Rose-Hulman Undergraduate Mathematics Journal

Volume 24
Issue 2

Article 5

Utilizing graph thickness heuristics on the Earth-moon Problem

Robert C. Weaver
York College of Pennsylvania, rweaver11@ycp.edu

Utilizing graph thickness heuristics on the Earth-moon Problem

Cover Page Footnote
N/a

Rose-Hulman Undergraduate Mathematics Journal
Volume 24, Issue 2, 2023

Utilizing graph thickness heuristics on the Earth-moon Problem

By *Robert Weaver*

Abstract. This paper utilizes heuristic algorithms for determining graph thickness in order to attempt to find a 10-chromatic thickness-2 graph. Doing so would eliminate 9 colors as a potential solution to the Earth-moon Problem. An empirical analysis of the algorithms made by the author are provided. Additionally, the paper lists various graphs that may or nearly have a thickness of 2, which may be solutions if one can find two planar subgraphs that partition all of the graph's edges.

1 Introduction

One of the most well-known theorems in graph theory is the Four Color Theorem. This theorem proves that for any planar graph, the maximum number of colors needed for a proper vertex coloring is four. Despite this problem being conjectured in 1852, along with four as a proposed solution, it wasn't until 1976 that the first correct proof was published [6, p. 116]. Consequently, it is a fact that any ordinary two-dimensional map can have the represented territories shaded in only four or less colors with no border-sharing territories having the same color.

However, there is a slight flaw with the Four Color Theorem in regards to real-world mapping. Maps containing countries in which a country's territories are split, such as the United States and Alaska or how Pakistan once was, cannot always be colored with four or less colors due to this separation [6, p. 117]. To address this issue, Heawood proposed the M-pire Problem [6, p. 117]. In such a map, each country has at most M territories. Naturally, Heawood sought the maximum chromatic number of such M-pire maps.

Heawood was able to completely solve the M = 2 case and the general case, which was that 6M colors would suffice for any M-pire map [6, p. 117]. However, in 1959, Gerhard Ringel proposed a simplification to this problem known as the Earth-moon Problem [6, p. 118]. Say that each country on Earth has a colony on the moon. How many colors are needed in order to properly color these maps such that no territories

Mathematics Subject Classification. 11A41
Keywords. graph thickness, planar graphs, Earth-moon Problem, chromatic number, graph coloring, heuristic algorithms, SageMath

sharing borders are the same color and each country's territory is the same color on both Earth and the moon?

However, the answer to this simplified problem remains unknown. While it is known that 12 colors will suffice due to Heawood's solution for the M-pire Problem, it is not known if there exists an Earth-moon graph where 12 colors is necessary. Ringel had proposed that 8 colors would suffice in his Earth-Mars map problem in 1959 [5, p. 21]. Yet, Thom Sulanke would provide an example needing 9 colors in 1974, pictured below [5, p. 21]. Therefore, it is not known for certain whether the maximum number of colors needed is 9, 10, 11, or 12.

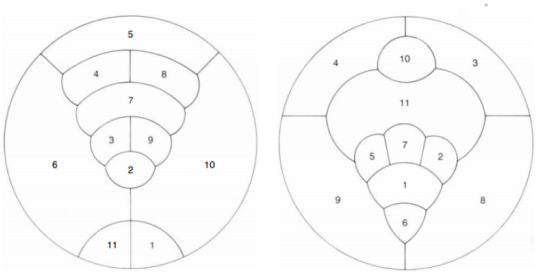

Figure 1: The Sulanke Graph

Often, Earth-moon graphs are described as having a thickness of 2. In general, we define the thickness of a graph G as the minimum number of planar subgraphs necessary to partition all of the edges in G, denoted $\Theta(G)$. Due to the Earth-moon Problem dealing with the union of two planar graphs on the same vertex set, the thickness of any Earth-moon graph will always be 1 or 2 [12, p. 88]. This is because the set of edges for any Earth-moon graph can always be separated into two subsets which form planar graphs on the vertex set, i.e. the Earth graph and the moon graph. In short, thickness-2 graphs are always Earth-moon graphs.

This paper discusses two attempts to find a 10-chromatic thickness-2 graph using heuristic algorithms made by the author. Heuristic algorithms are algorithms which generally sacrifice completeness or accuracy for speed. This is necessary because determining the thickness of a graph is an NP-Complete problem in the general case [10, p. 77]. As such, it takes a great deal of time for a proper algorithm to determine the thickness of a graph. These attempts aim to solve two of the open problems given by Sulanke and Gethner, which would prove the existence of thickness-2 graphs [7, p. 215]. Finding such a graph would eliminate 9 colors as a potential solution to the Earth-moon problem. A more detailed background on the author's approach and various useful theorems will be provided in Section 2. Section 3 will detail the algorithms created, as well as the reasoning and results of these strategies. Section 4 will provide a run-time analysis of the four thickness heuristics made by the author. An empirical analysis of these algorithms will also be shown to display the speed and accuracy of these algorithms against one

another. Section 5 will list various graphs on 19 vertices which are nearly thickness-2 graphs on 19 vertices. Section 6 will conclude this paper and discuss next steps regarding this approach and the Earth-moon Problem.

2 Background

This section will begin with key definitions. A graph G = (V, E) consists of a set of vertices, denoted V(G), and a set of edges, denoted E(G) [8]. An edge (u, v) is an ordered pair of vertices where u and v are contained in V and (u, v) is contained in E [8]. Vertices u and v are adjacent if they share an edge [8]. A subgraph H = (V′, E′) of G is a graph such that V′ is a subset of V and E′ is a subset of E [8]. A graph is planar if it can be drawn on the two-dimensional plane such that none of its edges cross [8]. The thickness of a graph G is the minimum number of planar subgraphs necessary to partition all of the edges in G, denoted $\Theta(G)$. Namely, an Earth-moon graph will always have a thickness of 1 or 2. Graphs which have a thickness of 1 or 2 are described as biplanar, as they are the union of two graphs. A proper vertex coloring of a graph G is a designation of colors to each vertex in G such that no adjacent vertices are assigned the same color [8]. The chromatic number of G is the minimum number of colors needed to complete a proper vertex coloring of G, denoted $\chi(G)$ [2, p. 2]. In short, the Earth-moon problem asks what the maximum chromatic number of thickness-2 graphs is.

There are various applications for graph thickness. As a concept related to outerthickness and arboricity, these three concepts all have applications regarding "graph drawing, information visualization, VLSI design, and resource location optimization" [10, p. 76]. According to Robert Cimikowski, graph thickness is relevant in "automatic graph drawing systems such as those used by CASE tools in laying out E-R diagrams of database schema" [4, p. 1]. In addition, there are applications to circuit layout as well as facility layout [4, p. 1]. The reason that graph thickness is relevant to VLSI design is that chip-designers need to avoid wire crossings [11, p. 59]. Unfortunately, wire crossings can lead to unwanted signals [11, p. 59]. Crossings can be accommodated through contact cuts, but "too many contact cuts leads to an increase in area and consequently to a higher probability of faulty chips" [11, p. 59]. Knowing the thickness can help to minimize the number of crossings. Similarly, it is desired that the representation of circuit and facility layouts can be represented with fewer crossings or layers that lack crossings, as such diagrams are easier to understand.

Various theorems are particularly useful in determining if a given graph potentially has a thickness of 2. These properties are helpful to know because they can be calculated in polynomial time and serve as bounds to the thickness of a graph. If the bounds suggest that a graph has a thickness greater than 2, then the graph is not an Earth-moon graph and would not be worth inspecting further.

One of these properties is arboricity. Graphs which are acyclic are called forests [8].

Arboricity is defined as the minimum number of spanning forests needed to partition all of the edges in a graph and is denoted as $\Upsilon(G)$. This is a concept very similar to thickness, except that the edges are partitioned into forests instead of planar graphs. Various algorithms can find the arboricity of a graph in polynomial time [3, p. 51]. As noted by Mutzel et al, the arboricity must be less than or equal to the thickness of a graph multiplied by three [11, p. 64]. This is because "a maximal planar subgraph is at most three times as large as a spanning tree" [11, p. 64]. Therefore, the arboricity of a graph can be used to find an upper bound on its thickness. For example, if the arboricity of a graph was 7, then the thickness would need to be at least 3.

Lemma 2.1. $\Upsilon(G) \leq 3 \times \Theta(G)$ [10, p. 64]

Another key property is girth. For a graph G, the girth is the size of the smallest cycle in G [8]. In addition, the number of vertices and the number of edges in a graph both help to determine a thickness bound. Beineke presents a theorem in his paper that provides two inequalities that hold for all graphs with a thickness of 1 or 2 [1, p. 2]. These are as follows:

Theorem 2.2 (Beineke)**.** *Let G be a biplanar graph with p vertices and q edges. Then*

(i) $q \leq 6p - 12$

(ii) $q \leq 2g \times \frac{p-2}{g-2}$ *if the girth of G is at least g.* [9]

The foundation of this strategy to utilize a thickness heuristic algorithm arose from an open problem proposed by Gethner and Sulanke. Gethner and Sulanke note that the following graphs would prove the existence of a 10-chromatic thickness-2 graph:

1. A thickness-2 graph on 19 vertices with a triangle-free complement.

2. A thickness-2 graph on 28 vertices with a K_4-free complement. [7, p. 215]

Alas, determining the thickness of a graph is an NP-Complete problem in the general case [10, p. 77]. As such, a heuristic algorithm would be needed to determine the thickness of a given graph in a reasonable amount of time. However, before determining the thickness of a graph, a graph with the given characteristics from Gethner and Sulanke would need to be found first. That is, the program will need to start by generating either a graph on 19 vertices with a triangle-free complement or a graph on 28 vertices with a K_4-free complement.

In the case of the 19 vertex graph, this program starts by generating a complete graph on 19 vertices. Edges are removed randomly from this graph so long as the graph's complement remains triangle-free. Edges are removed at random so that it is unlikely for the same graph to be produced on subsequent iterations of the program. Once the program fails to remove random edges for a set number of attempts, the graph is tested

to see if it could possibly have a thickness of 2. This includes ensuring that the number of vertices and number of edges fulfills the inequality in Theorem 2.2. In addition, the girth inequality in Theorem 2.2 and arboricity inequality in Lemma 2.1 are also checked. If the graph satisfies these inequalities, the program attempts to partition the edges into two planar subgraphs for a set number of attempts. Hence, the program generates two empty graphs to be used as subgraphs. At random, an edge from the generated graph is selected and placed into one of the subgraphs so long as doing so does not cause the graph to lose planarity. This process repeats until either an edge cannot be placed into one of the subgraphs or all edges are partitioned. In the case that all edges are partitioned, it is proven that the graph has a thickness of 2. Variations in this process are discussed in Section 3.

The 28 vertex case is a very similar process. The complete graph on 28 vertices is generated and edges are removed at random. However, instead of checking if the complement graph is triangle-free after each edge removal, it is instead ensured that the complement is K_4-free. Otherwise, the process is exactly the same. However, the paper will not explore the 28 vertex case. Due to the number of edges in graphs of this size, the heuristic algorithm proves to be much less effective.

3 Developing a Thickness Heuristic for the Earth-moon Problem

As noted by Gethner and Sulanke, finding a thickness-2 graph on 19 vertices with a triangle-free complement proves the existence of a 10-chromatic thickness-2 graph [7, p. 215]. This would eliminate 9 colors as a potential solution to the Earth-moon problem. The process of attempting to generate such a graph is described in detail in this section. Four variations of a thickness heuristic algorithm are presented in this section. The pseudocode for each variation will be provided, as will the worst-case asymptotic run-time of each algorithm.

In order to generate a graph on 19 vertices with a triangle-free complement, the program begins by generating the complete graph on 19 vertices. The program initializes variables to track the number of failed attempts to remove an edge from the graph and the number of edges removed. Until the program fails to remove an edge from the graph a given number of times, the program will continue to remove a random edge from the graph so long as the graph's complement remains triangle-free.

Once a triangle-free graph on 19 vertices has been generated, the program proceeds to check if the graph could have a thickness of 2. By Theorem 2.2, we know that the number of edges in this graph must be less than or equal to 102 if it has a thickness of 2. We know that the complete graph on 19 vertices has 171 edges. Therefore, the program must remove at least 69 edges from the graph. Hence, it is ensured that the variable tracking the number of removed edges is greater than or equal to 69.

If this is the case, the arboricity of the graph is calculated. By Lemma 2.1, the arboric-

ity of the graph must be less than 7 if it has a thickness of 2. If this is the case, the girth of the graph is checked next. By Theorem 2.2, we know that Inequality ii must hold if the graph has a thickness of 2. If the girth equation also holds, this graph is then tested by one of the four thickness heuristics.

3.1 RER (Random Edge Removal)

The first thickness heuristic is named Random Edge Removal (RER) and is the simplest of the four. This heuristic takes an input graph and attempts to partition the edges for a set number of attempts. To do this, two empty graphs on 19 vertices are created and a copy of the input graph is made. These empty graphs will be used as the planar subgraphs in which the edges will be partitioned. The program then selects edges at random and adds the edge to the first subgraph. If this results in the subgraph losing planarity, the edge is removed from this subgraph and inserted into the second subgraph. If this results in the second subgraph losing planarity, then the attempt has failed and the process restarts. However, if the edge can be added to either graph without losing planarity, the edge is deleted from the copy of the input graph. This process continues until all edges have been removed from the copy of the input graph or the program fails to partition the edges after a given number of attempts. If all edges are removed, which means that a a suitable graph has been found, the program will print out the images and Graph6 strings that represent the two planar subgraphs as well as the input graph.

Graph6 allows us to write a graph's data to a string. That is, Graph6 can transform the Graph object in Sage to a string of text. You can then use this Graph6 string to construct the Graph object again in Sage. This makes Graph6 strings one of the ideal way of sharing a graph once it is discovered, as anyone with Sage can use a Graph6 string to recreate a graph and test its properties.

Algorithm 1 Random Edge Removal

Input: G, $attempts$
Output: $sub1$, $sub2$, G or FALSE

```
for i ← 1 to attempts do
    removals ← 0
    // Initialize two empty graphs on 19 vertices called sub1 and sub2
    // Create a copy of the input graph G called copy
    for j ← 1 to |G.E| do
        // Select a random edge u,v from copy
        // Add u,v to sub1
        if sub1 is not planar then
            // Delete u,v from sub1
            // Add u,v to sub2
        end if
```

```
        if sub2 is not planar then
            break
        end if
        removals ← removals + 1
    end for
    if removals == |G.E| then
        return sub1, sub2, G
    end if
end for
return FALSE
```

We can see that the run-time varies depending on the given number of attempts. Let n be the number of vertices in G and m be the number of edges in G. Initialization takes $O(n+m)$ time for an arbitrary graph G. The run-time of determining if a graph is planar is dependent on the algorithm used for this process, but a state-of-the-art algorithm can do this task in $O(n)$ time. Since the outer loop takes $O(1)$ time, as the number of attempts is a constant, and the inner loop takes $O(m)$ time, the overall run-time is $O((m+n)+(m\times(m+n))) = O(m^2+mn)$.

3.2 RER-TT (Triangle Tactic)

The other three variations also use the Random Edge Removal tactic to try and partition all of the edges. The second variation introduces the Triangle Tactic, labeled RER-TT for short. In this version, the algorithm adds a removed edge as normal. After the edge has been successfully added, it checks to see if the edge forms a triangle with any of its adjacent edges. If it does, it adds these two edges to the current subgraph and determines if it is still planar. If it is still planar, the edges are removed from the copy graph and kept in the subgraph. The inner for loop was changed to a while loop for this tactic, as multiple edges can be removed per iteration of the inner loop. In addition, a boolean keeps track of if a triangle was added to subgraph 1 as to prevent adding the triangle to subgraph 2.

The reasoning behind the Triangle Tactic is that dense planar graphs tend to contain lots of triangles. Keeping adjacent edges in the same subgraph can help to reduce the number of edge crossings, as these edges are already meeting at shared vertices.

Algorithm 2 RER: Triangle Tactic

```
Input: G, attempts
Output: sub1, sub2, G or FALSE
  for i ← 1 to attempts do
      removals ← 0
      // Initialize two empty graphs on 19 vertices called sub1 and sub2
```

```
// Create a copy of the input graph G called copy
while |copy.E > 0 do
    triNotAdded = TRUE
    // Select a random edge u,v from copy
    // Add u,v to sub1
    if sub1 is not planar then
        // Delete u,v from sub1
        // Add u,v to sub2
    else
        for each o in u.neighbors do
            if copy has an edge o,v then
                // Make a copy of sub1 called sub1Copy
                // Add edges o,v and o,u to sub1Copy
                if sub1Copy is planar then
                    // Add edges o,v and o,u to sub1
                    // Delete o,v and o,u from copy
                    removals = removals + 2
                    triNotAdded = FALSE
                    break
                end if
            end if
        end for
    end if
    if sub2 is not planar then
        break
    else
        if triNotAdded then
            for each o in u.neighbors do
                if copy has an edge o,v then
                    // Make a copy of sub2 called sub2Copy
                    // Add edges o,v and o,u to sub2Copy
                    if sub2Copy is planar then
                        // Add edges o,v and o,u to sub2
                        // Delete o,v and o,u from copy
                        removals = removals + 2
                        triNotAdded = FALSE
                        break
                    end if
                end if
            end for
```

```
            end if
        end if
        removals ← removals + 1
    end while
    if removals == |G.E| then
        return sub1, sub2, G
    end if
  end for
  return FALSE
```

RER-TT is identical to RER, except that the edge partitioning is held within a while loop and it attempts to add edges forming a triangle with the most recently added edge. The for-each loop executes at most $n-1$ times for a vertex connected to all other vertices. The process of copying sub1 or sub2 is bounded by the number of vertices and edges in the original graph, which means this process has a bound of $O(n+m)$. The rest of the operations for finding a triangle takes constant time. Therefore, since RER takes $O(m^2)$ time, it follows that RER-TT takes $O(m^3+nm)$ time.

3.3 RER-SES (Subgraph Edge Shift)

The third variation uses the Subgraph Edge Shift, RER-SES for short. This version again mirrors RER. However, when the second subgraph first fails to be planar and a given threshold for the number of removals is met, the algorithm removes the recently added edge from subgraph 2 and moves any edge from subgraph 1 to subgraph 2 that can be added without subgraph 2 losing its planarity. This is done by generating and testing the planarity of a copy of subgraph 2. The routine then continues as in RER until completion or failure. Since edges are always added to subgraph 1 first, it is possible that shifting some edges into subgraph 2 may then allow some of the remaining edges to now be added to subgraph 1. The edges previously in subgraph 1 may have been preventing the remaining edges in the copy of the input graph from being added to subgraph 1.

Algorithm 3 RER: Subgraph Edge Shift

```
Input: G, attempts
Output: sub1, sub2, G or FALSE
  for i ← 1 to attempts do
    removals ← 0
    attemptedMove = FALSE
    // Initialize two empty graphs on 19 vertices called sub1 and sub2
    // Create a copy of the input graph G called copy
    while |copy.E > 0 do
        // Select a random edge u,v from copy
```

```
        // Add u,v to sub1
        if sub1 is not planar then
            // Delete u,v from sub1
            // Add u,v to sub2
        end if
        if sub2 is not planar then
            if attemptedMove == FALSE then
                // Delete u,v from sub2
                // Make a copy of sub2 called sub2Copy
                for q, r in sub1.E do
                    // Add q,r to sub2Copy
                    if sub2Copy is planar then
                        // Add q,r to sub2
                        // Delete q,r from sub1
                    else
                        // Delete q,r from sub2Copy
                    end if
                end for
                attemptedMove ← TRUE
            else
                break
            end if
        end if
        removals ← removals + 1
    end while
    if removals == |G.E| then
        return sub1, sub2, G
    end if
end for
return FALSE
```

RER-SES is also identical to RER, except it tries to move edges from sub1 to sub2 after it first fails to add an edge to sub2. This might move the conflicting edges out of sub1 and allow the edge that failed to be added to sub1 to now be inserted without sub1 losing planarity. Creating a copy of sub2 takes $O(n+m)$ time in the worst-case. Similarly, the added for loop executes at most m times. The operations within this for loop take constant time except for checking its planarity, which takes $O(n)$ time. Thus, this added step takes $O(n+m+nm) = O(nm)$ time. So, from the run-time of RER, RER-SES runs in $O(nm^3)$ time.

3.4 RER-TT-SES (Triangle Tactic and Subgraph Edge Shift)

The last variation combines the Triangle Tactic and Subgraph Edge Shift, denoted RER-TT-SES. The reasoning for this method is the same as the previous two versions. While it might be slower from a computational standpoint, more edges might be removed before the algorithm fails when compared to the original version of this algorithm. As such, this algorithm may outperform its counterparts.

Algorithm 4 RER-TT-SES

Input: G, $attempts$
Output: $sub1$, $sub2$, G or FALSE
  for $i \leftarrow 1$ to $attempts$ **do**
    $removals \leftarrow 0$
    $attemptedMove \leftarrow$ FALSE
    // Initialize two empty graphs on 19 vertices called sub1 and sub2
    // Create a copy of the input graph G called copy
    while $|copy.E| > 0$ **do**
      $triNotAdded$ = TRUE
      // Select a random edge u,v from copy
      // Add u,v to sub1
      if sub1 is not planar **then**
        // Delete u,v from sub1
        // Add u,v to sub2
      else
        for each o in $u.neighbors$ **do**
          if copy has an edge o,v **then**
            // Make a copy of sub1 called sub1Copy
            // Add edges o,v and o,u to sub1Copy
            if sub1Copy is planar **then**
              // Add edges o,v and o,u to sub1
              // Delete o,v and o,u from copy
              $removals \leftarrow removals + 2$
              $triNotAdded \leftarrow$ FALSE
              break
            end if
          end if
        end for
      end if
      if sub2 is not planar **then**
        if $attemptedMove ==$ FALSE **then**
          // Delete u,v from sub2

```
                // Make a copy of sub2 called sub2Copy
                for q, r in sub1.E do
                    // Add q,r to sub2Copy
                    if sub2Copy is planar then
                        // Add q,r to sub2
                        // Delete q,r from sub1
                    else
                        // Delete q,r from sub2Copy
                    end if
                end for
                attemptedMove ← TRUE
                break
            end if
        else
            if triNotAdded then
                for each o in u.neighbors do
                    if copy has an edge o,v then
                        // Make a copy of sub2 called sub2Copy
                        // Add edges o,v and o,u to sub2Copy
                        if sub2Copy is planar then
                            // Add edges o,v and o,u to sub2
                            // Delete o,v and o,u from copy
                            removals = removals + 2
                            triNotAdded = FALSE
                            break
                        end if
                    end if
                end for
            end if
        end if
        removals ← removals + 1
    end while
    if removals == |G.E| then
        return sub1, sub2, G
    end if
end for
return FALSE
```

RER-TT-SES combines RER-TT and RER-SES. Since these operations occur independently from one another, we can simply add together the bounds of their run-times. Thus, RER-TT-SES runs in $O((m^3 + nm) + nm^3) = O(nm^3)$ time.

3.5 Attempts to find a thickness-2 graph on 19 vertices with a triangle-free complement

In general, RER-TT-SES gets the closest to determining if a graph has a thickness of 2 or not, as it is able to remove more edges than the other variations until it fails. The addition of the triangle tactic appears to help the most, likely because it keeps edges that do not cross each other together. However, the edge shifting is also beneficial. Section 4 has a more in-depth analysis of how these algorithms compare to one another in terms of efficiency and accuracy.

While this strategy has yet to produce a thickness-2 graph with a triangle-free complement, it is easy to see that such a graph, in the very least, is close to having a thickness of 2. With this strategy, various graphs have been shown to be within three edges of having a thickness of 2. Some of these graphs have an arboricity as low as 5, too. A table listing these "close" graphs can be found in Section 6.

4 Empirical Analysis of Graph Thickness Heuristics

For the empirical analysis, a set of thickness-2 graphs were made to test each algorithm for accuracy, the number of steps taken until reaching a solution, and the number of attempts until reaching a solution. This will allow for the algorithms to be compared to one another in these regards. The thickness-2 graphs were made by removing edges from a complete graph on a given number of vertices, then adding these removed edges to one of two planar subgraphs. The edge was added so long as the subgraph remained planar, and this process repeated until it failed to add an edge a given number of times. Twenty graphs were made for each number of vertices, from 16 vertices to 45 vertices. The results of this analysis are given in the tables and graphs below.

The Success Rate Average represents the percent of graphs that an algorithm was able to show had a thickness of 2 after a set number of attempts. This gives us a means to compare the algorithms in terms of accuracy. Evidently, RER-TT-SES has the highest Success Rate Average and is thus considered the most accurate of the four algorithms. Similarly, RER is shown to be the least accurate.

The Average Attempts for a given number of vertices refers to the average number of attempts taken for the algorithm to determine that the graph has a thickness of 2. Each algorithm has an input parameter of *attempts*, which is the number of times that the outer for loop will execute the algorithm before deciding that the graph has a thickness greater than 2. If the loop was executed 5 times and found a suitable partition of edges on the 5th attempt, then we consider that the algorithm took 5 attempts. The Average Attempts gives a measurement of efficiency for each algorithm. A lower average indicates that the algorithm took less iterations of the outer loop, and thus less time to find a solution. Across most sets of graphs, the RER-TT-SES algorithm had the lowest Average Attempts while RER had the greatest Average Attempts.

Similarly, Average Code Lines represents the average number of lines of code executed before the algorithm found a solution. The process of counting the number of lines of code executed is a very common practice in determining the efficiency of algorithms. This is a more accurate representation of efficiency, as each attempt takes a different number of code lines across algorithms. Even so, RER-TT-SES is still shown to be the most efficient on average. Likewise, RER is shown to be the least efficient.

In summary, RER-TT-SES outperforms the other three algorithms in terms of accuracy and lines of code used. Since RER is the most random of the three approaches, it follows that it generally takes more attempts for this algorithm to succeed. From this, adding triangles to the planar subgraph when possible appears to be a very effective strategy in comparison to simply picking individual edges at random.

One issue to note with this data is that the graphs generated for analysis were purposefully made to be less dense. In general, dense thickness-2 graphs have fewer partitions into two planar subgraphs due to the increased number of edges. The odds of any of these algorithms finding one of an exponential number of partitions is fairly low. So, in order to analyze the algorithms efficiently and allow for any algorithm to be successful, the graphs were made to be less dense.

RER	RER-TT	RER-SES	RER-TT-SES
96.833%	99.167%	98.833%	99.500%

Table 1: Success Rate Average

Vertices	RER	RER-TT	RER-SES	RER-TT-SES
16	831.5	657.5	453.8	143.5
17	2699.0	976.4	1059.5	514.3
18	290.7	43.5	46.6	19.6
19	343.0	108.9	96.9	54.6
20	1213.0	912.2	1193.9	591.6
21	709.3	71.6	221.5	25.2
22	599.5	66.7	96.7	35.5
23	1276.9	295.2	580.3	35.6
24	884.4	527.3	558.5	510.1
25	1351.1	520.3	767.9	511.8
26	181.7	28.3	76.5	10.2
27	34.7	8.9	15.7	5.4
28	342.5	44.5	43.6	24.9
29	576.3	24.9	54.9	54.4
30	1239.5	641.5	916.8	251.6
31	602.3	39.1	165.5	9.9
32	172.2	12.1	37.7	6.4
33	117.7	7.5	29.8	5.3
34	210.5	27.7	14.8	7.4
35	909.0	16.6	521.4	40.5
36	517.3	11.9	78.4	4.7
37	82.8	11.6	22.2	9.7
38	83.6	8.4	24.3	7.6
39	127.1	18.3	13.1	8.6
40	59.8	8.4	9.9	2.8
41	51.0	13.9	12.7	4.2
42	80.9	7.8	14.4	4.4
43	80.1	6.9	20.9	2.9
44	55.1	5.3	16.9	4.6
45	819.6	32.1	215.0	5.4

Table 2: Average Attempts

Vertices	RER	RER-TT	RER-SES	RER-TT-SES
16	1946243.1	1747551.1	1058002.2	428454.0
17	6932390.6	2830915.2	2697007.1	1741522.8
18	822508.3	136357.0	136480.2	87356.3
19	1054733.2	369130.4	301017.2	239441.4
20	3990290.8	3356023.8	3893267.1	2290166.3
21	2494570.3	275971.6	804383.9	136753.3
22	2324129.3	283256.4	376974.1	197525.5
23	5253401.1	1346875.5	2352290.8	189481.8
24	3875497.5	2559535.5	2415102.1	2501150.3
25	6297130.7	2693690.6	3543051.7	2654665.0
26	895590.0	150792.8	385557.4	79022.1
27	184129.8	49753.0	86970.4	43502.8
28	1889273.4	265156.3	252083.4	187229.8
29	3360824.8	157102.5	325154.4	394253.7
30	7646247.7	4317753.8	5581355.5	1742799.1
31	4010111.5	280785.1	1098662.3	94569.0
32	1195564.2	87127.0	265711.0	67591.0
33	857118.7	57780.6	219427.9	56102.9
34	1605871.9	223945.1	116639.1	76188.4
35	7238911.5	141677.9	4150685.5	365367.2
36	4273401.5	103430.9	648478.4	61810.5
37	722716.4	106335.2	197706.2	117193.1
38	775083.1	79796.6	229143.7	99025.5
39	1184099.8	177375.5	123930.7	106584.4
40	590420.7	87290.9	102902.6	44374.0
41	521278.4	146871.3	134521.1	63027.4
42	866655.5	86838.4	159118.3	71832.7
43	894946.9	79721.3	241975.0	48112.9
44	631917.8	62583.1	197014.2	76211.2
45	9507958.3	392662.7	2477576.7	82646.2

Table 3: Average Code Lines

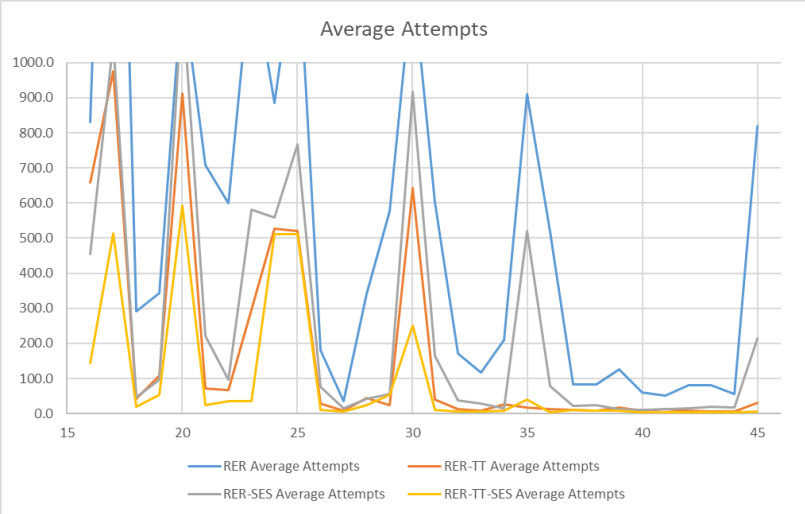

Figure 2: Average Attempts

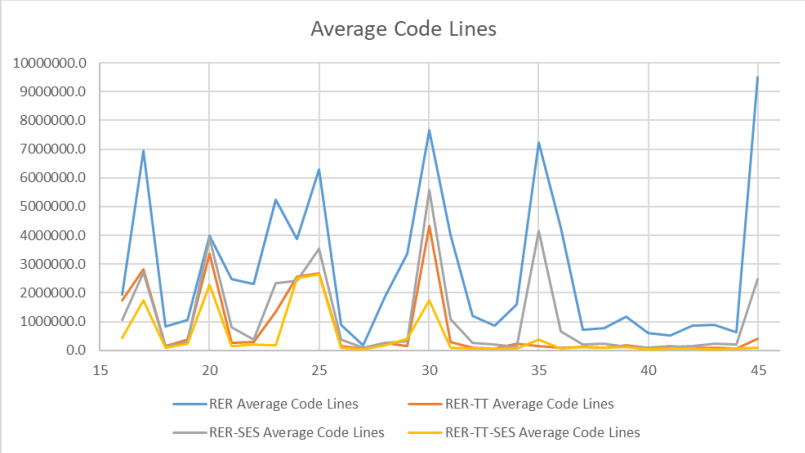

Figure 3: Average Code Lines

5 Nearly Thickness-2 Graphs on 19 Vertices

The table below contains graphs on 19 vertices which nearly have a thickness of 2. In these graphs, RER-TT-SES is able to partition all but five or less edges into two planar subgraphs. Closeness is defined as the number of times that the graph's edges are partitioned into two planar subgraphs with only five or less edges left to insert into the planar subgraphs out of 100,000 attempts. The table also includes Graph6 strings with the planar subgraphs containing the maximum number of edges removed. It is possible that one of these graphs has a thickness of 2, but this has yet to be determined.

G6 String	Closeness	Edges	Max Edges Removed	Subgraph 1 G6 String	Subgraph 2 G6 String	Arboricity
ReseuscupHCdusHJCdmucHJWclmuc_	6169	81	78	RE_A???oPG?duC@???A__@IG?Dk___	R?ScuscE_@C??oGJCdkECG@Oc_AUC?	5
RQTY|A`g\BsNg^g^joA{@V_L]?y{@w	6121	81	78	R?PAG??_GAOG_X_Yi_???O_LO?wG@O	RQCWTA`GD@cFGEGD@OA{@F??I?As?g	5
RhrNXbMrhaEHKRrkEHpa\KRm\~~\a?	368	89	86	R?`KX`I?_?A???AGC@OaPCQiWxWH??	RHQB?ACrHaCHKRpc?G`?KG@C@EfSa?	5
R`ouPjIqhiMlLRnuxU`i]q1GtNUTbO	281	89	86	R@?a?aA?g?MKHR@?GS`?OQh?dBCC?O	R?oCPHGq@i?`C?mupA?iM_CGOKQPa?	5
R`N@uGyMhtfV~QMnW}mIDpOnDA}IDw	249	89	86	R?@?O?IA``_A@?CLWUmA?o?HC?M?Cw	R`K@EGoKGSFT}OIa?g?GD@Oe@AoI@?	5
Rwe^FE[fJeRefLwqMXNERwq][u?x|G	238	90	87	RoCSC?Cb@C??BCga?ODA?gaMKs?X??	RGaJBEWCI_PecHOOMHI?ROOOOA?_|G	5
RnGh[[n{Ead{sU^1YJP^IJxVPYeatG	147	91	88	RMG_K?c[?A_oCQ[1OA?C?J_TO?E?`?	R`?GO[J_E_CKoCB?IHPZI?XA@Y_aOG	6
ReJPUjTI`SykIeyedR]jIye~TcP~|G	132	91	88	R_?OA`D?`S_K?cqE`?AIAQa{?c?b?G`	REJ@CIOI??Y_IAG_CP[`GgCBT?@[|?	6
Rv~~x?`G^{^pG[G[cM]{WvbD@rxBfG	74	90	87	RS]EP?_G\[?OG?GOCC?CW`b?@Q`?@?	Rb_xg?@?A_^`?[?K_A]w?S?D?`WBeG	5
RTPLaXLV|cURkfR[HmNZRkbd^vdc[o	62	92	89	R@@GaO@Oh?SQ?@ROGKD?Ogb@VUD??_	RSOD?HGFScA@ke?G@aIZBC?cG`__[O	6
RKdbJMW^zMrCeLeLrCzMo^}uWffMXg	49	93	90	R?d_?COSXKpC?GC@_??EoSy_Wcc?HG	RK?BJA?JaAA?eDaKRCxG?JCU?BBMO_	6
Res`HNWuHMVxnruKCxupmuLsfMbNrG	42	93	89	R_O_@HGE?ISwL?aG?OsoKaDo_Ea@??	REc@GEOoHCB@arSCC@A@_SGCF?@MrG	6
RpNeHpZU~K[pxbM[FMML{M[XZp~K[_	25	93	89	ROBe?`BOF?C_G?CW?CEG?EKPRoZGO?	R_K?HOWEwKWOpbI?FIGDsGOGG@cCK_	6
RvGh^bpJfbDq{YJdYLVbTJdRpiq[jO	22	90	86	RuG`PA?GFA??OAI@Q?A_?GdRpa?CI?	R@?GM@pB_@@qkW@cGLDBTB???GqW`O	5
R`^LeWyrUYuSrUkhFSmYsrUcydeM}o	7	92	88	R?PK_?_oEQS?AOKh??G?OqEcAdeCy?	R`M??WYBOGaSpE_?FSeYc@O?w??I?o	6

Figure 4: Nearly Thickness-2 Graphs on 19 Vertices

6 Conclusion

This paper explored a programming-attack on the Earth-moon Problem using thickness heuristics. While this approach has yet to discover the desired thickness-2 graph on 19 vertices with a triangle-free complement, RER-TT-SES has gotten within 3 edges of partitioning all of a graph's edges into two planar subgraphs for various graphs. In addition, it was shown that RER-TT-SES was the most effective of the four heuristic algorithms created by the author through an empirical analysis. Perhaps with some more attempts, this strategy will find a suitable graph.

Noted previously, finding a thickness-2 graph on 28 vertices with a K_4-free complement would prove the existence of a 10-chromatic thickness-2 graph [7, p. 215]. However, a major flaw in this strategy is the difficulty in removing enough edges from the complete graph. By Theorem 2.2, a thickness-2 graph on 28 vertices can have at most 156 edges. Since the complete graph on 28 vertices has 378 edges, at least 222 edges need to be removed at random from the graph for it to possibly have a thickness of 2. The program struggles to get past this step in general. Furthermore, determining if the complement is K_4-free is a more difficult task than determining if a graph is triangle-free. As such, the inability to generate a K_4-free graph with enough edges removed has stalled these attempts. A more effective strategy of generating such graphs would be needed to continue with the heuristic strategy as in Section 3.

Regarding next steps, it may be beneficial to investigate other heuristics for determining the thickness of a graph. Doing so may improve accuracy, efficiency, or perhaps both when it comes to determining a given graph's thickness. In addition, the K_4-free complement graph approach was halted by an inability to generate such a graph that could possibly have a thickness of 2. Refining the generation of such graphs would allow for this heuristic strategy to be utilized.

References

[1] L. W. Beineke. Biplanar graphs: a survey. *Computers and Mathematics with Applications*, 34(11): 1-–8, 1997.

[2] D. L. Boutin, E. Gethner, and T. Sulanke. Thickness-two graphs part one: new nine-critical graphs, permuted layer graphs, and Catlin's graphs. *Journal of Graph Theory*, 57(3): 198-–214, 2008.

[3] A. Chatzigeorgiou et al. *SOFSEM 2020 : Theory and practice of computer science*. Springer International Publishing AG, 2020.

[4] R. Cimikowski. On heuristics for determining the thickness of a graph. *Information Sciences*, 85(1-3): 87–98, 1995.

[5] M. Gardner. Mathematical Games. *Scientific American*, 242(2): 14–23, (1980).

[6] E. Gethner. To the moon and beyond. *Graph theory: Problem books in mathematics*. Ed. by R. Gera, T. Haynes, and S. Hedetniemi. Springer, 2018.

[7] E. Gethner and T. Sulanke. Thickness-two graphs part two: more new nine-critical graphs, independence ratio, cloned planar graphs, and singly and doubly outerplanar graphs. *Graphs and Combinatorics*, 25: 197—217, 2009.

[8] D. Guichard. *Combinatorics and graph theory*. https://www.whitman.edu/mathematics/cgt_online/book/.

[9] B. Jackson and G. Ringel. Variations on Ringel's earth-moon problem. *Discrete Mathematics* 211: 233–242, 2000.

[10] E. Mäkinen and T. Poranen. An annotated bibliography on the thickness, outerthickness, and arboricity of a graph. *Missouri Journal of Mathematical Sciences*, 24(1): 76–87, 2012.

[11] P. Mutzel, T. Odenthal, and M. Scharbrodt. The thickness of graphs: A survey. *Graphs and Combinatorics*, 14: 59–73, 1998.

[12] I. Stewart. Empires on the moon. *Scientific American*, 277(2): 86-–88, (1997).

[13] M. Wichman and H. Critchfield. *Exploring the earth-moon problem for doubly outerplanar graphs*.

Robert Weaver
York College of Pennsylvania
robbycw9@gmail.com