and last graphs in the sequences:

$$G_0 \xrightarrow[\text{of flips}]{\text{finite number}} G_T = H_T \simeq H_0.$$

This completes the proof. Basically, we are turning the graphs G_0 and H_0 into each other, one face at a time. In the i-th step, we have the two graphs G_i and H_i, that both contain the subgraph G_i^S. Because G_i^S consists of i faces, the graphs G_i and H_i have (at least) i faces in common.

i = 0, 1, 2
The first two steps of the construction, where we make the graphs G_1, H_1, G_2 and H_2, are the easiest to construct. An example is shown in Figure 9. In G_1 and H_1, we make the outer faces of the graphs equal, by permuting vertices in H_0. To construct G_2 and H_2, we make sure both graphs contain the face v_2, v_3, v_5. We only have to permute vertex v_5 and v_6 in H_1 to achieve this.

The subgraphs G_1^S and G_2^S consist of the equal faces in the graphs. These subgraphs are precisely the outside areas of C_1 and C_2 respectively. We always colour cycles of sequence $C_0, ..., C_T$ red in our examples. To make sure that the equal faces we have created are preserved, we never flip any edge or permute any vertex in subgraphs G_i^S. For example, in graph G_2 of Figure 9, we cannot flip any of the red edges, nor the edge (v_2, v_3). In graph H_2, we cannot permute the vertices v_1, v_2, v_3 and v_6.

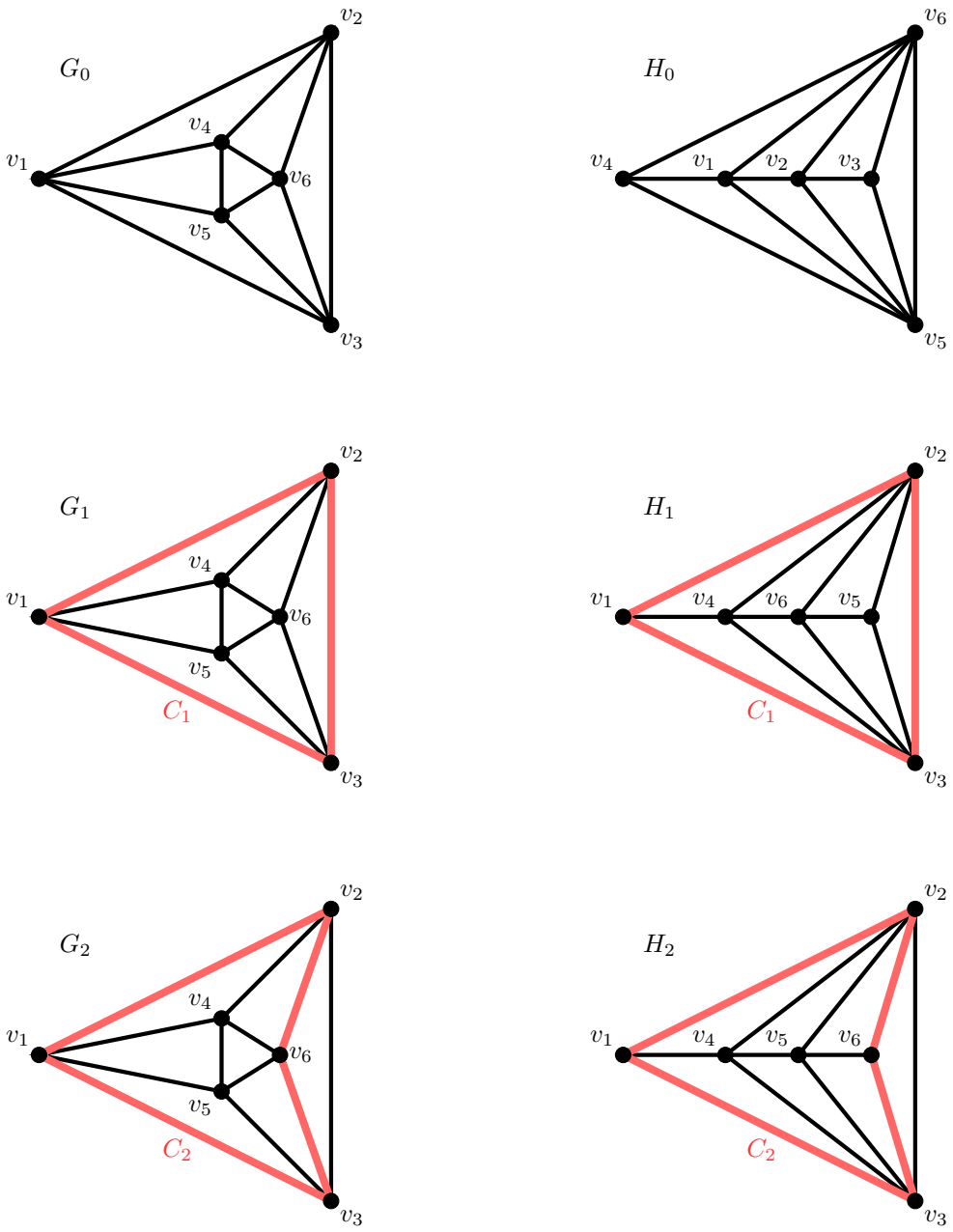


Figure 9: The construction of graphs G_1, H_1, G_2 and H_2.

For any pair of graphs G_0 and H_0, we can do the first two steps, by permuting at most 4 vertices in H_0. No edge flips are required and thus, $G_0 = G_1 = G_2$ holds.

i = 3, 4, ..., T

From $i = 3$ onwards, the construction becomes more complicated. Each iteration we choose an edge (x, y) of the cycle C_i. In graph H_i, the edge (x, y) is adjacent to two faces: one face that is in G_i^S and one that is not. The face that is not in G_i^S is the face we are interested in. We will call this face x, y, z, where z is the third vertex of this face. Next, we flip vertices in G_i and swap vertices in H_i, to make the face adjacent to (x, y) equal in the two graphs. We add this new face to G_i^S and repeat the process.

The exact flips and permutations we do, depends on the location of vertex z in the graph H_i. There are three cases we have to consider, which are depicted in Figure 10.

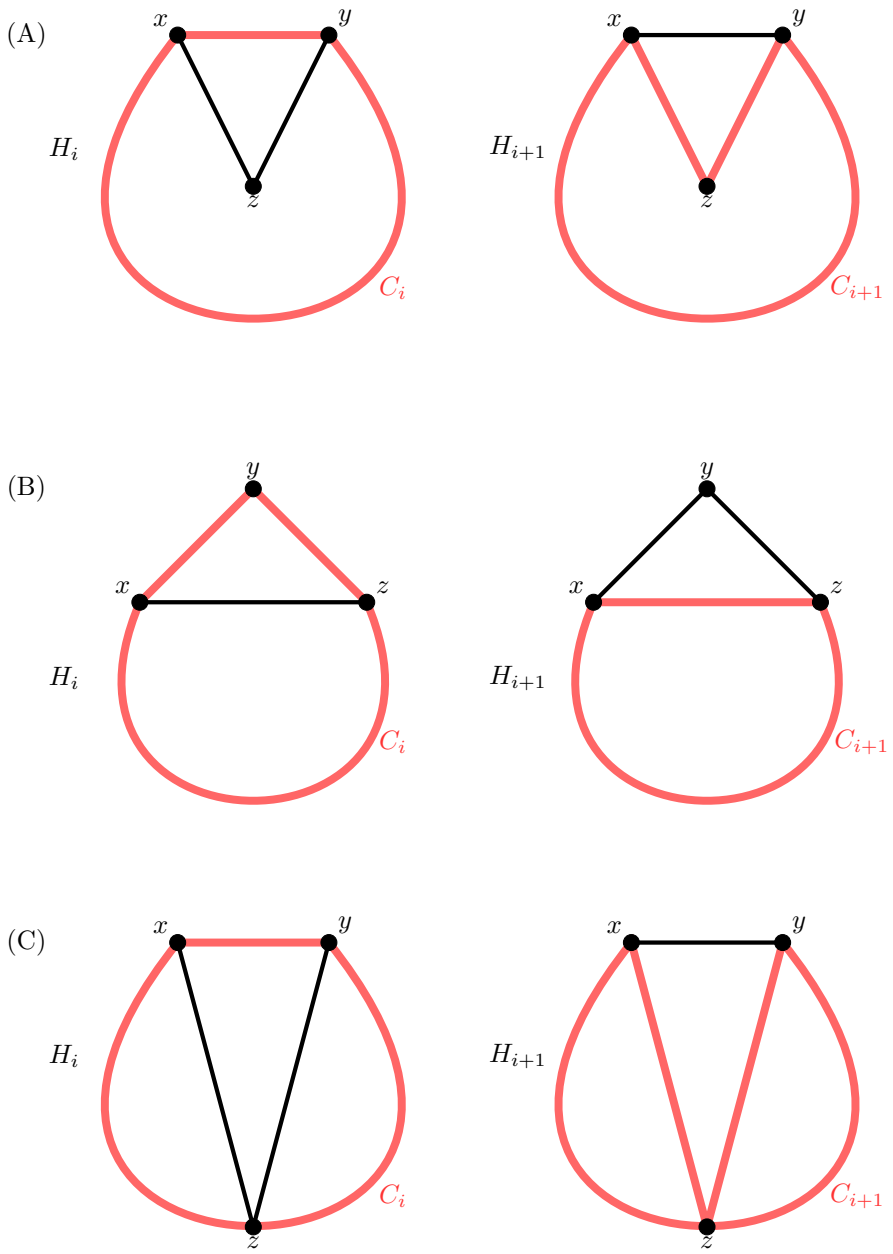


Figure 10: All three cases for face x, y, z in H_i (left) and what graph H_{i+1} would look like if face x, y, z is moved to the outside of C_i to create C_{i+1} (right).

(A) Vertex z is inside the cycle C_i, but not on C_i.

(B) Vertices x, y, z are three consecutive vertices on the cycle C_i.

(C) Vertex z is on C_i, but x, y, z are not consecutive vertices on C_i.

Case (C)

Case (A) and (B) are perfectly fine for our proof. Case (C) however, is the problematic case, which we discuss first. When face x, y, z is moved to the outside of the cycle C_i, we get a path C_{i+1} that is not a cycle, whenever it is a case (C) face. This is shown in Figure 10. This is a problem, because we require that the sequence $C_1, ..., C_T$ solely consists of cycles. This requirement is relevant for case (A) faces.

If edge (x, y) leads to a case (C) face, we must find a different edge in C_i that leads to a case (A) or (B) face. To continue our construction, we must show that there always exists an edge (x, y) on C_i that creates a case (A) or (B) face in H_i.

Let $(x_1, x_2) \in C_i$ be an edge that forms a case (C) face with vertex z_1, as shown in Figure 11. If this

happens, we can select the next edge (x_2, x_3) counter-clockwise along C_i to get a new face. If this is a case (A) or (B) face, we are done. Else, the face adjacent to (x_2, x_3) contains a third vertex z_2 that is on the cycle C_i. Either $z_2 = z_1$ holds, or z_2 is a new vertex, that appears clockwise from z_1 along C_i. If z_2 would be located counter-clockwise from z_1 along C_i, the two faces x_1, x_2, z_1 and x_2, x_3, z_2 cross, which is not possible in the planar graph H_i.

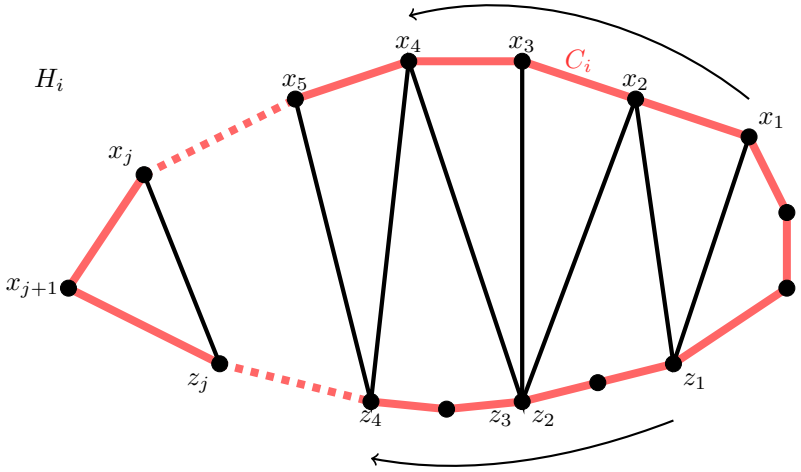


Figure 11: Visual proof that not all edges in C_i can lead to case (C) faces.

If we repeat this procedure and only find case (C) faces, we observe that the number of available vertices along C_i decreases. The vertices x_j and x_{j+1} for $j \geqslant 1$ move counter-clockwise along C_i and the vertices z_j for $j \geqslant 1$ are either stationary or move clockwise along C_i. Since there are a finite amount of edges in the cycle C_i, the vertices x_{j+1} and z_j will eventually meet up, when j large enough. When this happens, we find the face x_j, x_{j+1}, z_j, consisting of three vertices that appear consecutively on C_i. This is a case (A) face. Thus, not all faces along C_i are case (C) faces.

All that is left to prove, is that we can always resolve case (A) and case (B) faces. We do this by creating the graphs G_{i+1}, H_{i+1} and subgraph G^S_{i+1}, from the graphs G_i, H_i and subgraph G^S_i respectively. We start out with case (A).

Case (A)
We have the triangular face x, y, z in H_i and this face is inside C_i. Vertices x and y are on C_i and vertex z is not. In G_i, the edge (x, y) is also adjacent to a face inside C_i. We call this face x, y, v. If we can permute vertex v and z in H_i, we can make the two faces equal. Then, the faces are ready to be added to G^S_i and the case (A) face is resolved.

Sometimes, we cannot permute vertex v. This only happens when v is on the cycle C_i. When this happens, we want to do flips in G_i, such that either the face x, y, z or x, y, w is in G_i for some permutable vertex $w \in G^S_i$. If we end up with the face x, y, w in G_i, we can permute w and z in H_i. That permutation creates the face x, y, w in H_i, creating a face that matches with the face in G_i.

When v is not permutable, we attempt the most straightforward solution first: doing flips to add the face x, y, z to G_i. We can create a sequence of flips in G_i, that would create the face x, y, z, with the help of a path P along the dual of G_i. A path along the dual is a path that traverses faces, rather than vertices. Let P be a path with the following properties:

- P starts in face x, y, v.
- P ends in a face that contains vertex z. This is the only face in P that contains z.
- P contains no face of the subgraph G^S_i.

As long as any face adjacent to z and the face x, y, v are in the same connected component of the dual, there always exists such a path P. Fortunately, all faces inside C_i form one connected component

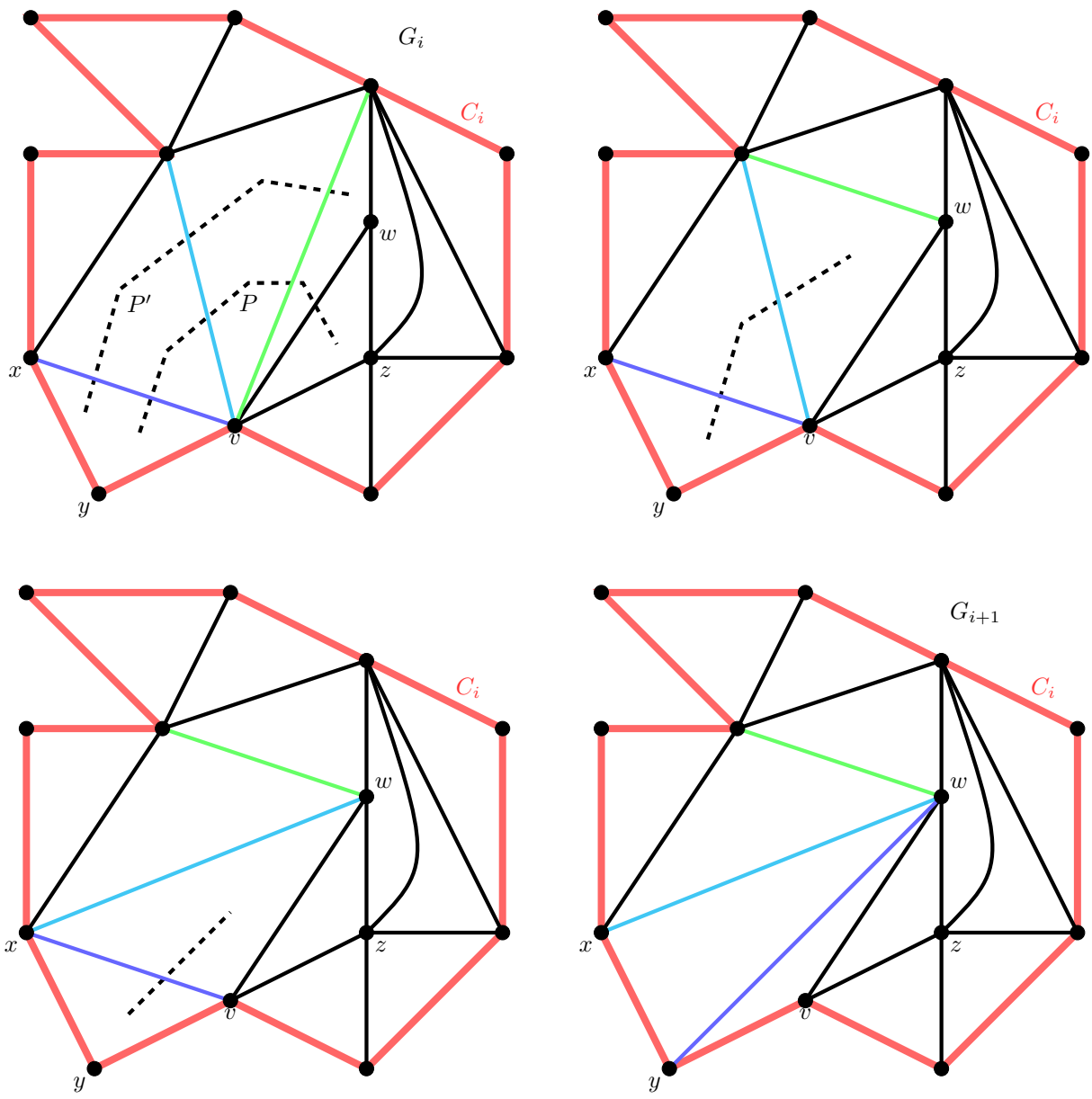


Figure 12: Example of case (A). Flipping all vertices along path P' in G_i creates the face x, y, w. Flipping all vertices along path P creates the face x, y, v, but the first flip creates an edge parallel to the curved edge.

in the dual, as long as C_i is a cycle. This is the reason why resolving case (C) faces is problematic. When a case (C) face is added to G_i^S, the inside of C_i gets split into two disconnected components in the dual, which is shown on the bottom right of Figure 10.

Flipping all edges crossed by path P in order, starting from the face that contains z, creates the face x, y, z. Figure 12 shows an example of flipping edges along a path on the dual. Note that, the edges along path P' are flipped and not P in the example. Path P' is used a bit later in the construction.

The figure also shows, that all edges along the path are adjacent to the same vertex after they are flipped. Flipping edges along path P' makes all edges adjacent to w and flipping edges along path P makes all flipped edges adjacent to z. Hence the faces x, y, w and x, y, z are obtained respectively, depending on which path is chosen.

We are not done yet, because path P might lead to non-simple flips. For example, if we flip vertices along path P in Figure 12, we run into a non-simple flip. The very first flip creates an edge parallel to the curved edge. To solve this, we create a new path P', such that all flips along path P' are simple.

Walk along path P, starting from the face x, y, v, and note all faces we visit. Stop at the first face visited by P, that contains a vertex $w \notin C_i$. Denote the resulting path by P'. It is possible that the path P and P' are equal. If that happens, $w = z$ holds. The shorter path P' contains only one face that does not have all its vertices in C_i, namely the last face along path P'. Only vertex w of this last face is not on C_i. We use this fact to prove, that flips given by path P' are all simple.

Each flip of edge e into $\tilde{e}$ along P', uses two faces. Before the flip, only one face has a vertex that is not on C_i, namely the vertex $w \notin C_i$. See Figure 12. e consists of two vertices on C_i and $\tilde{e}$ has only one vertex on C_i. This exact configuration guarantees that the flip is simple. Flipping edges along path P' guarantees that each flip has this configuration, which is not true for path P.

Once all edges along P' are flipped, we obtain the face x, y, w in G_i. We permute w and z, to create the face x, y, w in H_i. Lastly, we add face x, y, w to G_i^S and update cycle C_i.

Case (B)

In case (B), all three vertices x, y, z are in C_i. The edges (x, y) and (y, z) are in the subgraph G_i^S. This means that we only have to make sure the edge (x, z) is in graph G_{i+1}, to create a face that is in both graphs. A generalisation of this case is shown in Figure 13. If edge (x, z) is already in G_i, we get a so called closed fan as shown on the right of the figure.

If the edge (x, z) is not in G_i, then y has at least two adjacent faces within C_i, as shown on the left side of Figure 13. We call all these faces together an open fan, which we want to turn into a closed fan with flips. If we flip all the centre edges in an open fan, we get the closed fan we want. The edges we want to flip are highlighted in blue in Figure 13. Every time one of the centre edges is flipped, the number of faces in the fan decreases by one. The fan closes when all the centre edges are flipped, independent of the order in which they are flipped. Nonetheless, the order should still be chosen carefully to prevent non-simple flips.

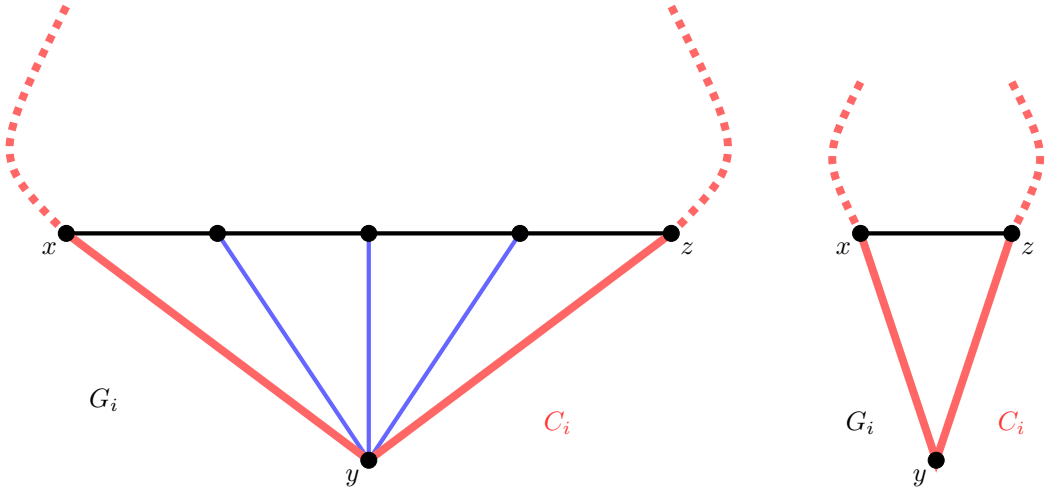


Figure 13: Closed fan (left) and open fan (right) in the graph G_i, which are used to solve case (B).

There is a strategy to prevent non-simple flips, when closing the fan. Let (y, v) and (y, w) be two edges in the fan that share a face, but neither edge contains vertex x or z, as shown on the left of Figure 14. (y, v) and (y, w) are of the two edges we must flip to close the fan and are therefore highlighted in blue in the figure. Unfortunately, it is possible that flipping (y, v) or (y, w) creates a parallel edge together with an edge outside of the fan. However, if both of these adjacent edges would form a parallel edge when flipped, the graph is no longer planar due to the presence of the two conflicting edges. These two hypothetical problematic edges are depicted with dashed lines.

If neither of the blue edges in Figure 14 correspond to a simple flip, both dashed edges must be present in G_i. Then, G_i cannot be planar, which is a contradiction. It follows, that at least one of the two blue edges correspond to a simple flip.

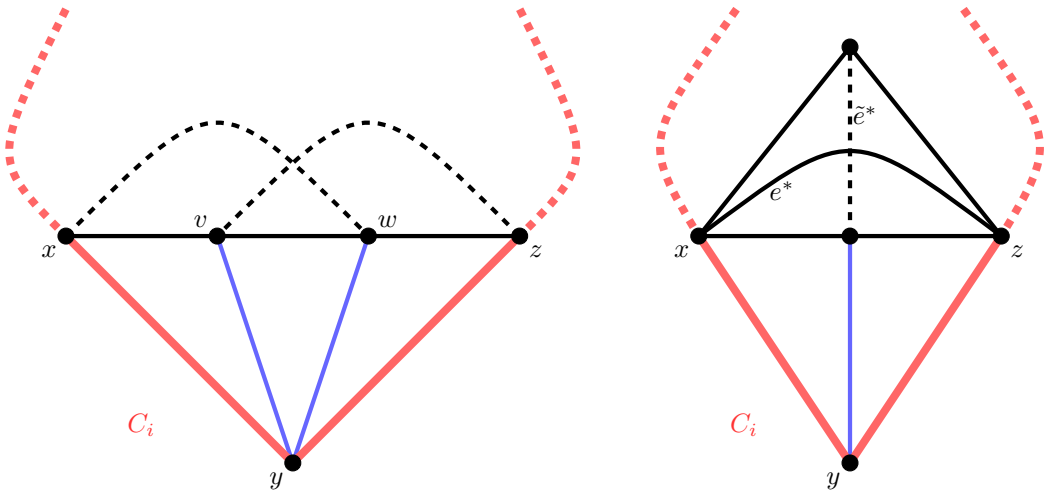


Figure 14: Two examples in which the black dashed edges would make the graph non-planar. Triangles represent faces and non-triangular areas consist of at least one face.

Fans that contain three or more faces always contain two edges that are adjacent in this way, thus we can always flip an edge with a simple flip to make fans smaller. By repeatedly flipping edges in an open fan this way, we eventually end up with a fan with exactly two faces. Only one more edge needs to be flipped to close the fan. If the last flip is a simple flip we are done, because we can simply perform the last flip.

Call the last edge that needs to be flipped e, which flips into the edge $\tilde{e} = (x, z)$, as shown on the right side of Figure 14. If this last flip is non-simple, there must be an edge $e^* = (x, z)$ outside the fan. If $e^* \in G_i^S$ holds, e^* is also present in the graph H_i. However, the edge $\tilde{e}$ is also in graph H_i. This leads to the contradiction of H_i containing the two parallel edges $\tilde{e}$ and e^*.

Thus, if the last flip is non-simple, the conflicting edge e^* must be within C_i. It follows, that we can attempt to flip the edge e^* in G_i. Let $\tilde{e}^*$ be the complementary edge of e^*. Figure 14 shows the exact configuration the edges e^* and $\tilde{e}^*$ have in this situation. From the figure, it is clear that an edge parallel to $\tilde{e}^*$ would make G_i non-planar, thus there does not exist an edge parallel to $\tilde{e}^*$. Therefore, we can do the simple flip that replaces vertex e^* with $\tilde{e}^*$. This makes the flipping e a simple flip and we can close the fan, which completes case (B).

There is one special version of case (B), which is encountered in the second to last graphs G_{T-1} and H_{T-1}. There is only a single face x, y, z within the cycle C_{T-1}. This means, that all edges and vertices of G_{T-1} and H_{T-1} are fixed. We cannot do any flip or swap any vertex. This also means, that the face x, y, z is already present in both G_{T-1} and H_{T-1}. No flips or vertex swaps are required.

This concludes the construction of the sequences $G_0, ..., G_T$, $H_0, ...H_T$, $G_0^S, ..., G_T^S$ and $C_0, ..., C_T$. Specifically, we have constructed the graph G_T by performing simple flips, such that $G_T \simeq H_0$ holds. This proves, that there does indeed exist a finite sequence of flips that makes graph G_0 isomorphic to H_0. □

Next we prove the stronger version of Theorem 2.5. Instead of making isomorphic graphs, we want to make equal graphs. Only then can we be sure, that Sulanke's Algorithm can create any triangulated graph on n vertices, no matter which two initial graphs on n vertices are chosen.

Theorem 2.6 *Given two triangulated graphs G and H on the same vertex set V and a planar representation of both graphs. We can transform graph G into graph G' via a finite sequence of simple flips on G, such that $G' = H$ holds.*

Proof. Let two triangulated graphs G and H on vertex set V be given and denote $n = |V|$ as the number of vertices. With the proof of Theorem 2.5, we can create the graph G' by doing simple flips

in G, such that $G' \simeq H$ holds. Denote π as the permutation on V, for which $\pi(G') = H$ holds.

First, we will show that we can perform some vertex permutations in G' by doing simple flips. This means that we can create a new graph G'', using simple flips on G', such that $G' \simeq G''$ holds. Then, we expand the number of permutations we can do in G' with flips, so that we can eventually express π as a sequence of flips on G'.

We start with the most simple permutation in G'. Let G_0 be a triangulated graph on n vertices that contains the two adjacent vertices v_1 and v_2. Let v_1, v_2, v_3 be a face of G_0. We will construct a sequence of simple flips that permutes v_1 and v_2. Such a permutation is denoted as $(v_1 v_2)$.

Let H_0 be a trivial graph on n vertices as seen in Figure 15. The trivial graph has the interesting property, that it contains two vertices that are connected to all other vertices, which we call the two high degree vertices. Let these vertices be denoted by v_1 and v_2. Let $v,1, v_2, v_3$ span a face in H_0, as seen Figure 15. Next, we apply the construction in the proof of Theorem 2.5 to the graphs G_0 and H_0. Thus, we do simple flips in G_0 to create the graph G_T and permute vertices in H_0 to create the graph H_T, such that $G_T \simeq H_T$ holds. It is important that we never permute vertex v_1 or v_2 in H_T.

We do the first step of the construction manually, to make sure v_1 and v_2 are not permuted. We choose our graph H_0 in such a way that G_0 and H_0 both contain the face v_1, v_2, v_3. We add this face to the subgraph G_1^S, which fixes the vertices v_1, v_2 and v_3. This guarantees we never swap the vertices v_1 and v_2 in any of the graphs H_i. The first step of the construction is now completed, without doing any flips or permutations. We continue the remainder of the construction as normal.

The final result is a sequence of flips that turns G_0 into the trivial graph $G_T = H_T$. Because we never permuted vertex v_1 or v_2, they are still the two high degree vertices of H_T, which makes them the high degree vertices of G_T too. Because v_1 and v_2 are connected to all edges, they are symmetric in the trivial graph G_T, see Figure 15. This allows us to undo all the flips we did to get graph G_T, but with vertex v_1 and v_2 permuted in all flips. After undoing all flips in this manner, we get graph G_0 again, but with vertex v_1 and v_2 permuted. All in all, we have found a sequence of flips in G_0 that permutes the two adjacent vertices v_1 and v_2.

Multiple permutations of adjacent vertices, allows us to permute non-adjacent vertices too. Since triangulated graphs are connected, there always exists a path between any two vertices. Let v_0 and v_N be two vertices in a triangulated graph and let $v_0, v_1, v_2 ..., v_{N-2}, v_{N-1}, v_N$ be a path of length N between these two vertices. Consider the following product of permutations σ:

$$\sigma = \Big((v_0 v_1)(v_1 v_2)...(v_{N-2} v_{N-1})(v_{N-1} v_N)(v_{N-2} v_{N-1})...(v_1 v_2)(v_0 v_1)\Big).$$

σ solely consists of permutations between adjacent vertices and the entire product is equal to the permutation (vw). It follows that, we can permute any pair of vertices in a triangulated graph with simple flips.

A well known theorem from algebra is, that any permutation can be expressed as a product of permutations that permute two elements. Specifically, the permutation π we want to perform with simple flips on G', can be expressed as a product of permutations that permute two vertices. Since each permutation of two vertices can be done with simple flips, permutation π can also be executed with simple flips. We conclude that we can turn any triangulated graph G into any other triangulated graph H on the same vertex set by only doing simple flips in G. □

From Theorem 2.6 it follows that the starting condition for Sulanke's Algorithm is not too important, since all graphs are obtainable by the two initial triangulated graphs. We still have to consider which initial graphs we choose. Since we want to generate a thickness 2 graph with few cliques in its complement, we prefer that the initial thickness 2 graph also has few cliques in its complement. Thus, a simple way to initiate Sulanke's Algorithm is to run another instance of Sulanke's Algorithm. We can use two different trivial triangulated graphs for the first instance of the algorithm and we can use the resulting output as the initial state for the second instance.

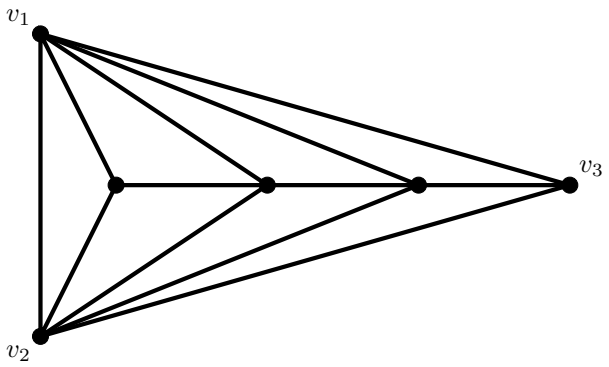


Figure 15: The trivial triangulated graph for $n = 6$ with v_1 and v_2 as symmetric vertices.

A different strategy would be to start with two different triangulated graphs again, but rather than running the algorithm twice, we run it once but with more iterations. Testing this strategy has shown that it performs worse. On the other hand, increasing the number of times we run the algorithm, at the cost of having less iterations for each run, turns out to improve the performance. The best ratio of how often we run the algorithm and how many iterations each algorithm has, depends on the other variables used in the algorithm.

Using two trivial graphs as initial state is far from optimal. We did not improve our choice for the initial state, because our tests have shown that Sulanke's Algorithm is extremely fast in removing the first lot of cliques in the complement. Far more time is required to remove cliques, when there are few cliques left in the complement. This difference is so big, that the amount of time saved is negligible when using a better initial state. From our experience of running the algorithm, starting with two trivial triangulated graphs works. For readers who are interested in more optimized initial states, I recommend van der Beek's thesis on the Earth-Moon Problem [1]. He proposes the use of so called *permuted layer graphs* to generate a random initial state, instead of always using the same initial graphs as we do.

It is important to note that the initial state is not completely irrelevant. When picking two equal triangulated graphs, we have to remove half of the edges when merging them in the thickness 2 graph. This adds a lot of edges to the complement, which tremendously increases the number of cliques in the complement. Such an initial state is ill-advised.

2.5 Adapted Algorithm

So far, we have constructed fast and explicit methods for the important parts of Sulanke's Algorithm. There are still some details left to fill in, before we can combine everything we learned into an adapted version of the algorithm.

In Subsection 2.2 we observed that doing random flips in a triangulated graph can create parallel edges. To circumvent this problem, we keep selecting a new random flip until we find a non-simple flip. However, there is a second way to get undesired edges. We have two triangulated graphs that form a thickness 2 graph when combined. Sometimes the triangulated graphs both contain the same edge. When this happens, we only add one of these edges to the thickness 2 graph. If we add both edges to the thickness 2 graph, we would get an undesired parallel edge.

At first this sounds like a perfectly fine solution. When we lose an edge while merging, due to an edge being present in both triangulated graphs, it will only increase the number of edges in the complement. More edges in the complement leads to more cliques in the complement, thus the odds of accepting a flip that creates a parallel edge is at a disadvantage. After doing the last flip in the algorithm, we prefer that there are few to none of these parallel edges, but in practice we often end up with quite a lot of them.

We can improve the algorithm by adding a more direct method of discouraging parallel edges, that

are present in both of the two triangulated graphs. We could reject every single flip that creates a parallel edge. But if we do that, we cannot say for sure whether we can still make all triangulated graphs, because we cannot apply Theorem 2.6 any more. Instead, we propose to give the addition of such a parallel edge the same weight as the addition of one clique in the complement. Thus, a flip that creates an edge that is already present in the other triangulated graph, is just as likely to be rejected as a flip that adds one additional clique. This change increases the performance of Sulanke's Algorithm. Do note, giving a parallel edge the same weight as one clique is an arbitrary choice. A different weighting might be better, but finding the best weight would require thorough testing.

We also have to consider the probability function that accepts unfavourable flips. The function we ended up using is the best performing one from van Valkengoed's research on implementing Sulanke's Algorithm [11]. The best performing probability function from her research is one of the most commonly used functions for simulated annealing. Let b be the total number of iterations we want to do and j the iteration we are currently at. Assume the flip proposed in iteration j increases the number of cliques by $x > 0$. Then, the probability of accepting this flip is as follows:

$$\exp^{\left(\frac{-x}{1-j/b}\right)}.$$

There are differences between our and van Valkengoed's version of Sulanke's Algorithm. For example, in our version of the Algorithm x is increased by 1 if the flip in iteration j creates a parallel edge between the two triangulated graphs. All the other changes to the algorithm we mentioned earlier, further differentiate our algorithms. The biggest difference is that van Valkengoed's implementation did not prioritize a fast runtime. With a lot of help from van Valkengoed's version of Sulanke's Algorithm, we managed to make our own version that is designed for having a high performance. Our algorithm is approximately 21,000 times faster than van Valkengoed's implementation. This creates the problem that her data on which probability function works best, has become negligible compared to the amount of data our new algorithm can generate. Together with the fact that our algorithm is also defined slightly differently, we conclude that more research is required to determine which probability function is best for our adapted version of Sulanke's Algorithm.

This covers all changes we made to create our own version of Sulanke's Algorithm, which we call the adapted version of Sulanke's Algorithm. The exact algorithm is shown in Algorithm 5.

Algorithm 5 shows our final algorithm that contains everything we have discussed in this section. We have also tried other ideas to improve the algorithm, but all of these got scrapped. We list them here to give some insight in our development of the altered version of Sulanke's Algorithm.

- Reject flips that create vertices with a degree below 8 or 9 in the thickness 2 graph.
  - *Vertices with a low degree are not directly relevant for the chromatic number in the graphs we consider.*
- Reject flips that reduce connectivity.
  - *In Subsection 3.1 we discuss why a high connectivity might be favourable.*
- Reject flips that reduce the approximated chromatic number.
  - *Our approximations of the chromatic number are not very accurate. Therefore they poorly indicate whether a flip is favourable.*

These ideas slowed down the algorithm due to being slow to compute or due to having a negligible impact on the algorithm's outcome. Most of our scrapped ideas were both slow and had a negligible impact on the outcome.

Algorithm 5 Adapted version of Sulanke's Algorithm

1: Let $n \geqslant 5$ be a number of vertices.
2: Let $q \geqslant 3$ be a clique size.
3: Let $a \geqslant 0$ be the maximum number of times we run Sulanke's Algorithm.
4: Let $b \geqslant 0$ be the maximum number of iterations we run each Sulanke's Algorithm.
5:
6: Let $G_1 = G_2$ be two adjacency matrices of the trivial triangulated graph on n vertices.
7: Let $E_1 = E_2$ be the edge list of the trivial triangulated graph that also contains the two adjacent faces for each edge.
8: Let $F_1 = F_2$ be the face list of the trivial triangulated graph that also contains the three adjacent vertices for each face.
9: Randomly shuffle the vertices of the second graph and change G_2, E_2 and F_2 accordingly.
10: Define $G_0 = (V, E_1 \cup E_2)$.
11: List all cliques of size q in G_0^c in the list Q.
12: Define $\mathcal{G} = \mathcal{H} = (G_0, Q, G_1, E_1, F_1, G_2, E_2, F_2)$.
13: Define $|Q_\mathcal{G}| = |Q_\mathcal{H}| = |Q|$.
14: **if** $|Q_\mathcal{G}| = 0$ **then**
15: **return** $\mathcal{G}$ *Graph G_0 has the desired number of cliques and is outputted.*
16: **for** $i = 0, ..., a$ **do**
17: **do** $|Q_\mathcal{G}| := S(\mathcal{G}, q, |Q_\mathcal{G}| - 1, b)$. *Do Sulanke's Algorithm, which ends after b iterations or when a graph is found that has fewer cliques than our current best graph.*
18: **if** $|Q_\mathcal{G}| \leqslant |Q_\mathcal{H}|$ **then**
19: **if** $|Q_\mathcal{G}| \leqslant 0$ **then**
20: **return** $\mathcal{G}$
21: $\mathcal{H} := \mathcal{G}$ *Overwrite the saved graph with the new graph, because the new graph is at least as good.*
22: **else**
23: $\mathcal{G} := \mathcal{H}$ *Discard the new graph, because it is worse than the saved graph.*
24: **return** $\mathcal{G}$

25: **function** $S(\mathcal{K}, p, s, d)$ ($\mathcal{K}$ contains the thickness 2 graph K_0, the two triangulated graphs K_1, K_2, the clique list $Q_\mathcal{K}$, among other data, as shown in line 12.)
26: **for** $j = 0, ..., d$ **do**
27: Randomly select $\ell = 1$ or $\ell = 2$.
28: Select a random edge e from edge list $E_\ell \in \mathcal{K}$.
29: Find the corresponding flip $\tilde{e}$ in graph $K_\ell \in \mathcal{K}$ with Algorithm 2.
30: **if** $\tilde{e} \in K_\ell$ **then**
31: Return to line 27. *Flip is non-simple, thus we try again to find a simple flip.*
32: Define Q_- as the list of all cliques with size p in $Q_\mathcal{K}$ that contain edge $\tilde{e}$.
33: Define Q_+ as the list of all new cliques in K_0^c if $\tilde{e}$ is replaced with e in the complement.
34: Define $x = |Q_+| - |Q_-|$
35: **if** $\tilde{e} \in K_0$ **then** *The edge $\tilde{e}$ is already in the other triangulated graph.*
36: $x := x + 1$ *This flip is undesirable, thus we decrease the odds of it being accepted.*
37: **if** $x \leqslant 0$ **then**
38: Go to line 19. *flip is accepted*
39: **else if** the flip is accepted with a probability of $\exp\left(\frac{-x}{1-j/d}\right)$ **then**
40: Go to line 19. *flip is accepted*
41: **else**
42: Go to line 22. *flip is rejected*
43: Flip edge e into $\tilde{e}$ in all objects of $\mathcal{K}$.
44: **if** $|Q_\mathcal{K}| \leqslant s$ **then**
45: **return** $|Q_\mathcal{K}|$ *A new best graph has been found.*
46: $j := j + 1$
47: **return** $|Q_\mathcal{K}|$

Vertices n	Clique size q	Flips/Iteration b	Time	Cliques
17	3	500,000	4 seconds	0
25	4	4,000,000	15 minutes	0
19	3	30,000,000	10 hours	5
28	4	4,000,000	10 hours	30

Figure 16: Different tests ran with the Adapted Sulanke's Algorithm.

2.6 Results of the algorithm

Unfortunately the time allotted for this research did not leave much room for generating and reviewing data with the adapted version of Sulanke's Algorithm. We do have some interesting results to show, but we cannot draw significant conclusions without a proper analysis.

Most importantly, we want our algorithm to be at least as effective as the original version of Sulanke's Algorithm as seen in *Thickness-Two Graphs Part Two* by Gethner and Sulanke [6]. Three types of differently sized graphs with chromatic number 9 are shown in their paper. These correspond to thickness 2 graphs with 17, 25 and 33 vertices that do not have any K_3, K_4 and K_5 in their complement respectively.

We successfully found graphs on 17 vertices with without any K_3 in their complement and on 25 vertices without any K_4 in their complement, by using the adapted version of Sulanke's Algorithm. How fast we could generate these graphs is shown in the first two rows of the table in Figure 16. We can eliminate all triangles in the complement of a thickness 2 graph on 17 vertices within seconds. Eliminating all occurrences of the K_4 on a graph on 25 vertices takes about 15 minutes with our algorithm. Due to timing restrictions, we were unable to program a version of the Adapted Algorithm that minimizes the number of K_5s in the complement. Therefore a graph with 33 vertices and without any K_5 in its complement is missing from our results.

The choice of parameter b is important for getting good results from the algorithm. b corresponds to the number of flips done in each iteration of the algorithm. We have chosen the values of b for our different experiments with some trial and error. We made a rough approximation for which value of b the algorithm eliminates cliques the fastest. When the value of b is too high, the algorithm becomes slow, as it wastes a lot of time when a sequence of flips is unfavourable. A value of b that is too small, restricts the number of flips to much to create a graph that is sufficiently different. This leads to new graphs being very similar to the input, but we require notably different graphs to progress the algorithm. Thus, when the value of b is too low, the algorithm is not able to eliminate all cliques.

In the same table we also show attempts at finding graphs with a chromatic number of 10. These are mostly shown as a point of reference for others who want to implement Sulanke's Algorithm. According to the Complement Theorem, 19 vertices is the best choice when eliminating instances of the K_3 and 28 vertices is the best choice when eliminating instances of the K_4. Our attempts were unsuccessful as in our best results we were left with 5 K_3s and 30 K_4s respectively. The table does not show, that we found the first graph on 19 vertices with 5 K_3s in its complement within 1 minute. Thus, running the algorithm for almost 10 hours did not yield any improvement beyond the 1 minute mark.

Similarly, it took about 2 hours to find a graph on 28 vertices with only 30 instances of the K_4 in the complement. In the other 8 hours the algorithm was unable to improve on this graph. The time required to remove an additional clique seems to be worse than exponential. Even if we manage to reduce the number of cliques below 5 and 30 respectively, it seems unlikely that we can reduce these numbers all the way to 0 with the current algorithm, due to the removal of each subsequent clique taking tremendously longer than the previous clique.

Since minimizing for cliques with a size of 3 and 4 does not look favourable, we could look at the

performance of the algorithm with greater clique sizes. However, the algorithm seems to perform worse, the higher the clique size is. We do not know whether this is because the Complement Theorem gives a worse bound the bigger the clique size is, or whether the altered Sulanke's Algorithm becomes too slow to find good graphs when a bigger clique size is chosen. Not only does the algorithm become exponentially slower, when we search for cliques of a larger size, we also have to increase the number of vertices to satisfy the Complement Theorem. More vertices slows down the algorithm further and the graph space we have to traverse becomes a lot bigger too. Due to the graph space being bigger, we expect that the amount of flips per iteration b needs to be increased to prevent the algorithm getting stuck. With three different factors slowing down the algorithm, we doubt it is feasible to generate useful graphs with a large clique size.

According to Theorem 2.6, we can make any triangulated graph by doing flips, thus our adapted algorithm has a non-zero chance of generating any edge-maximal thickness 2 graph. If there exists a thickness 2 graph on 19 vertices, without any K_3 in its complement, we expect the algorithm to eventually find this graph. However, extrapolating our results gives the impression that our algorithm will never find a qualifying graph on 19 vertices. This gives the impression that there does no exist a thickness 2 graph on 19 vertices, without any K_3 in its complement.

Since increasing the clique size and the number of vertices gives worse results, we do not expect the algorithm to find a chromatic number 10 graph by eliminating cliques of size $a \geqslant 4$. These observations together lead us to a conjecture, which might explain why Sulanke's Algorithm is seemingly unable to break the chromatic number 10 barrier. Sulanke's Algorithm should be able to find chromatic number 10 graphs if they exist within the bounds of the Complement Theorem. Since our results get nowhere close to discovering a chromatic number 10 graph, we predict that these graphs do not exist within the bounds of the Complement Theorem. Seemingly, the Complement Theorem's bound is not tight enough to actually contain any chromatic number 10 graphs in the thickness 2 case. Do note, we have not done enough tests to confidently back up this conjecture.

Conjecture 2.7 *Let $G = (V, E)$ be a graph with thickness 2 and a chromatic number of 10. Denote n as the number of vertices in E and a as the biggest integer for which the edge complement of G contains no K_a. We conjecture that the following inequality holds:*

$$\left\lceil \frac{n}{a-1} \right\rceil < 10.$$

If our conjecture is true, it would imply that there does not exist a thickness 2 graph, that the Complement Theorem 2.1 would identify as a graph with a chromatic number of 10 or higher. As a result, our current version of Sulanke's Algorithm would not be able to find any chromatic number 10 graph. Regardless of the validity of the conjecture, Sulanke's Algorithm still is an efficient algorithm for finding edge-maximal graphs with a specific property. The Complement Theorem's bound can be replaced with a different bound in Sulanke's Algorithm, to try to solve the problems caused by the Complement Theorem. The discovery of a different bound for indicating chromatic number 10 graphs, one that is tighter than the Complement's Theorem bound, might lead to a new breakthrough for the Earth-Moon Problem.

3 Research on the upper bound

In Subsection 1.3 we quoted that the upper bound for the Earth-Moon problem is 12, because we know, that there cannot exist any thickness 2 graph with a chromatic number of 13. Finding tighter upper bounds is tricky, because that requires a solid proof, rather than a simple counterexample that would be enough to improve the lower bound. We will investigate two different ideas, that might lead to a new upper bound for the Earth-Moon problem. We start out by examining new developments on the four colour theorem, followed by the use of mappings between thickness 2 graphs and 2-pire graphs in Subsection 3.2.

3.1 Developments on the four colour theorem

In 2018 J. A. Tilley wrote a paper on a possible alternate proof for the 4 colour theorem 1.3 [3]. Since the 4 colour theorem is closely related to the Earth-Moon problem, Tilley's research could be helpful for the Earth-Moon problem. We investigate, whether that is the case, despite the paper still being peer-reviewed. This entire subsection is a short summary of the preprint *Kempe-Locking Configurations by J. A. Tilley* [3] and we discuss how it relates to the Earth-Moon problem. Our summary is quite brief, so I recommend reading Tilley's preprint instead for readers, who are interested in the specifics.

All known proofs of the 4 colour theorem are extremely tedious to verify without the help of computers. These proofs do not offer much insight on the reason why the 4 colour theorem is true. Tilley has formulated a conjecture that would provide a more insightful proof of the 4 colour theorem, provided that the conjecture is true of course. This is an interesting development that has reignited the interest of mathematicians on this already solved problem.

Before we start explaining Tilley's conjecture, we assume that the 4 colour theorem is merely a conjecture. Otherwise a lot of the upcoming statements are trivialised by the fact, that we already know that no planar graphs exist that have a chromatic number of 5.

To reduce the number of planar graphs, that we have to consider in order to be a counterexample to the 4 colour conjecture, we restrict ourselves to minimal counterexamples: planar graphs with the smallest number of vertices while still having a chromatic number of 5. There are two properties all minimum counterexamples share and these properties are seemingly mutually exclusive. If they are indeed mutually exclusive, this would provide an alternate proof of the 4 colour theorem.

The first property is similar to k-connectedness, which we briefly covered in Theorem 2.3. We repeat the definition as a reminder: a graph is k-connected if the graph has more than k vertices and at least k vertices need to be removed before the graph becomes disconnected. An average degree of at least 6 is required for a graph to be 6-connected, but from Euler's Formula it follows that the average degree of a planar graph is strictly smaller than 6. Therefore the highest connectivity a planar graph can have is 5. There does exist a class of planar graphs that consists of almost 6-connected graphs.

Definition 3.1 (Internally 6-connected) A planar graph with a planar representation is said to be internally 6-connected if its minimum degree is 5 and if it has no cycle of length 5 or less for which there are two or more vertices both inside and outside the cycle.

It turns out that a minimal counterexample must have a very high connectivity, hence our interest in internally 6-connected graphs. This leads to the first property a minimum counterexample must have.

Theorem 3.2 *A minimum counterexample to the 4-colour theorem is internally 6-connected.*

Proof. This theorem has originally been proven by Birkhoff in 1913 [15]. A modern version of this proof can be found in Steinberger's work from 2009 [16]. □

The second property is all about Kempe chains. These are helpful and well-known tools for graph colouring problems. These chains are named after the mathematician Kempe, who worked on the 4-colour problem and got very close to a proof of said problem [17]. Let a coloured graph G, a vertex v with colour a and a second colour b be given. A Kempe chain is the maximal connected subgraph of vertices that have the colour a or b. There might be multiple disconnected subgraphs on vertices with the colour a and b, thus we also require that vertex v is contained in the subgraph to get a unique subgraph. We call this an (a, b)-Kempe chain. When all vertices coloured a or b form a single connected component, all vertices with these two colours are in this Kempe chain. Else there are multiple disconnected (a, b)-Kempe chains in the graph.

Kempe chains offer an easy method of finding different colourings of a graph, given that we already have a first colouring of that graph. We can simply take a Kempe chain and swap all colours along the chain. Since all vertices adjacent to the Kempe chain do not share a colour with vertices in the chain, the colouring is still proper after the swap.

We also introduce the concept of a "near triangulated graph", in order to formulate the second property. Given a triangulated graph G and an edge (v, w), the *near triangulated graph* $G_{(v,w)}$ is the graph G with edge (v, w) removed. The resulting graph has triangular faces for all but one face. The one non-triangular face is quadrangular, because it is a merging of the two triangular faces adjacent to the removed edge (v, w).

Definition 3.3 (Kempe-locked) Given a triangulated graph G and an edge (v, w), we say that G is Kempe-locked with respect to (v, w) if the following holds: in every 4-colouring of the near triangulated graph $G_{(v,w)}$, in which v and w have the same colour, all Kempe chains that contain one of these two vertices also contain the other vertex.

A graph G being Kempe-locked with respect to the edge (v, w) basically means that in the near triangulated graph $G_{(v,w)}$ it is difficult to give vertex v and w a different colour. If v and w are the same colour in a given colouring, we can use the trick with Kempe chains as mentioned before to try to give them two different colours. If there is a Kempe chain that contains vertex v, but not w, we can swap the two colours in that Kempe chain. This would result in a proper colouring where v has gotten a different colour than w. However, such a Kempe chain does not exist in this case, because G is Kempe-locked with respect to the edge (v, w). If we use a Kempe chain to swap the colour of v, that Kempe chain also contains w which gets swapped to the same colour simultaneously.

Theorem 3.4 *A minimum counterexample to the 4-colour theorem is Kempe-locked with respect to each edge in the graph.*

Proof. We use contradiction to prove the theorem. Let $G = (V, E)$ be a minimum counterexample to the 4-colour theorem and $(v, w) \in E$ an edge for which G is not Kempe-locked with respect to (v, w). Thus there must exist a 4-colouring of the near triangulated graph $G_{(v,w)}$ in which the vertices v and w have the same colour, such that there is a Kempe chain that contains v, but not w. We can swap the colours along that Kempe chain to get a proper colouring of $G_{(v,w)}$ in which vertex v and w have different colours. If we add back the edge (v, w), the colouring is still proper, because the two edges have different colours. Now we have coloured our counterexample to the 4-colour theorem with 4 colours. This is a contradiction and we conclude that a counterexample to the 4-colour theorem must be Kempe-locked with respect to every single edge.

We have left some details out of this proof to keep this summary brief. A complete proof can be found in the preprint *Kempe-Locking Configurations.* [3] □

When we combine Theorems 3.2 and 3.4, we know that a minimum counterexample to the 4-colour theorem is both internally 6-connected and Kempe-locked with respect to each edge in the graph. Individually these are two strong properties, that not many triangulated graphs satisfy and it is seemingly impossible for a triangulated graph to satisfy both properties. During his research Tilley has not been able to find a triangulated graph that is internally 6-connected, which is also Kampe-locked to even a single edge! This led to the main conjecture of his work.

Conjecture 3.5 *No planar triangulation that is Kempe-locked with respect to an edge, is internally 6-connected.*

At this point Tilley's research becomes even more interesting. In a systematic search among triangulated graphs, he found an infinite amount of Kempe-locked graphs. Furthermore, all triangulated graphs that are known to be Kempe-locked, all share a very specific property. Every single one of them contains a Birkhoff diamond, which is shown in Figure 17. A triangulated graph that contains a Birkhoff diamond is clearly not internally 6-connected, because the six vertices in the hexagonal shape can be removed to split the graph into two disjoint sets that contain two or more vertices. Thus graphs that contain a Birkhoff diamond cannot be a minimum counterexample.

The Birkhoff diamond plays a crucial role in the original proof of the 4-colour theorem, thus it is interesting to see it show up again in this different approach to the problem. The diamond is

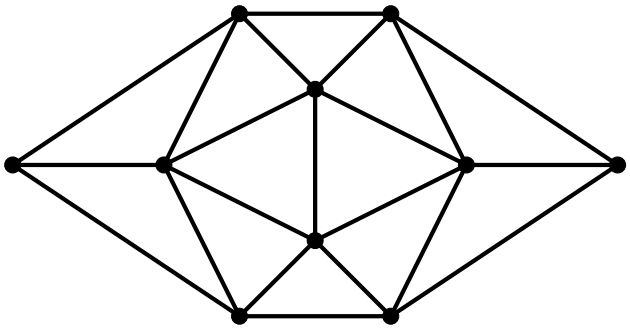


Figure 17: The Birkhoff diamond.

named after the same mathematician who proved the internally 6-connected property of a minimal counterexample, which we talked about earlier.

All in all, we have two properties that a minimal counterexample for the 4-colour theorem must satisfy and they are seemingly incompatible. How does all this translate into the Earth-Moon problem? Theorem 3.4 tells us that a minimal thickness 1 graphs with a chromatic number of 5 must be Kempe-locked to each of its edges, but the proof of this theorem does not rely on the graph having a thickness of 1. If we extend the definition of Kempe-locked 3.3, such that it is also applicable to thickness 2 graph, the proof of Theorem 3.4 becomes applicable to thickness 2 graphs too. It follows that a hypothetical minimal thickness 2 graph with a chromatic number of 12 is Kempe-locked for each edge.

We are not sure how practical the Kempe-locked requirement is in the thickness 2 case. This property is already difficult to prove for thickness 1 graphs. Imagine how difficult it becomes to do the same for thickness 2 graphs. Furthermore, all thickness 1 graphs that are Kempe-locked with respect to a single edge seem to contain the fundamental Birkhoff diamond, but we do not know whether a subgraph with similar properties exists in the thickness 2 case.

The minimal connectivity a minimal counterexample for the Earth-Moon problem must have is more difficult to derive. Since we are attempting to colour thickness 2 graphs with 12 colours, we expect that the minimal counterexample for this case would have a minimal connectivity of somewhere around 12. Unfortunately Birkhoff's proof that a minimal counterexample of the thickness 1 case must be internally 6-connected [15] does not apply to thickness 2 graphs. We were unsuccessful at finding any non-trivial connectivity requirements a minimal thickness 2 graph with a chromatic number of 12 must have.

This leaves us with the question: "What is the highest k-connectivity a minimal thickness 2 graph with a chromatic number of 12 must have?" Admittedly we barely know anything about the answer to this question, because this is beyond our area of expertise. We do not even know if the answer to this question has any relevance in advancing the Earth-Moon problem.

Lastly, the two properties that a minimum counterexample to the 4-colour theorem must have are not too interesting on their own. It is the combination of the two properties and the unavoidability of a Birkhoff diamond that makes Tilley's conjecture so interesting. Currently we have no reason to believe we can construct a similarly powerful conjecture for the Earth-Moon problem. Furthermore, the Kempe-locked property is hard to compute and the connectivity property is seemingly hard to prove in the thickness 2 case.

There is also another problem with extending Tilley's conjecture to the Earth-Moon problem. Tilley states that his conjecture is *"likely to be true but beyond difficult to prove"* [3], which does not bode well for extending it to the even more complex thickness 2 case. There is one part of Tilley's conjecture that might not be computationally difficult if it could be extended to the Earth-Moon problem. It would be interesting, if there exists a decently sized subgraph that all minimum thickness 2 graphs with a chromatic number of 12 must contain, similarly to the Birkhoff diamond. There are no leads to the existence of such a graph, thus this is mere speculation.

3.2 Mappings between graphs

In this section we introduce a different approach to search for upper bounds of the Earth-Moon problem. The idea is to use mappings between thickness 2 graphs and 2-pire graphs. Since more is known about 2-pire graphs, we might be able to extend some of that knowledge to thickness 2 graphs with the help of these mappings. Corollary 1.10 is a basic example of this concept, which proves that thickness 2 graphs have a chromatic number of at most 12. For this proof a mapping from the set of 2-pire graphs to the set of thickness 2 graphs is used. We can also use different mappings to find more relations between these two sets of graphs. We came up with two theorems that show how other mappings could be useful for the Earth-Moon problem. Note, we have not done much research on these theorems. They are merely meant as food for thought.

Theorem 3.6 *Denote $\mathcal{G}_{em}$ as the set of all graphs with thickness 2 and $\mathcal{G}_{2p}$ as the set all 2-pire graphs. If there exists a mapping $f : \mathcal{G}_{em} \to \mathcal{G}_{2p}$ such that $\chi\big(f(G)\big) > \chi(G)$ is true for all $G \in \mathcal{G}_{em}$, there does not exist a thickness 2 graph with a chromatic number of 12.*

Proof. Let f be a mapping as given in the theorem and assume that there exists a thickness 2 graph $G \in \mathcal{G}_{\text{em}}$ with $\chi(G) = 12$. Then there also exists the 2-pire graph $f(G) \in \mathcal{G}_{\text{2p}}$, for which we have this lower bound on its chromatic number: $\chi\big(f(G)\big) > 12 = \chi(G)$. A 2-pire graph with a chromatic number greater than 12 does not exist as it contradicts with Theorem 1.9. We conclude that if a mapping f exists, there does not exist a thickness 2 graph with a chromatic number of 12. □

A mapping f as seen in Theorem 3.6 does not seem realistic, because it is very difficult to increase the chromatic number of a (thickness 2) graph without restricting the mapping to a very specific subset of graphs. The theorem does show that some mappings could lead to interesting information about thickness 2 graphs. The following theorem we derived is a more complex example of the mapping concept.

Theorem 3.7 *Let $G = (V, E)$ be a 2-pire graph with $\chi(G) = 12$ that contains an empire $v \in V$ consisting of only a single vertex. Furthermore, the 2-pire graph G', which is the graph G without the 1-pire v, must satisfy $\chi(G') = 11$. If there does not exist a 2-pire graph G with these properties, no thickness 2 graphs with a chromatic number of 12 exist either.*

Proof. By inverting the theorem we get the following equivalent statement: If there exists a thickness 2 graph H with $\chi(H) = 12$, then there exists a 2-pire graph G as described in the theorem. Assume such a graph H exists. This implies the existence of a minimum thickness 2 graph with a chromatic number of 12. Let $H' = (V', E')$ be a thickness 2 graph that has the smallest amount of vertices that satisfies $\chi(H') = 12$.

We use the thickness 2 property to split H' into two planar graphs. Let $H'_1 = (V', E'_1), H'_2 = (V', E'_2) \subseteq H'$ be two disjoint planar subgraphs of H' such that E'_1 and E'_2 are a partition of E. We can now construct the graph G merging H'_1 and H'_2 into one 2-pire graph. Two vertices that are equal are replaced by a 2-pire in G. We still need to create an empire which has 1 rather than 2 vertices. To do so, we pick any 2-pire v. The 2-pire v has two vertices, one in each disjoint subgraph of G. We can merge the two vertices of v by placing the two disjointed subgraphs on top of each other, which preserves the adjacency matrix of the M-pires. An example of this procedure with a smaller graph is shown in Figure 18. The size and the chromatic number of the thickness 2 graph does not matter, because we can always merge the two planar subgraphs into a 2-pire graph this way.

We have $\chi(G) = 12$, because it G has the same adjacency matrix as H'. Since H' is a minimum graph with a chromatic number of 12, removing any vertex in H' or empire in G decreases the chromatic number. Notably, removing the empire v from G reduces the chromatic number to 11.

Under the assumption that there exists a thickness 2 graph H with $\chi(H) = 12$, we have constructed a 2-pire graph G with $\chi(G) = 12$. We can create the 2-pire graph G' by removing empire v from G, such that $\chi(G') = 11$ holds, where v is an empire that consists of a single vertex. This is exactly a

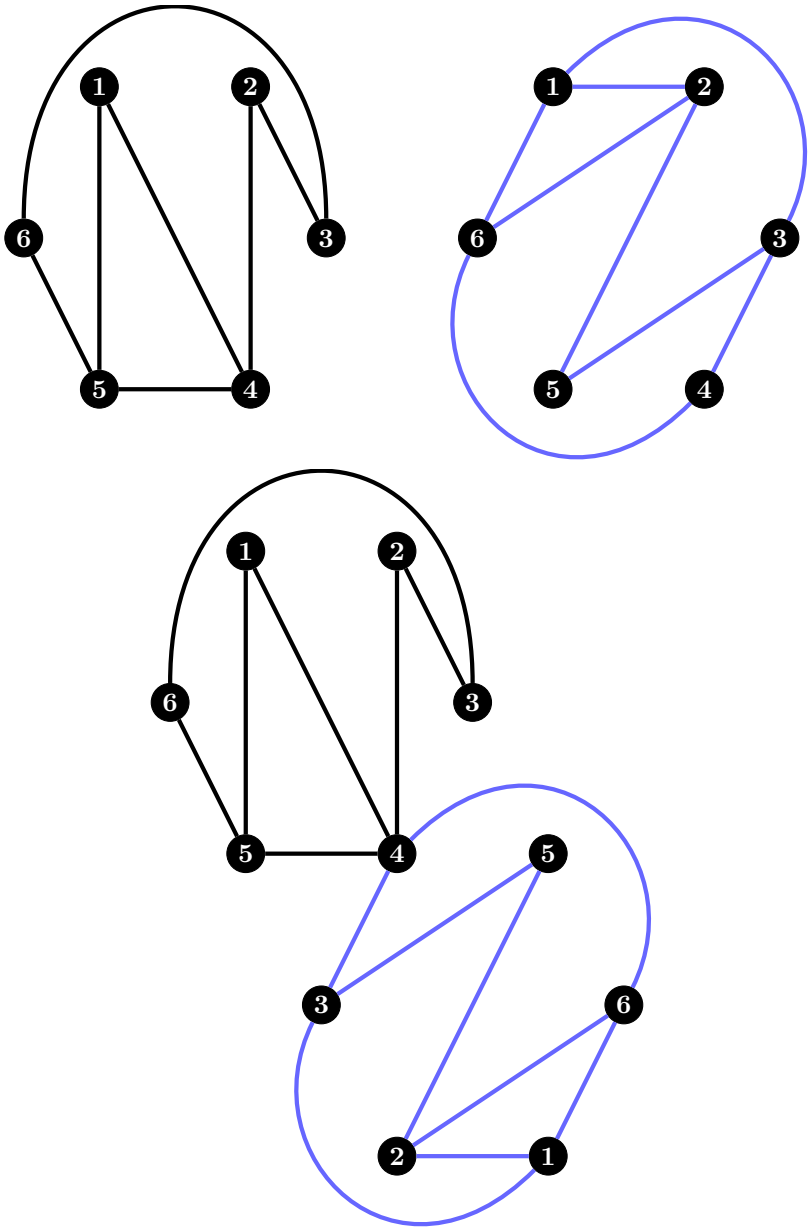


Figure 18: The thickness 2 graph K_6 drawn in two different ways. On top we have the K_6 split into two planar graphs and below the K_6 as a 2-pire. The 2-pire graph is created by placing the two planar graphs on top of each other.

2-pire graph G we wanted to prove the existence of, as mentioned at the beginning of the proof. With this we have proven the statement that is equivalent to the theorem, thus the theorem is true too. □

The mapping in the proof of Theorem 3.7 shows that 2-pire graphs can be used in different ways to upper bound the Earth-Moon problem. Checking the existence of such a graph G might give more insight on the problem. Currently, we have no clue whether such a graph G exists or not.

4 Future of the Earth-Moon problem

Our research did not lead to an improvement to the bounds of the Earth-Moon problem, but did gather new knowledge on this topic. In this section, we summarize the parts of this thesis that, according to our opinion, have the most potential for future research on the Earth-Moon problem.

4.1 Potential of Sulanke's Algorithm

The results of our altered version of Sulanke's Algorithm shown in Subsection 2.6 fall short to prove anything meaningful due to the small amount of analysed data. We have not done extensive research to optimize the parameters of our algorithm and we do not have the data to back up Conjecture 2.7 in which we predict that Sulanke's Algorithm cannot find a thickness 2 graph with a chromatic number of 10.

By optimizing the different parameters, Algorithm 5 can be improved. Most importantly we do not know the best choice for the parameters a and b. a is the maximum number of times we run the original Sulanke's Algorithm and b is the maximum number of iterations we run each of these algorithms. Increasing a or b improves the output of the algorithm, but the runtime of the algorithm is roughly linear in $a \cdot b$. Therefore we cannot simply increase a and b indefinitely. A good balance between the two parameters has to be found, which depends on a lot of different factors.

Another unoptimized part of the algorithm is the probability to accept a flip. In Subsection 2.5 we explain in detail, why we chose this probability and why it is unlikely to be optimal. Specifically, the probability function that accepts flips, can probably be improved. Also the weight of additional cliques compared to adding a parallel edge leaves room for optimization. Currently, they have the same weight. This is a choice made out of simplicity rather than efficiency. By running a lot of tests, it should be possible to find a good approximation for which probability to accept flips works best.

I believe that the altered version of Sulanke's Algorithm can be speeded up quite a bit when it is optimized. However, the improvement might be polynomial at best and this will not have a big impact on the grand scheme of things. According to our results, we still are far off a potential graph with a chromatic number of 10 and we might never achieve it with our algorithm. Nonetheless, it is still interesting so see how close we can get to such a graph after optimizing the parameters of our algorithm.

Our results show, that removing cliques from thickness 2 graphs gets slower, the fewer cliques are left. This time required for removing each additional clique seems a lot slower than exponential. The time required to remove an additional clique grows so fast, that it does not seem realistic to ever generate a chromatic number 10 graph with the algorithm. We hope, that someone can give more insight whether this is actually true by running more experiments. The best possible outcome would be that our extrapolation of our results is a poor representation of reality. If our prediction is correct on the other hand and Conjecture 2.7 is true, it would mean that we have reached the limit of Sulanke's Algorithm.

Regardless of the outcome, it would be interesting to see someone beat our results shown in Figure 16. A thickness 2 graph on 19 vertices and less than 5 K_3s in its complement would be an improvement on the graph we found. Similarly, a thickness 2 graph on 28 vertices with less than 30 K_3s in its complement would be an improvement too. We are also missing data on graphs where we minimize over the number of cliques with a size of 5 or greater. Maybe there is a clique size $q \geqslant 5$ for which Sulanke's Algorithm performs better than the ones we investigated.

4.2 Size of a chromatic number 12 graph

One of the reasons why we doubt that we can make progress on the Earth-Moon problem with Sulanke's Algorithm, is because the algorithm seemingly performs worse the bigger the graph gets. This could be problematic, because big graphs might be key to find graphs with a high chromatic number.

We used the Complement Theorem 2.1 to find the most promising numbers of vertices to use with Sulanke's Algorithm. As an example, let G be a graph with $n = 23$ vertices and no cliques of size $a = 3$ in the complement. This graph would have a chromatic number of 12. The following theorem shows that G does not exist, because graphs with a chromatic number of 12 have at least 24 vertices.

Theorem 4.1 *Let $\mathcal{G}$ be the set of graphs that have both the properties $\theta(G) = 2$ and $\chi(G) = 12$. All graphs $G = (V, E) \in \mathcal{G}$ satisfy $|V| = n \geqslant 24$.*

Proof. If $\mathcal{G} = \varnothing$, the theorem is trivial and true by default. Otherwise we consider a graph $H \in \mathcal{G}$ that has the least number of vertices in $\mathcal{G}$. We use a similar proof to that of Theorem 1.9. Assume, that H has a vertex v with degree $\leqslant 10$ and create a new graph H' by removing vertex v. Since H was the smallest graph with a chromatic number of 12, we can colour H' with 11 colours. Next, we apply the same colouring with 11 colours to H, which leaves v without a colour. v is adjacent to at most 10 vertices in H, thus there is at least 1 of the 11 colours missing among the neighbours of v. We can colour v with this colour to get a proper colouring of H with 11 colours. This is a contradiction with H having a chromatic colour of 12. Therefore all vertices of H have a degree of at least 11.

Since all vertices in H have a degree of at least 11, the average degree D is at least 11 too. Theorem 1.4 gives us the upper bound $m \leqslant 3n - 6$ on the number of edges in a planar graph. A thickness 2 graph can have at most double that amount, with $2(3n - 6) = 6n - 12$ edges. We can use this to bound the average degree.

$$11 \leqslant D = \frac{2m}{n} \leqslant \frac{2 \cdot 2(3n-6)}{n} = \frac{12n-24}{n} = 12 - \frac{24}{n}$$

For $n = 24$ equality holds. Since $12 - \frac{24}{n}$ is strictly increasing in n, the equality does not hold for all $n < 24$. We conclude that a graph with the least number of vertices in $\mathcal{G}$ has at least 24 vertices, which must also be true for all other graphs in $\mathcal{G}$. □

The proof of Theorem 4.1 shows that a graph on $n = 23$ vertices and no K_3s in its complement has too low of an average degree to have a chromatic number of 12. On top of that, it is unrealistic that a thickness 2 graph G with 24 vertices and a chromatic number of 12 exists. It would require that in all colourings of G each vertex is adjacent to exactly 11 colours, without any duplicates colours among neighbouring vertices. This is a bit of an extreme example, but Theorem 4.1 does show us that a high average degree is important for getting high chromatic numbers in thickness 2 graphs. A high average degree in a thickness 2 graph requires the graph to have a lot of vertices. The Theorem also shows us that an average degree of 12 is not achievable in a thickness 2 graph, but we can get asymptotically close to 12 by increasing the number of vertices.

We think that the best way to approach the lower bound of the Earth-Moon problem is by constructing an algorithm that can generate very large thickness 2 graphs with a high chromatic number. Theorem 4.1 seems to point into that direction and other mathematicians also share this vision. For example, Gethner has stated the following about thickness 2 graph with a chromatic number of 10: *"Our intuition is that such graphs exist, but the smallest examples may have 100s of vertices and hence have been, up until now, computationally infeasible to find"* [5].

Currently, there is no such algorithm that can generate interesting large graphs with a high chromatic number. Van der Beek does discuss the idea of an algorithm that might be able to generate large graphs [1]. In Sulanke's Algorithm we start out with a thickness 2 graph and alter it to make the chromatic number as high as possible. We can also do the opposite where we start out with a graph that has a chromatic number of 10 and try to split it into two planar graphs to prove it has a thickness of 2. The biggest advantage of such an algorithm is, that we can check in linear time whether a subgraph is planar. This was proven by Hopcroft and Tarjan [18]. Van der Beek goes into more detail of what such an algorithm might look like [1], but we currently do not know of any algorithms that use this strategy. This type of algorithm brings its own complications. Determining the exact thickness is NP [19]. Computing the exact thickness of large graphs is infeasible with algorithms, thus we have to rely on heuristics to prove a graph has a thickness of 2. Furthermore, we require graphs that have a chromatic number of 10 and have a small thickness. Finding and selecting these graphs can prove challenging too.

A Bibliography

[1] B.W.B. van der Beek. *Graph-Theoretical Methods for Ringel's Earth-Moon Problem*. 2022.

[2] K. Appel and W. Haken. "Every planar map is four colorable". In: *Bulletin of the American Mathematical Society* 82.5 (1976), pp. 711–712.

[3] J. A. Tilley. *Kempe-Locking Configurations (preprint)*. 2018.

[4] G. Ringel. *Farbungsprobleme auf Flachen und Graphen*. Deutscher Verlag der Wissenschaften, 1959.

[5] E. Gethner. "To the Moon and Beyond". In: *Graph Theory, Favorite Conjectures and Open Problems* 2 (2018), pp. 115–133.

[6] T. Sulanke E. Gethner. "Thickness-Two Graphs Part Two: More New Nine-Critical Graphs, Independence Ratio, Cloned Planar Graphs, and Singly and Doubly Outerplanar Graphs". In: *Graphs and Combinatorics* 25 (2009), pp. 197–217.

[7] P.J. Haywood. "Map-Colour Theorem". In: *Quarterly Journal of Mathematics, Oxford* 24 (1890), pp. 332–338.

[8] B. Jackson and G. Ringel. "Solution of Heawood's empire problem in the plane." In: *Journal für die reine und angewandte Mathematik* 1984.347 (1984), pp. 146–153.

[9] T. Husfeldt A. Björklund and M. Koivisto. "Set Partitioning via Inclusion-Exclusion". In: *SIAM Journal on Computing* 39.2 (2009), pp. 546–563.

[10] L. Lovász. "On the Shannon capacity of a graph". In: *IEEE Transaction on Information Theory* IT-25.1 (1979), pp. 1–7.

[11] L.E. van Valkengoed. *The Earth-Moon Problem: A Better Approach*. 2024.

[12] S. L. Hakimi and E. F. Schmeichel. "On the connectivity of maximal planar graphs". In: *Journal of Graph Theory* 2.4 (1978), pp. 307–314.

[13] H. Whitney. "Congruent Graphs and the Connectivity of Graphs". In: *American Journal of Mathematics* 54.1 (1932), pp. 150–168.

[14] N. Chiba and T. Nishizeki. "Arboricity and Subgraph Listing Algorithms". In: *SIAM Journal on Computing* 14.1 (1985), pp. 210–223.

[15] G. D. Birkhoff. "The Reducibility of Maps". In: *American Journal of Mathematics* 35 (1913), p. 115.

[16] J. Steinberger. *An unavoidable set of D-reducible configurations*. 2009.

[17] A. B. Kempe. "On the Geographical Problem of the Four Colours". In: *American Journal of Mathematics* 2.3 (1879), pp. 193–200.

[18] J. Hopcroft and R. Tarjan. "Efficient Planarity Testing". In: *J. ACM* 21.4 (1974), pp. 549–568.

[19] Anthony Mansfield. "Determining the thickness of graphs is NP-hard". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 93.1 (1983), pp. 9–23.

B Generated Graphs

```
 0  2  3  4  5  6  7  8 12 14 16 18
 1  3  8  9 10 11 13 15 17
 2  3  4  5  6  7  8  9 10 11 13 14 15 16 18
 3  5  8  9 10 11 14 15 17 18
 4  5  6  7  8 12 13 14 16 18
 5  6  7 12 14 16 18
 6  7 12 14 16 18
 7 12 13 14 16 18
 8  9 10 11 13 15 16 17
 9 10 11 13 14 15 17
10 11 13 15 16 17
11 13 15 16 17
12 14 16 18
13 15 16 17 18
14 15 18
15 17
16 17 18
17
18
```

Adjacency list of the thickness-2 graph G on 19 vertices and 5 triangles in its complement.

```
 0  2  9 11 12 13 14 15 16 17 18 19 24
 1  2  3  4  5  6 19 21 23 26 27
 2  3  4  5  6 14 19 20 21 22 23 24 25 26
 3  4  5  6 19 20 21 23 26 27
 4  5  6 19 21 23 26 27
 5  6 20 21 22 23 26 27
 6 19 21 23 26 27
 7  8 10 12 15 19 20 21 22 24 25
 8 10 12 18 19 20 21 22 24 25
 9 11 13 14 15 16 17 18 26 27
10 12 15 19 20 21 22 24 25 27
11 13 14 15 16 17 18 27
12 16 18 19 20 22 24 25
13 14 15 16 17 18
14 15 16 17 18 19 24
15 16 17 22 25 26 27
16 17 18 19
17 18 27
18 19 24
19 23 24
20 21 22 24 25
21 22 25 26 27
22 24 25 26
23 24 26 27
24 25
25
26 27
27
```

Adjacency list of the thickness-2 graph H on 28 vertices and 30 triangles in its complement.

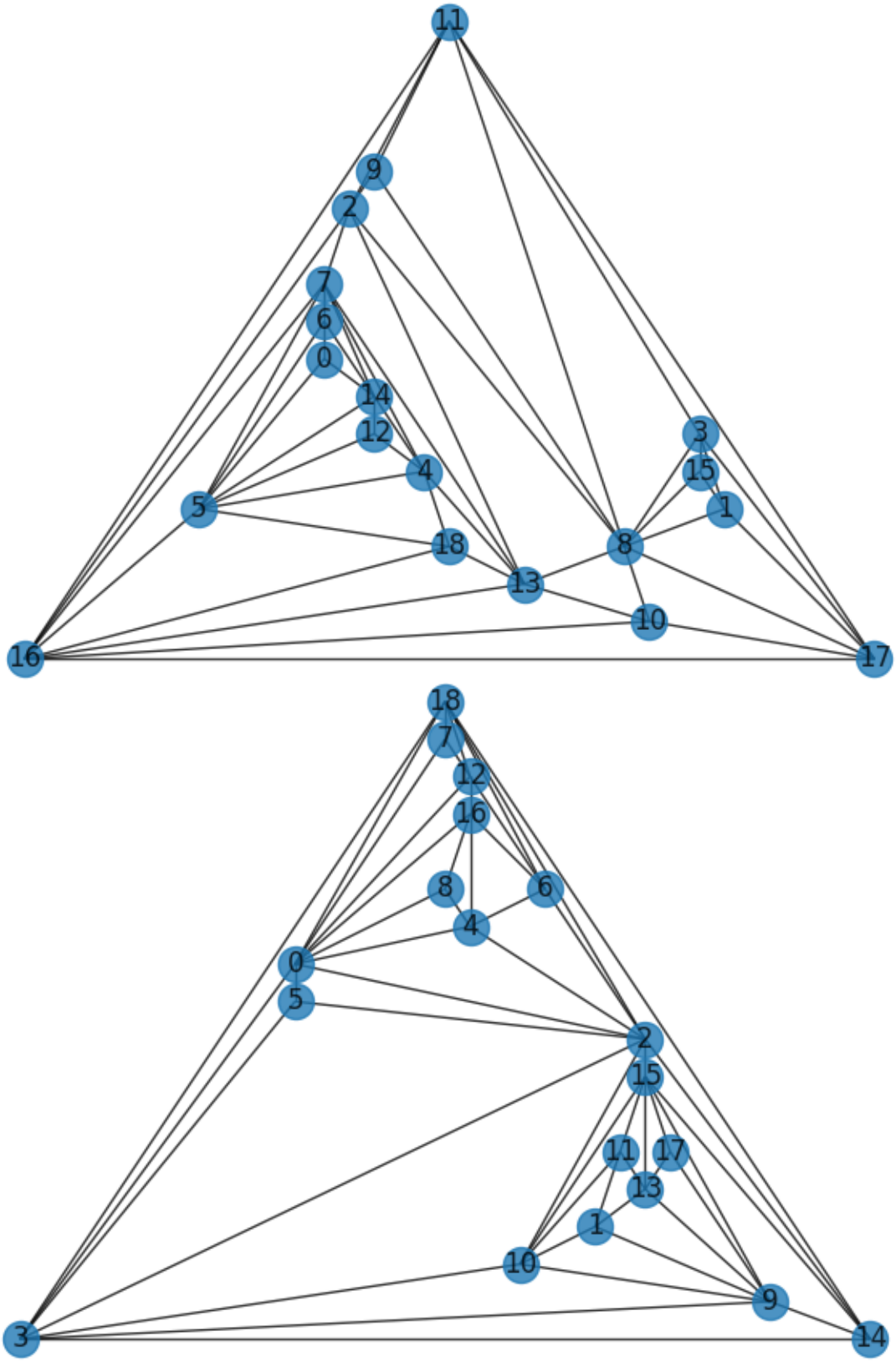


Figure 19: Two planar graphs that form graph G when combined, which shows $\theta(G) = 2$.

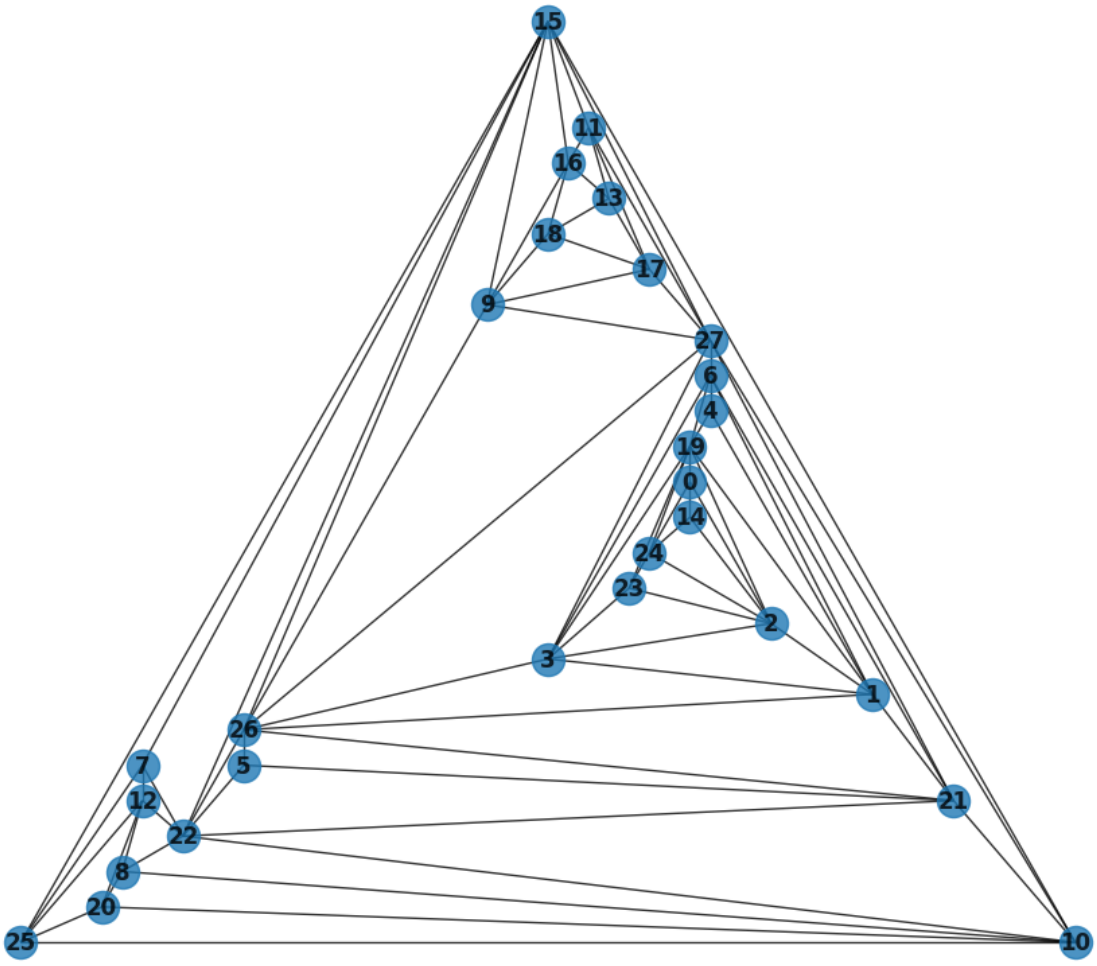


Figure 20: First planar graph that forms graph H when combined with its other half, which shows $\theta(H) = 2$.

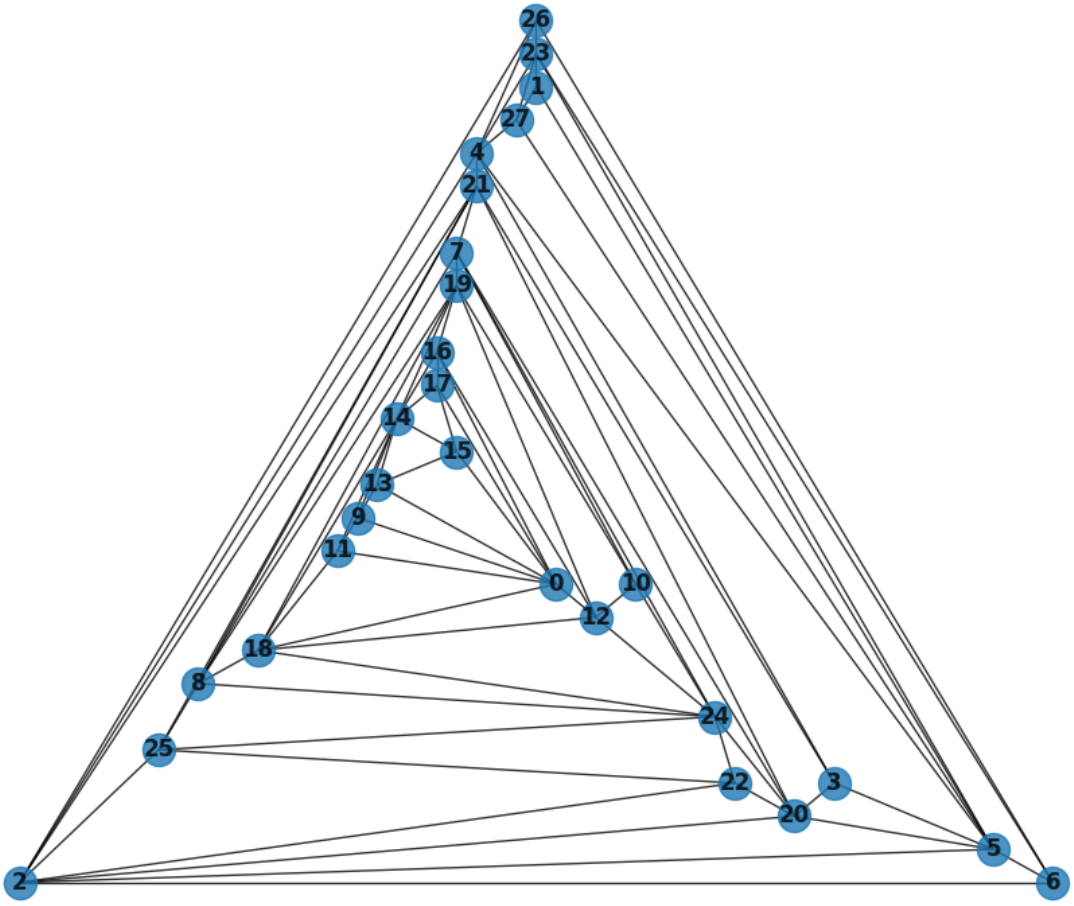


Figure 21: Second planar graph that forms graph H when combined with its other half, which shows $\theta(H) = 2$.