Edge Turncoat Graphs in the Game of Cops and Robbers: The good, the bad and the ugly

Colin Garnett
Black Hills State University
1200 University St., Spearfish SD, 57799-9003
Colin.Garnett@bhsu.edu

Kerry Tarrant
University of Iowa
14 MacLean Hall, Iowa City IA 52242-1419
Kerry-Tarrant@uiowa.edu

Jeffrey Winter
Black Hills State University
1200 University St., Spearfish SD, 57799-9003
Jeffrey.Winter@bhsu.edu

Dear Gary,
A very happy Birthday with many happy returns! If you find yourself in the Black Hills let us know and we can take you out for a meal.
Sincerely,
Colin Garnett, Jeffrey Winter and Kerry Tarrant

Abstract

The game of cops and robbers on a graph is a vertex pursuit game played by two players with perfect information. Per the rules of the game, a given graph is either inherently cop-win or robber-win. It is possible that adding *any* edge changes the inherent nature of a particular graph. Such a graph is *maximal* in the sense that no edge can be added without changing its "win-state". Furthermore, if deleting *any* edge changes the "win-state", then this graph is *minimal*. Join us as we walk this thin blue line between cop-win and robber-win and explore the good, the bad and the ugly.

JCMCC 116 (2021), pp. 143-168

1 The Good: Definition and Preliminary Theorems

Cops and robbers is a vertex-pursuit game played on a simple graph by two players. Given a graph G the set-up and rules for the game are as follows. The players are assigned a role, one is the cop and the other is the robber. Each player has a single piece to play. The game begins with the cop player placing her/his piece on one vertex in the graph G. After the cop has placed her/his piece, the robber will place his/her piece on one vertex in the graph G. After each player has chosen the placement of their piece, the game starts with the cop taking the first turn. On a turn the cop or robber may move his/her piece to any adjacent vertex connected by a single edge to the player's currently occupied vertex, or they may leave their piece on the current vertex. Play alternates between the cop player and the robber player until either the cop occupies the same vertex as the robber, in which case the cop wins, or the robber can prolong the play indefinitely guaranteeing that the cop will never occupy the same vertex as the robber, in which case the robber wins.

A graph on which there is a strategy that the cop player can employ to win this game is called *cop-win*. A graph on which there is a strategy that the robber player can employ to win this game is called *robber-win*. Given a simple graph G it is inherently either cop-win or robber-win and never both, as was shown in [2]. How robust is the inherent state given the addition or deletion of edges in the graph? It is natural to investigate whether the deletion or addition of any edge switches this state. In order to proceed we will need the following definitions and preliminary theorems.

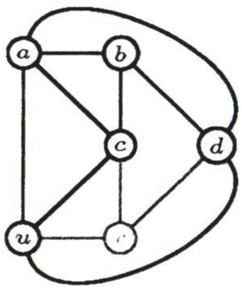
The game of cops and robbers is played on a simple graph and we will only consider simple graphs in this paper. Some knowledge of the terminology of simple graphs is assumed and may be found in any book on graph theory, for example [4] or [7]. We will provide some nonstandard definitions for clarity.

We define the path P_n to be the path with n edges and we say that this path has length n. A path with end vertices u and v is a **non-branching path** if every vertex on this path, with the possible exception of the end vertices, has degree 2. A **unicyclic graph** is a connected graph containing exactly one cycle. The **join** of two graphs G and H is the graph formed by adding an edge from each vertex in G to all vertices in H.

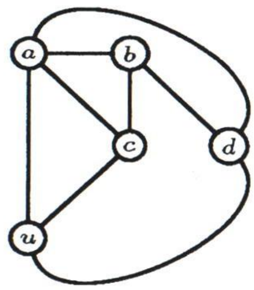
A graph G is called **self-centered** if the center consists of all of the vertices of the graph and it is called **k-self-centered** if G is self-centered and the radius and the diameter are both k.

The following technical definitions are relevant to the study of cops and robbers and can be found in [2]. Vertex u **dominates** vertex v if the closed neighborhood of v is a subset of the closed neighborhood of u. In this case, v is called a **corner** and v is dominated by u. When vertex u dominates vertex v, we perform a **retraction** on the graph G by deleting v and the edges connected to v in G to for a graph H. We say v **retracts onto** u, and H is a **retract** of G. A graph G is **dismantlable** if there is a sequence of retractions on G that results in a single vertex.

Example 1.1. The following graphs have the property that vertex v retracts onto vertex u and H is a retract of G.

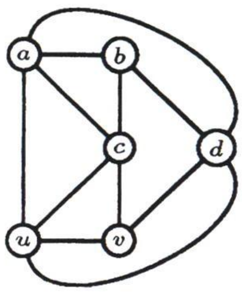


Graph G

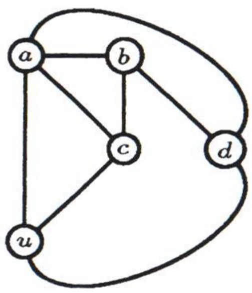


Graph H

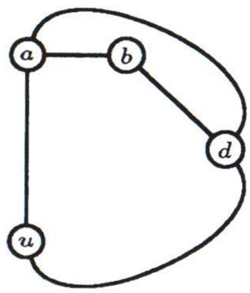
Example 1.2. The following graphs demonstrate a sequence of retractions for a dismantlable graph.



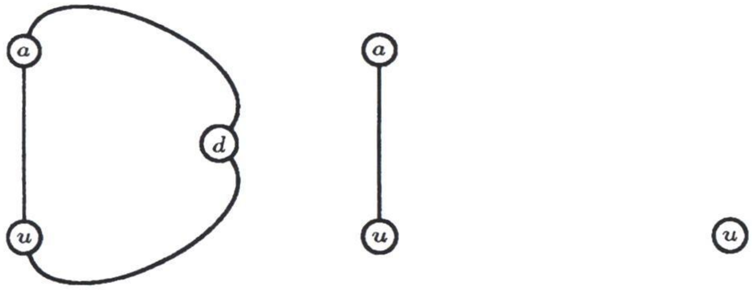
Stage 1:
Original Graph



Stage 2:
v retracted onto u



Stage 3:
c retracted onto a



Stage 4: *b* retracted onto *a*   **Stage 5:** *c* retracted onto *a*   **Stage 6:** *a* retracted onto *u*

The preliminary theorems below will give us some tools that will help us investigate the robustness of cop-win and robber-win graphs.

Lemma 1.3. *[2, Corollary 1.10]*
If G is cop-win, then so is each retract of G.

Lemma 1.4. *[2, Lemma 2.1]*
If G is a cop-win graph with at least two vertices, then G contains at least one corner.

Lemma 1.5. *[1] If G contains a universal vertex, then G is cop-win.*

There has been some work published on questions of critical values in graphs [3]. Specific to the interests of this paper, we focus on critical values on edges investigated in [8], [5], and [6]. These graphs change state from cop-win to robber-win or vice-versa with the addition (or deletion) of any edge. A graph that changes state with the addition (or deletion) of an edge will be called an **edge turncoat graph**. An edge turncoat graph that changes with the addition of an edge will be called **maximal** and one that changes with the deletion of an edge will be called **minimal**. Specifically there are four types of edge turncoat graphs, namely **edge minimally cop-win**, **edge maximally cop-win**, **edge maximally robber-win**, and **edge minimally robber-win** graphs. In Sections 2 and 3 we will describe edge maximally robber-win graphs and characteristics and produce an algorithm for determining edge maximally robber-win, but we stop short of a characterization. In Section 4 we work on edge minimally robber-win graphs and again stop short of a characterization.

The authors in [8] and [5] have characterized edge minimally cop-win graphs and edge maximally cop-win graphs. We include their characterizations with our terminology below.

The definition of such a graph is minimal in the sense that deletion of any edge will result in a robber-win graph. In other words, every edge is necessary in an edge minimally cop-win graph for maintaining a cop-win outcome. It turns out that the property defining edge minimally cop-win graphs is very restrictive. As shown in [5] the only edge minimally cop-win graphs are trees.

Theorem 1.6. *[5] A graph G is edge minimally cop-win if and only if G has at least 2 vertices and is a tree.*

In [5] they also describe the type of graph that becomes robber-win with the addition of any edge.

This leads to the following Theorem.

Theorem 1.7. *[5] There are no graphs that are edge maximally cop-win.*

The types of cop-win graphs that change in outcome due to adding and deleting edges is quite limited compared to the types of graphs in the following sections.

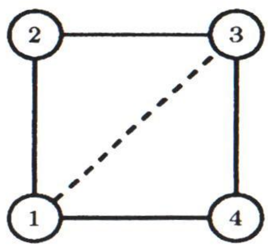
2 THE BAD: Edge Maximally Robber-win

We now turn our attention to robber-win graphs. The graphs that are labeled as cop-win edge-critical graphs are characterized for planar graphs in [6]. These particular robber-win graphs become cop-win with the addition of any non-edge, which we rename in our definition below.

Definition 2.1. An **edge maximally robber-win** graph G is a graph that is robber-win and the addition of any edge results in a cop-win graph. Fitzpatrick refers to such graphs as *cop-win edge-critical with respect to addition (CECA)* in [6].

These types of graphs are maximal in the sense that adding any edge to the graph will change the graph from robber-win to cop-win.

Example 2.2. The graph of a 4-cycle, C_4, is an example of an edge maximally robber-win graph. Observe that C_4 is robber-win and that the addition of either of the two diagonal edges results in a universal vertex and therefore a cop-win graph.

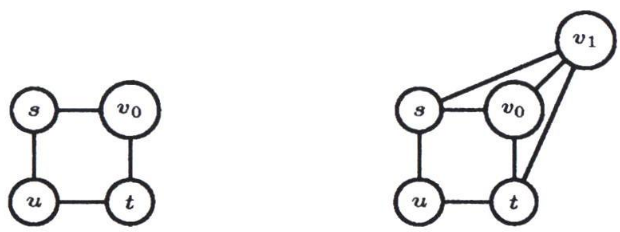


Example 2.2 can be generalized to any $n-2$ regular graph such that n is an even positive integer, as shown in [8]. The addition of an edge to such a graph will result in two universal vertices.

Theorem 2.3. *[8, Theorem 4.37] Take n to be an even positive integer, all $n-2$ regular graphs on n vertices are edge maximally robber-win.*

We diverge from [5] and [8] by including edge maximally robber-win graphs that contain corners. The next example describes the construction process for creating a non-regular edge maximally robber-win graph.

Example 2.4. Consider the following construction: starting with a vertex v_0, add a new vertex v_1 and an edge from v_1 to v_0 and an edge from v_1 to all neighbors of v_0.

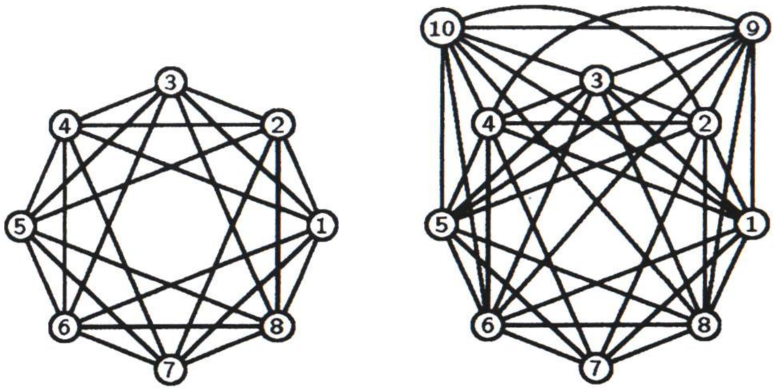


G_1 G_2

The resulting graph G_2 is robber-win because the only retractions possible result in C_4 which is robber-win. Therefore, by the contrapositive of Lemma 1.3, it follows that G_2 must also be robber-win. The addition of any edge to G_2 will result in at least one universal vertex, therefore the addition of any edge will result in a cop-win graph. Combining these facts we conclude that G_2 is edge maximally robber-win.

This construction can be done on any $n-2$ regular graph described in Theorem 2.3.

Example 2.5. Begin with a 6 regular graph on 8 vertices, labeled G_1 in the figure below. Consider the following construction: starting with vertex 3, add a new vertex 9, and an edge between vertex 9 and vertex 3 and edges between vertex 9 and all the neighbors of vertex 3. Similarly, add vertex 10 and edges between vertex 10 and vertex 3, all the neighbors of vertex 3, and including vertex 9. The result of this construction is G_2 in the figure below.



G_1 G_2

There are two retractions possible: vertex 9 retracts to vertex 3 and vertex 10 also retracts to vertex 3. The resulting graph is G_1 and so it is robber-win. The addition of any edge missing from G_2 results in at least one universal vertex, thus G_2 is edge maximally robber-win.

This construction can be better clarified by the following definition from [9].

Definition 2.6. Given a graph G and a vertex v of the graph, **substituting** the vertex v with a graph H means delete vertex v and connect all neighbors of v to all vertices in H.

Note that in Example 2.5 that vertex 3 is being substituted with the complete graph K_3. In fact, larger edge maximally robber-win graphs can be constructed by further substitution for any vertex in a smaller edge maximally robber-win graph by a complete graph.

Theorem 2.7. *Let G be an edge maximally robber-win graph. Then by substituting any number of vertices of G with complete graphs of any size we obtain a graph that is edge maximally robber-win.*

Proof. Let G be an edge maximally robber-win graph on n vertices and substitute a vertex v with the complete graph K_r and call the resulting graph G'. Notice that in G' all vertices in K_r can retract to a single vertex and the result is G where the retraction of K_r is v, hence G' is robber-win. When we add an edge to G' we must show that the result is cop-win. Note that no edges may be added to K_r and so there are two cases.

Case 1:
The two vertices on the new edge are disjoint from K_r. In this case the vertices of K_r retract to a single vertex and the result is the graph G with

one additional edge. Now since G is edge maximally robber-win it follows that G with the additional edge is cop-win, therefore G' with the additional edge is also cop-win.

Case 2:
Suppose the edge uv is added to G, where u is a vertex in K_r and v is not. Notice that the neighborhood of u contains all neighborhoods of the other vertices in K_r, therefore these vertices retract onto u. Now this is again G with one additional edge, incident with v. Since G is edge maximally robber-win it follows that G with the additional edge is cop-win, therefore G' with the additional edge is also cop-win.

Therefore substituting one vertex with a complete graph results in a new edge maximally robber-win graph. Continuing this process we can replace any number of vertices in an edge maximally robber-win graph with complete graphs of any size. □

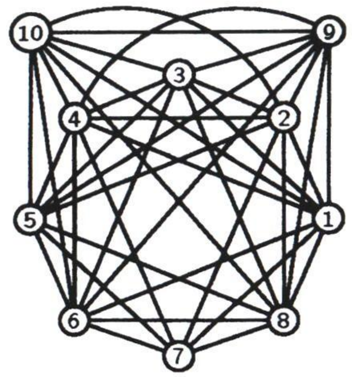
Corollary 2.8. *A sequence of K_2 substitutions in a graph is the same as substitution by complete graphs on one or more vertices.*

Proof. Any given sequence of K_2 substitutions on r vertices with the same neighborhood will be equivalent to substituting by K_r on one of the vertices. Thus any sequence of K_2 substitutions will lead to substitution by complete graphs on one or more vertices. □

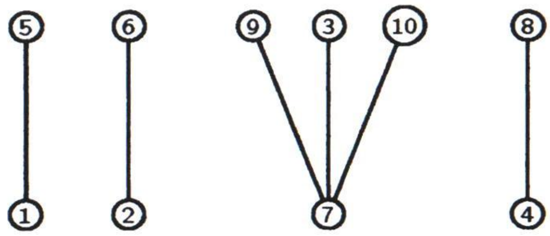
Notice that Corollary 2.8 suggests that we only need to consider sequences of K_2 substitutions on an edge maximally robber-win graph rather than substitutions by larger complete graphs. We will see more of the relevance of this corollary in Section 3.

The downside to using substitution to create larger edge maximally robber-win graphs is the abundance of edges. In order to deal with the increased number of edges it will be beneficial to consider the complement of the graph. Consider the complement of G_2 from Example 2.5 shown in Figure 2.9 In the complement this is substituting vertex 3 with the empty graph on three vertices, $\overline{K_3}$. In general, substituting H for vertex u in G corresponds to substituting $\overline{H}$ for vertex u in $\overline{G}$.

Figure 2.9. Consider the complement of the graph G_2 from Example 2.5.



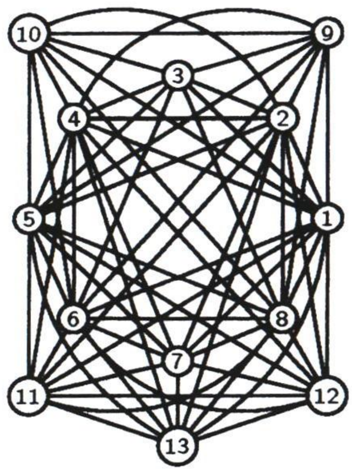
G_2



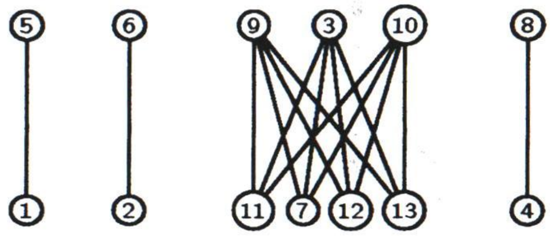
$\overline{G_2}$

Note that when an edge is added to G_2 we obtain a universal vertex. This is convenient but not necessary as more substitutions are performed.

Figure 2.10. Continuing from Example 2.5, see Figure 2.9, substitute vertex 7 with the complete graph K_4.



G_3



$\overline{G_3}$

Since the graph G_2 in Figure 2.9 is edge maximally robber-win we can conclude that G_3 is also edge maximally robber-win by Theorem 2.7. Notice that the addition of the edge between 7 and 3 in G_3 does not result in a universal vertex.

The graphs G_2 and G_3 demonstrate the motivation for working in the complement.

The neighborhood of edge maximally robber-win graphs was studied by Hill [8]. He showed that corners share the same closed neighborhood as the their dominating vertex.

Theorem 2.11. *[8, Lemma 4.28] If G is edge maximally robber-win and vertex u retracts to vertex v in G, then $N[u] = N[v]$. Furthermore $\overline{N[u]} = \overline{N[v]}$ in $\overline{G}$. Note that $\overline{N[u]}$ is the complement in the vertex set of $N[u]$ and does not include vertex u.*

We use this theorem and the below Theorem 2.12 to obtain further results about edge maximally robber-win graphs.

Theorem 2.12. *If G is edge maximally robber-win and G' is a retract of G, then G' is also edge maximally robber-win.*

Proof. If G is edge maximally robber-win and G' is a retract of G then G' is robber-win by Theorem 2.3 in [2]. Without loss of generality G' is obtained from G by retracting vertex u to vertex v. We want to show that adding an edge to G' will result in a cop-win graph. Notice that adding an edge to G' is the same as adding an edge to G that does not contain vertex u and then retracting u onto v. Note that this retraction is still possible regardless of the choice of edge, since the closed neighborhood of u is still contained in the closed neighborhood of v. Since G is edge maximally robber-win it follows that G with this additional edge is cop-win and by Corollary 1.10 in [2] it follows that G' with this additional edge is also cop-win. Therefore G' is edge maximally robber-win. □

Corollary 2.13. *Let G be an edge maximally robber-win graph. Every retraction in G is the result of a K_2 substitution.*

Proof. By Theorem 2.11 if vertex u retracts onto vertex v in G they share all of the same neighbors, which implies that u could be created by a K_2 substitution on v in the retract of G. By Theorem 2.12 each retract will still be edge maximally robber-win and so Theorem 2.11 will still hold. Therefore subsequent retractions will also be the result of a K_2 substitution. □

Theorem 2.14. *If G is edge maximally robber-win, then every retraction in a sequence of retractions is already possible in the original graph G.*

Proof. By Corollary 2.13 every retraction in a sequence of retractions is the result of a K_2 substitution. By Corollary 2.8 every retraction possible in an edge maximally robber-win graph will be the result of substitution by complete graphs on one or more vertices. Therefore every possible retraction in a sequence of retractions will already be possible in the original graph. □

Theorem 2.15. *If G is an edge maximally robber-win graph and $\overline{G}$ contains a leaf, then the connected component of $\overline{G}$ containing the leaf is a nontrivial star.*

Proof. Call the leaf in $\overline{G}$ vertex v and its unique neighbor in $\overline{G}$ vertex w. For each $u \in \overline{N[w]}$ with $u \neq v$ it follows that $\overline{N[v]} \subseteq \overline{N[u]}$. Therefore every vertex in $\overline{N[w]}$, other than v, will retract into v. By Theorem 2.11 it follows that all neighbors of vertex w in $\overline{G}$ are also leaves. Therefore the connected component in $\overline{G}$ containing vertex v is a nontrivial star. □

Not only are all edge maximally robber-win components that contain a degree 1 vertex in the complement nontrivial stars, but all edge maximally robber-win that contain trees in the complement can be characterized as such. The next theorem shows all trees and forests in the complement of an edge maximally robber-win graphs are nontrivial stars and vice versa.

Theorem 2.16. *Let G be a graph such that $\overline{G}$ is a tree or a forest. The graph G is edge maximally robber-win if and only if $\overline{G}$ is a nontrivial star or a disjoint union of nontrivial stars.*

Proof. In the forward direction, G is an edge maximally robber-win graph such that complement $\overline{G}$ is a tree or a forest. Since each connected component in $\overline{G}$ is a tree, each consequently contains a vertex of degree 1. By Theorem 2.15 each of the components of $\overline{G}$ will be a nontrivial star. Thus the complement is either a nontrivial star or a disjoint union of nontrivial stars.

In the reverse direction, the complement is the disjoint union of nontrivial stars and so the original graph is robber-win. Now the addition of any edge in G results in the deletion of an edge in the complement $\overline{G}$. Notice that the deletion of an edge in $\overline{G}$ results in a leaf becoming an isolated vertex, which is a universal vertex in G with the addition of the edge. Therefore the addition of any edge results in a cop-win graph and so the original graph is edge maximally robber-win by Lemma 1.5. □

Beyond the fact that nontrivial stars are edge maximally robber-win in the complement, so are complete bipartite graphs in the complement.

Theorem 2.17. *[8, Lemma 4.27] Let G be a disconnected graph. The graph G is edge maximally robber-win if and only if $\overline{G}$ is a complete bipartite graph.*

Theorem 2.18. *[8, Lemma 4.36] The join of any two edge maximally robber-win graphs is edge maximally robber-win.*

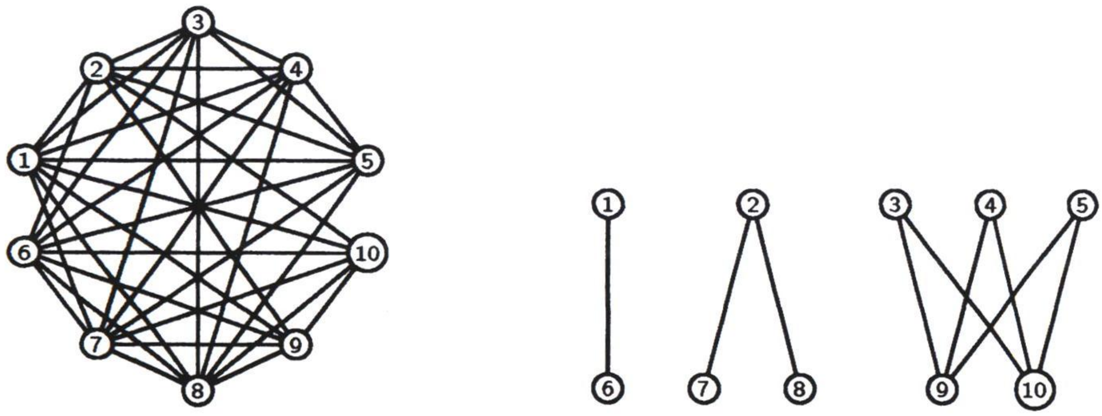
Using these two theorems we can get the following result.

Theorem 2.19. *If $\overline{G}$ is the disjoint union any number of complete bipartite graphs, then G is edge maximally robber-win.*

Proof. By Theorem 2.17 we know that the complement of a complete bipartite graph is edge maximally robber-win. The complement of the join of two graphs F and H is the disjoint union of $\overline{F}$ and $\overline{H}$. Therefore by Theorem 2.18 it follows that the disjoint union of any number of complete bipartite graphs in the complement is edge maximally robber-win. □

The results of Theorem 2.19 were demonstrated in Example 2.10. The following is another example of Theorem 2.19.

Figure 2.20. Consider the graph G and its complement $\overline{G}$ below.



G $\overline{G}$

The graph G is edge maximally robber-win by Theorem 2.19 since its complement is a disjoint union of complete bipartite graphs.

The above theorems and examples give several families of edge maximally robber-win graphs. The next section describes some results that hold for all edge maximally robber-win graphs.

3 The Ugly: Edge Maximally Robber-Win Graphs, The Great Escape

In the previous section we captured many edge maximally robber-win graphs. In fact, the conditions found in Theorems 2.11, 2.15, 2.19, and 2.18 seem broad enough to suspect that all of the edge maximally robber-win graphs have been found. It turns out that there are many more edge maximally robber-win graphs, as shown by Hill [8] and Fitzpatrick [6, Theorem 7]. In this section, we use a characterization of retractions viewed in the complement of a graph to find more information about edge maximally robber-win graphs.

Theorem 3.1. *If G is a graph and u is a leaf in $\overline{G}$, then any vertex that is distance 2 from u in $\overline{G}$ will retract to u.*

Proof. Let G be a graph such that $\overline{G}$ contains a leaf, call it u. Suppose u is distance 2 away from vertex v in $\overline{G}$. Both $\overline{N[u]}$ and $\overline{N[v]}$ share a vertex, call it w. Since u is a leaf in $\overline{G}$, w is the only vertex in $\overline{N}[u]$. Thus, $\overline{N(u)} \subseteq \overline{N(v)}$ and so v retracts onto u. □

If u is a leaf in $\overline{G}$, then the retraction of vertex v onto u is called a **pruning**.

Notice that the retractions in the complement of the graphs in Examples 2.10 and 2.20 were only on vertices of distance 2 or ∞ away from each other. In Corollary 2.8, the complement of substituting K_2 graphs also resulted in dominating and corner vertices distance 2 away from each other. As the next theorem shows, there are no other lengths possible.

Lemma 3.2. *If vertex u retracts onto vertex v in G, then the distance from u to v in $\overline{G}$ must be either 2 or ∞. Furthermore if the distance is ∞ then v is a universal vertex in G.*

Proof. If vertex u retracts onto vertex v, then the neighborhood of one vertex must be contained in the neighborhood of the other in G. If the two vertices are in the same connected component in $\overline{G}$, then since one neighborhood is contained in the other in G, they must share at least one vertex in both neighborhoods or else the retraction would not be possible. If the two vertices are adjacent in $\overline{G}$ then retraction is not possible, since they are not adjacent in the original graph G. This means that the distance between two retracting vertices in the same connected component in $\overline{G}$ must be 2.

If u and v are not in the same connected component in $\overline{G}$, then they do not share any neighbors in $\overline{G}$. Now since u and v share no neighbors in $\overline{G}$, the only way that vertex u could retract onto vertex v is if $\overline{N[v]} \subseteq \overline{N[u]}$, meaning vertex v has no neighbors in $\overline{G}$, i.e. v is a universal vertex in G. Therefore the distance between u and v in $\overline{G}$ is ∞ and v is a universal vertex in G. □

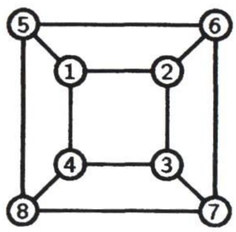
Theorem 3.1 and Lemma 3.2 give a better picture of what kind of retractions are possible in the complement of a graph. In fact, all retractions can be narrowed down to three types.

Theorem 3.3. *The only three types of retractions possible in $\overline{G}$ are pruning retractions, retractions involving 4-cycles, and retractions involving an isolated vertex.*

Proof. From Lemma 3.2, we know the distance between two vertices that retract in $\overline{G}$ must be either 2 or ∞. Furthermore, if the distance is ∞ then by Lemma 3.2 one of the vertices must be a universal vertex in G, which is an isolated vertex in $\overline{G}$.

Now we must consider what kinds of retractions happen if two vertices are a distance of two from each other in $\overline{G}$. Suppose one of the two vertices involved in the retraction is a leaf in $\overline{G}$. Then by Theorem 3.1 we know that pruning is possible. So now consider if neither vertex involved in the retraction is a leaf (meaning that each vertex has at least two neighbors in $\overline{G}$). This means that each vertex must have at least two neighbors and because they retract, at least one vertex's entire neighborhood in $\overline{G}$ is contained in the other vertex's neighborhood. This means they must share at least two neighbors in $\overline{G}$ and they are not connected in $\overline{G}$, which implies that they form a 4-cycle in $\overline{G}$. □

Example 3.4. Not every two vertices in the complement of an edge maximally robber-win graph that are at distance two and are in a 4-cycle in the complement will retract. This is possible if each of these two vertices have a neighbor in the complement that is not in the neighborhood of the other vertex. One such example of this is the complement of a cube graph.



$\overline{G}$

Each vertex in $\overline{G}$ is in a 4-cycle and none of the 4-cycles retract.

Lemma 3.5. *If the complement of the graph G is a path P_r, then G is cop-win if and only if r is congruent to 0 modulo 3.*

Proof. Let G be a cop-win graph such that $\overline{G}$ is a path P_r. Note that either $r = 0$, i.e the complement is a single vertex which is cop-win, or $r \geq 3$. When $r = 1$ or $r = 2$, $\overline{G}$ is a nontrivial star which is robber-win by Corollary 2.19. If $r \geq 3$, a leaf will prune a vertex two away as in Theorem 3.1, and the pruned vertex's connected edges are deleted. This results in a $K_{1,1}$ star and a path of length $r - 3$. We can continue pruning the path until we are left with a path of length 0, 1, or 2 and a disjoint union of $K_{1,1}$ stars. If r is congruent to 0 modulo 3, then pruning the complement will leave a path of length 0 and a disjoint union of $K_{1,1}$ stars, which has a universal vertex in G after the sequence of retractions which is cop-win by Lemma 1.5. If r is congruent to either 1 or 2 modulo 3, then pruning leaves a disjoint union of nontrivial stars, which is robber-win by Corollary 2.19. Therefore G is cop-win and $\overline{G}$ is a path P_r, if and only if r is congruent to 0 modulo 3.

□

Theorem 3.6. *[8, Lemma 4.41] If $\overline{G}$ is connected and unicyclic, then G is edge maximally robber-win if and only if $\overline{G}$ is a cycle of length congruent to* $1 \mod 3$.

Theorem 3.7. *Let G be an edge maximally robber-win graph. The complement of the graph cannot contain a non-branching path of length congruent to 1 modulo 3 between two vertices of degree greater than 2, where all four cycles in the complement are vertex disjoint from this path.*

Proof. Suppose G is an edge maximally robber-win graph. Without loss of generality we need only consider the subgraph H of G where all retractions have been made. Suppose there exists a non-branching path in $\overline{H}$ of length $1 \mod 3$ between two vertices u, v of degree 3 or more.

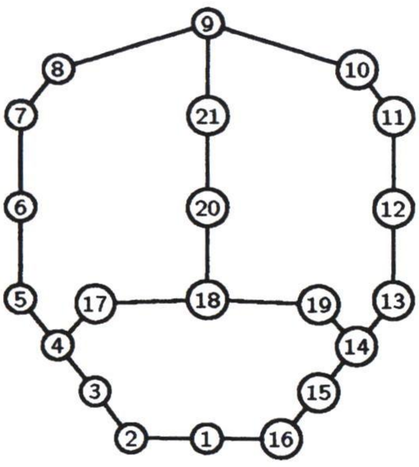
Let x be the vertex adjacent to u on the path between u and v. Delete the edge ux in $\overline{H}$ and note this is equivalent to adding an edge in H. This will either result in x being a leaf in $\overline{H} - \{ux\}$ or there will be no retractions possible, since the edge is not part of a 4-cycle in $\overline{H}$. If it gives us a leaf, x, then the length of the path from x to v is $0 \mod 3$. We can now prune what remains of this path to a union of disjoint edges and a separate connected component containing v that does not allow any more retractions. Since u and v both have degree at least 3, after these prunings each will be contained in a component that contains at least three vertices and no retractions, contradicting the fact that H is edge maximally robber-win and consequently G is edge maximally robber-win. Therefore an edge maximally robber-win graph cannot contain a non-branching path of length congruent to 1 modulo 3 between two vertices of degree greater than 2 vertex disjoint from all four cycles in $\overline{G}$. □

Corollary 3.8. *Let G be an edge maximally robber-win graph. If $\overline{G}$ contains a non-branching path between two vertices u and v of degree greater than 2 which is not vertex disjoint from a 4-cycle that allows a retraction, then the complement cannot contain a non-branching path of length congruent to 1 modulo 3 between u and v.*

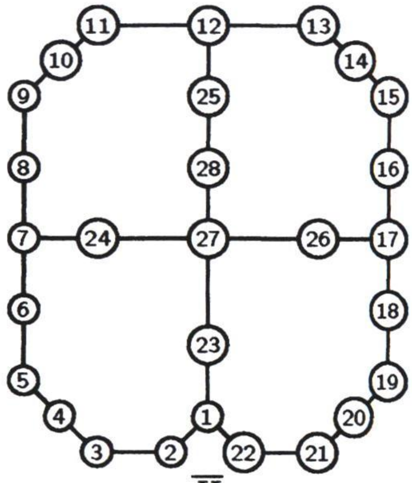
Proof. Notice that once we perform all of the retractions to obtain H these two vertices will still have degree at least 3 and so we apply Theorem 3.7 to H. □

If the edge maximally robber-win graph contains a non-retracting 4-cycle that contains either vertex, then after the deletion of the edge in the proof of Theorem 3.7 it is possible that we introduce a new 4-cycle retraction that may still result in a cop-win graph.

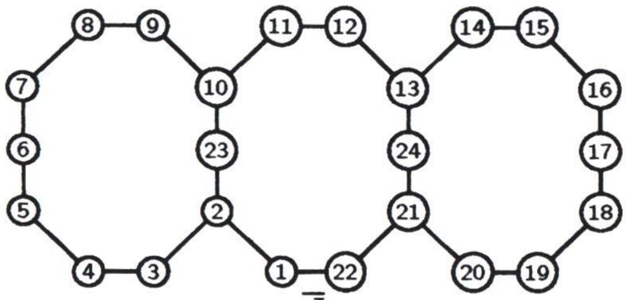
Example 3.9. The graphs below—labeled $\overline{G}$, $\overline{H}$, and $\overline{J}$—are complements of edge maximally robber-win graphs which can be verified by utilizing SageMath version 8.9 [11], with code provided in Appendix A.



$\overline{G}$



$\overline{H}$



$\overline{J}$

The sage code works as follows:

The code requires the user to input the list of edges in the complement of the graph in question into the function *Is_mrw()*. Then using this list, we create an adjacency matrix of the complement graph to verify whether or not the neighborhoods of two vertices match as well as to determine if two vertices are on opposite sides of a $4-cycle$. We also count the number of edges in the complement to help verify that every edge has been deleted to verify whether the graph is edge maximally robber-win. We then define a couple of functions that will be called several times in the code. The *retract()* function simply deletes the row and column entries of that vertex in the adjacency matrix which is exactly what happens when a vertex is retracted. The *rwCheck()* function determines whether or not the graph entered into it is cop-win or robber-win by looking for the following in the complement of the graph: isolated vertices, leaves, and retractable 4-*cycles*.

1. **Isolated Vertex:** The function first determines whether or not the complement has an isolated vertex by checking for a row with row sum 0 in the adjacency matrix. If it does, then since every vertex can retract onto it by Lemma 3.2 it must be a cop-win graph.

2. **Leaves:** If an isolated vertex was not found, the code then looks for any leaves by checking for a row with row sum 1 in the adjacency matrix. If it finds a leaf, it will determine whether it is a K_2 edge. If it is a K_2 edge, to help with efficiency, the K_2 edge is deleted since it is clear that those vertices will no longer retract onto anything other than an isolated vertex. However it will not delete the K_2 edge if they are the only two vertices left. In this case the graph will be deemed robber-win. If the leaf is not in a K_2 edge, then it will perform a pruning which will retract a vertex that is distance 2 away from the leaf.

3. **4-cycles:** If neither an isolated vertex nor a leaf was found, then the code will look for a 4-*cycle*. The code does this by looking at the square of the adjacency matrix of a graph. If there are any entries in the square of the adjacency matrix at least 2 or greater (not including entries on the diagonal), then the two vertices that make up that entry are on opposite sides of a 4-*cycle*. Then to determine whether or not those vertices retract onto one another, we look back at the adjacency matrix. We check whether they are neighbors (we cannot retract them if they are neighbors) and if not then focus on vertices i and j on opposite sides of a 4-*cycle*. We first subtract the jth row of the adjacency matrix from the ith row. This will create a row of entries that are a mix of $-1's$, $0's$, or $1's$. An entry is -1 if vertex j has a

neighbor that is not a neighbor of i, 1 if vertex i has a neighbor that is not a neighbor of j, or 0 if both vertices i and j either are neighbors or are not neighbors of the vertex. After subtracting the two rows we use the special case of equality in the triangle inequality on the list of the subtracted entries to verify whether or not a retraction is possible. In other words, we are comparing whether or not the absolute value of the sum of these entries is equal to the sum of the absolute value of these entries.

$$\left|\sum(\text{row } i - \text{ row } j)\right| = \sum |(\text{row } i - \text{ row } j)|$$

If they are equal then by the triangle inequality the two vectors are equal. If all of the differences in each column are $0's$, then both vertices have the same neighborhood and can retract onto each other. Otherwise, one vertex has a larger neighborhood in the complement than the other. In the case where all of the differences are non-negative or all $0's$, the code retracts vertex i onto vertex j. If the differences are $-1's$ and $0's$, it will retract vertex j onto i.

The *rwCheck()* will go through the above steps one at a time in the order listed. If any retraction or K_2 deletion was made, then it restarts back to the first step since a retraction could create other retractions. The function only terminates if an isolated vertex was found (meaning that it is cop-win), or the code checked every vertex and could not find any leaves or retractable 4-cycles (meaning that the graph is robber-win).

The last function the code defines will delete an edge from the graph. This is used to verify whether or not deleting any edge in the robber-win graph will create a cop-win graph.

The code runs *rwCheck()* on the original graph. If it finds that the graph is cop-win it returns "Not Maximally Robber-Win". Otherwise it runs *rwCheck()* on each possible subgraph of the form $\overline{G}$ with edge e removed for each edge e in $\overline{G}$. If *rwCheck()* gives cop-win for every edge deleted it will output "Maximally Robber-Win!!!", if it ever gives robber-win, then the code will output "Not Maximally Robber-Win."

This concludes our results on edge maximally robber-win graphs as we now bring our attention to the last type of edge turncoat graph.

4 Wanted: Rounding up Edge Minimally Robber-Win Graphs

We now turn to the question of deleting edges in a robber-win graph.

It is clear that no edge minimally robber-win graph can have a cut edge (i.e. if the deletion of an edge disconnects the graph, then the resulting disconnected graph is also robber-win). This was shown by Alan Hill in [8] where he does a considerable amount work investigating edge minimally robber-win graphs. Examples of edge minimally robber-win are given below.

Example 4.1. [8, Lemma 4.16] The cycle C_k is edge minimally robber-win if $k \geq 4$. The graph C_k is robber-win if $k \geq 4$ since the robber can maintain a distance of 1 or more edges away from the cop. Now note that the deletion of any edge in a cycle results in a path, P_{k-1} and a path is cop-win. Therefore the cycle C_k is edge minimally robber-win for $k \geq 4$.

Example 4.2. Any $n-2$ regular graph on n vertices is edge minimally robber-win for any even positive integer n. By Theorem 2.3 we see that the graph is robber-win. The complement of the graph contains $n/2$ disjoint edges. Now the deletion of an edge in the original graph corresponds to the addition of an edge in the complement. The addition of any edge that is missing from this graph will result in a path of length 3 with $n/2-2$ disjoint edges. We can perform a pruning retraction on the resulting complement and this will give an isolated vertex in the complement. This corresponds to a universal vertex in the retraction of the original graph and so the deletion of any edge in the original graph results in a cop-win graph. Therefore any $n-2$ regular graph is edge minimally robber-win.

It is interesting to note that Example 4.2 and Theorem 2.3 show that $n-2$ (n is an even, positive integer) regular graphs are both edge minimally robber-win and edge maximally robber-win. With these theorems in mind one might say $n-2$ regular graphs barely escape being robber-win. However [8] shows us that not all edge minimally robber-win graphs are regular.

Theorem 4.3 ([8]). *The join of any $n-2$ regular graph and any cycle C_k is edge minimally robber-win for even, positive integer n and an integer $k \geq 4$.*

We show that for any edge minimally robber-win graph G the diameter of $\overline{G}$ is either 2 or 3. In fact we show that if the diameter of $\overline{G}$ is 2, then $\overline{G}$ is 2-self-centered.

Theorem 4.4. *Let G be edge minimally robber-win and assume $\overline{G}$ has no isolated edges. Then $\overline{G}$ must be connected.*

Proof. Let G be edge minimally robber-win and assume $\overline{G}$ has no isolated edges. This means that $\overline{G}$ does not have any K_2 components. Also, since G has no corners by [8, Lemma 4.11] each connected component in $\overline{G}$ cannot have any vertices of degree 1 (i.e. it cannot have any leaves), otherwise pruning would be possible. Furthermore $\overline{G}$ does not have an isolated vertex.

Suppose we delete an edge in G, this introduces at least one retraction since G is edge minimally robber-win. In the complement, this means that the addition of any edge to $\overline{G}$ results in at least one retraction involving a 4-cycle. Note that it is impossible to create an isolated vertex or a leaf in $\overline{G}$ by adding an edge.

Suppose $\overline{G}$ has at least two connected components. If we add an edge uv between the two separate components in $\overline{G}$, then the resulting graph will not have an additional 4-cycle. This implies that u (and likewise v) cannot retract onto a vertex from another component. Therefore either u or v must retract onto another vertex in their respective component in $\overline{G}$. Without loss of generality, suppose u retracts onto another vertex x in its component after the addition of the edge uv. Since G is edge minimally robber-win, u could not have retracted onto x without the addition of uv. This is only possible if x was already a neighbor of v in the complement since we must have $\overline{N[x]} \subseteq \overline{N[u]}$ in order for vertex u to retract onto x. This contradicts the fact that u and v are in separate components. Therefore $\overline{G}$ must be connected. □

Lemma 4.5. *Let G be a graph with $\overline{G} = C_k$. The graph G is edge minimally robber-win if and only if $k = 5$.*

Proof. Let G be edge minimally robber-win, assume that $\overline{G}$ is a cycle. If $\overline{G}$ was either C_3 or C_4, then G would be disconnected which implies that G would not be edge minimally robber-win.

If $\overline{G}$ is C_5 then G is also C_5. By Example 4.1 C_5 is edge minimally robber-win.

If $\overline{G}$ is C_k for some $k \geq 6$, suppose we add an edge in $\overline{G}$ that creates a 3-cycle and a $(k-1)$-cycle but since $k \geq 6$, this is not a 4-cycle. Since a new 4-cycle is not created this prevents the addition of any new retractions.

Therefore if $\overline{G} = C_k$, then G is edge minimally robber-win if and only if $k = 5$. □

Lemma 4.6. *Let G be edge minimally robber-win and assume that $\overline{G}$ has no isolated edges. Then the degree of each vertex in $\overline{G}$ with n vertices is at least 2. Equivalently, the maximum degree of a vertex in G is $n-3$.*

Proof. Let G be edge minimally robber-win and assume that $\overline{G}$ has no isolated edges. We know $\overline{G}$ must be connected from Lemma 4.4. By [8, Lemma 4.11] G does not have any corners (which implies no dominating vertex) and so $\overline{G}$ does not have any leaves or isolated vertices. Therefore, each vertex in $\overline{G}$ must have degree at least 2. Equivalently the maximum degree of a vertex in G is $n-3$. □

Theorem 4.7. *Let G be edge minimally robber-win where $\overline{G}$ contains no isolated edges. Then the diameter of $\overline{G}$ is at most 3.*

Proof. Let G be edge minimally robber-win where $\overline{G}$ contains no isolated edges and assume that the diameter of G is $k \geq 4$. Suppose u and v are vertices in G such that the shortest distance between u and v in $\overline{G}$ is k. Since G is edge minimally robber-win, adding any edge in $\overline{G}$ will make a retraction possible. However, if an edge was added to connect vertex u and v we have a new cycle of length $k+1 \geq 5$. This means a new 4-cycle was not introduced. Therefore the addition of the edge in $\overline{G}$ must make it possible for either u or v to retract onto another vertex. Without loss of generality, suppose vertex u retracts onto another vertex x in $\overline{G}$. By Lemma 3.2 it follows that the distance from u to x is 2. However, since u does not retract onto vertex x without the addition of the edge uv it follows that v must be a neighbor of x in $\overline{G}$. Now since the distance from u to x is 2 in $\overline{G}$, and v is adjacent to x in $\overline{G}$, there is a path from u to v in $\overline{G}$ which has length 3. This contradicts the fact that the length of the shortest path from u to v in $\overline{G}$ is at least 4. Therefore, the diameter of $\overline{G}$ is at most 3. □

Theorem 4.8. *Let G be edge minimally robber-win where $\overline{G}$ contains no isolated edges. If the diameter of $\overline{G}$ is 2, then the radius of $\overline{G}$ must also be 2. In other words, $\overline{G}$ is 2-self-centered.*

Proof. Let G be edge minimally robber-win where $\overline{G}$ contains no isolated edge and assume that $\overline{G}$ has diameter 2. By Theorem 4.4 it follows that $\overline{G}$ is connected and so if $\overline{G}$ has radius 1, then there exists a vertex u such that u is adjacent to all of the other vertices in $\overline{G}$. This means that u is isolated in G, contradicting the fact G is edge minimally robber-win. Therefore, the radius of $\overline{G}$ must be 2. Furthermore, it also follows that the eccentricity of each vertex is 2, since the diameter of $\overline{G}$ is 2, and so $\overline{G}$ is self-centered. Therefore $\overline{G}$ is 2-self-centered. □

There is a characterization of 2-self-centered graphs in [10] which has not yet yielded any counter-examples to Conjecture 4.14 in [8].

5 CONCLUSION: Wanted Dead or Alive

The research of this paper offers a few avenues of further investigation into cops and robbers. It appears that edge maximally robber-win graphs and edge minimally robber-win graphs are difficult to characterize and both merit more research. In particular the following questions should lead to further research.

- What is the characterization of edge maximally robber-win graphs? What is the characterization of edge minimally robber-win graphs?
- In [8] the author investigates several other operations on graphs. Can our method of looking at the complement help characterize different kinds of turncoat graphs?
- We show that the diameter of an edge minimally robber-win graph is at most three in Theorem 4.7. Is it possible to show that the diameter is 2?
- Can we use the diameter to prove or find a counterexample to Conjecture 4.14 from [8]?
- Does the characterization of 2-self-centered graphs in [10] lead to counterexamples of any conjectures in [8]?

Acknowledgements

The authors would like to thank Shannon Fitzpatrick for helping us find previous work on this topic. The authors would also like to thank the anonymous referee for the great comments that helped improve the paper.

References

[1] A. Bonato, G. Kemkes, and P. Prałat. Almost all cop-win graphs contain a universal vertex. *Discrete Mathematics*, 312(10):1652 – 1657, 2012.

[2] A. Bonato and R. Nowakowski. *The Game of Cops and Robbers on Graphs.* Student mathematical library. American Mathematical Society, 2011.

[3] B. Cameron and S. Fitzpatrick. Edge contraction and cop-win critical graphs. *Australasian Journal of Combinatorics*, 63(1):70–87, 2015.

[4] G. Chartrand, L. Lesniak, and P. Zhang. *Graphs & Digraphs.* Textbooks in Mathematics. CRC Press, 2015.

[5] N. Clarke, S. Fitzpatrick, A. Hill, and R. Nowakowski. Edge critical cops and robber. *Discrete Mathematics*, 310(17):2299 – 2309, 2010.

[6] S. L. Fitzpatrick. Edge-critical cops and robber in planar graphs. *Discrete Mathematics*, 329:1 – 11, 2014.

[7] C. Godsil and G. Royle. *Algebraic Graph Theory.* Graduate Texts in Mathematics. Springer New York, 2013.

[8] A. Hill. *Cops and robbers: Theme and variations.* PhD thesis, Dalhousie University, 2008.

[9] L. Lovász. Normal hypergraphs and the perfect graph conjecture. *Discrete Mathematics*, 306(10):867 – 875, 2006. 35th Special Anniversary Issue.

[10] M. Shekarriz and M. Mirzavaziri. A characterization for 2-self-centered graphs. *Discussiones Mathematicae Graph Theory*, 38:27– 37, 2018.

[11] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.9)*, 2019.

Appendix A

The following code can be implemented in Sage version 8.9 [11] in order to check whether the complement of a graph is the complement of an edge maximally robber-win graph.

```
import numpy as np
#The function Is_mrw(L) takes a list of edges of the complement of a graph as input
def Is_mrw(L):
    G = Graph(L)
    H = G.adjacency_matrix()
    G.show()
    M = H.numpy()
    #define Matrix
    adjMatrixMC = M
    edgecount = int(np.sum(adjMatrixMC)/2)
    results1 = 0

    def retract(vertex,adjmat):
        #delete row
        adjmat = np.delete(adjmat,vertex,axis = 0)
        #delete column
        adjmat = np.delete(adjmat,vertex,axis = 1)
        return adjmat

    def rwCheck(adjMatrix):
        #define List of Retractions
        retractList = []
        vertexNames = []
        adj2Matrix = []
        copwin = 0
        robberwin = 0
        for i in range(0,len(adjMatrix)):
            vertexNames.append(i)
        while (copwin == 0 and robberwin == 0):
            rows = len(adjMatrix)
            leaf = 0
            #check for isolated vertex
            for i in range(0, rows):
                if adjMatrix[i].sum() == 0:
                    copwin = 1
                    break
```

```
        #check for leaves (pruning)
        if (copwin == 0 and robberwin == 0):
            adj2Matrix = adjMatrix.dot(adjMatrix)
            for i in range(0, rows):
                find = False
                if adjMatrix[i].sum() == 1:
                    for j in range(0, rows):
                        #The following removes K_2 edges
                        if (adjMatrix[i,j] == 1 and adjMatrix[j].sum() == 1 and
                            len(adjMatrix)>2):
                            #deletes vertex i
                            adjMatrix = retract(max([i,j]), adjMatrix)
                            #deletes vertex j
                            adjMatrix = retract(min([i,j]), adjMatrix)
                            leaf = 2
                            find = True
                            break
                        elif adj2Matrix[i,j] == 1 and i != j:
                            adjMatrix = retract(j,adjMatrix)
                            leaf = 1
                            find = True
                            break
                    if find:
                        break
        #check for 4-cycles
        if (leaf == 0 and (copwin == 0 and robberwin == 0)):
            for i in range(0, rows):
                find = False
                for j in range(0, rows):
                    if (i == rows-1 and j == rows-1):
                        robberwin = 1
                    #If it finds a 4 cycle then it looks at the neighborhoods
                    elif (adj2Matrix[i][j] >= 2 and adjMatrix[i][j] == 0 and i > j):
                        if np.absolute(np.sum(adjMatrix[i]-adjMatrix[j])) == np.sum(np.absolute(adjMatrix[i]-adjMatrix[j])):
                            if np.sum(adjMatrix[i]-adjMatrix[j]) >= 0:
```

```
                                adjMatrix = retract(i,adjMatrix)
                            else:
                                adjMatrix = retract(j,adjMatrix)
                            find =  True
                            break
                if find:
                    break
    return copwin

#checks for original copwin/robber win
results1 = rwCheck(adjMatrixMC)
def deleteEdge(matrx,c,d):
    deletedEdgeMatrix = matrx
    deletedEdgeMatrix[c][d] = 0
    deletedEdgeMatrix[d][c] = 0
    return deletedEdgeMatrix

#this will count how many times an edge was deleted below
countMe = 0
delAdjMatrix = [0,0]
if results1 == 0:
    for a in range(0, len(adjMatrixMC)):
        findbreak = False
        for b in range(0, len(adjMatrixMC)):
            if (a > b and adjMatrixMC[a][b] == 1):
                countMe = countMe + 1
                #only need to check half of adjacency matrix by deleting an edge
                delAdjMatrix = deleteEdge(adjMatrixMC,a,b)
                results1 = rwCheck(delAdjMatrix)
                #reset delAdjMatrix to = adjMatrix
                delAdjMatrix[a][b] = 1
                delAdjMatrix[b][a] = 1
                if results1 == 0:
                    findbreak = True
                    break
        if findbreak:
            break
if (countMe == edgecount and results1 == 1):
    print("Maximally Robber Win!!!!")
else:
    print("Not Maximally Robber-Win")
```