A Game of Cops and Robbers in Some Networks

An MTech dissertation by
Sheikh Shakil Akhtar
CS1836

Under the supervision of
Dr. Sandip Das
ACMU, ISI, Kolkata

This project is submitted as a dissertation for the requirements of the Master of Technology programme in Computer Science at Indian Statistical Institute on Friday 10th July, 2020

Preface

Cops and Robber games are pursuit-evasion games played on connected graphs. They have been studied extensively and finding the cop number of certain classes of graphs has been one of the major problems in these games. Networks, on the other hand, have been some of the most interesting graph classes. In this work, we will investigate the cop number problem of two types of networks, whose cop numbers were not known in general, namely, *butterfly networks* and *solid grid graphs*.

Acknowledgement

I would like to thank my dissertation supervisor, Dr. Sandip Das, Professor, Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, for agreeing to guide me and helping me with interesting problems. Without his continuous guide and support, this would not have been possible. I am also very thankful to Mr. Harmender Gahlawat, Senior Research Fellow, ACMU, ISI, Kolkata. He helped me throughout my project with his valuable suggestions and thus I was able to give this work a final form.

Sheikh Shakil Akhtar
MTech(CS),2nd year
Roll No.: CS1836
ISI,Kolkata
Friday 10th July, 2020

1 Cops and Robber Games: An Introduction

Cops and Robber is a pursuit-evasion game played on finite connected graphs with the following rules:

- It is a two-player game, where one player controls k cops, where $k \geq 1$, and the other controls the robber
- The game begins with the k cops occupying a vertex or some vertices and then the robber occupies a vertex
- The cops and robber makes alternating moves
- On the cops' turn, any number of the k cops can move to their adjacent vertices or stay put on their current vertex
- The same goes for the robber as well. On its turn, it can move to any of the adjacent vertices of its current vertex or stay put
- This is a perfect information game,i.e., all the *agents*, (an agent is either a cop or the robber) can see the entire graph as well as other *agents* and their moves
- If at least one of the cops succeed in occupying the same vertex as the robber, we call it a *capture*
- The cops win if they capture the robber within finite time, otherwise, the robber wins

This is the classical version of the *Cops and Robber* game which was introduced in [5].

The ***cop number of a graph*** G, denoted by $c(G)$, is the minimum number of cops required to capture the robber, irrespective of the sequence of moves of the robber. The cop number is finite as the number of vertices of a graph is an obvious upper bound. If a graph G has $c(G) = 1$, then we say that G is *copwin* graph.

2 Some Known Results

We have the following known theorems for cop numbers([1]):

Theorem 1. *The cop number of a tree is 1.*

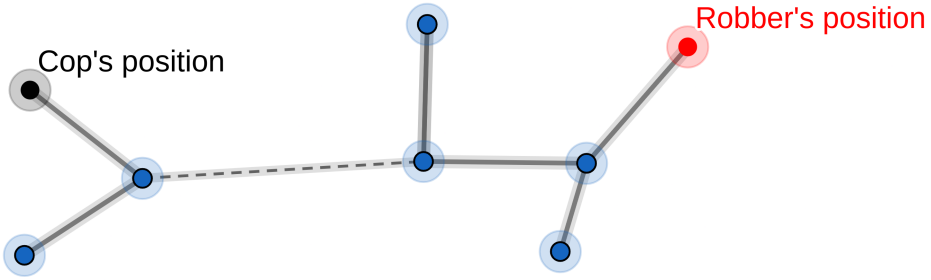

Figure 1: Chasing the robber in tree

Proof. The cop can start anywhere, and at each step move to the unique neighbour that is closer to the robber. The uniqueness is ensured by the acyclic nature of tree since between any two vertices there is a single path in the graph as shown in figure 1. Each of the cop's steps reduces the size of the subtree that the robber is confined to, so the game eventually ends with the cop capturing the robber. □

Theorem 2. *The cop number of a cycle of length 3 is one.*

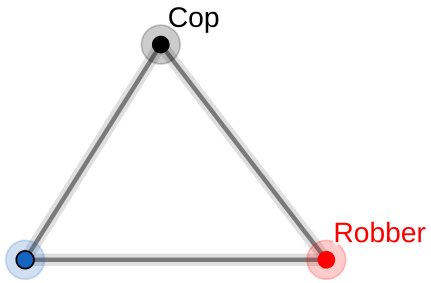

Figure 2: Chasing the robber in 3-cycle

Proof. Let G be a $3-cycle$. Let a cop C start at any of the vertices. We see that in figure 2 no matter where the robber starts it will either be in the same vertex as that of the cop or in any of its two neighbours. In the former

case, the robber gets captured right away whereas in the latter case the cop can capture the robber in its next move. □

Theorem 3. *The cop number of a cycle of length more than three is 2.*

Proof. Let G be a cycle with length greater than 3. Let a cop C start at any vertex, say v, in G. Let the robber R start at a vertex w, such that the distance between v and w is at least 2, which is possible since the length of the cycle G is at least 2. At each turn of C, it can move only one vertex closer to R thus decreasing the distance by 1. However, R can simply move one vertex further away from C, thus evading the effect of C's last move and bringing the distance back to what it was before that. This way R can ensure that the distance between itself and C never decreases and thus it never gets captured.

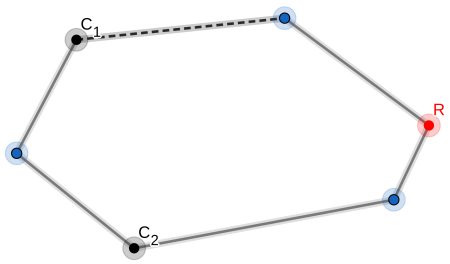

Figure 3: Chasing the robber in a cycle of length greater than 3 using two cops

Let there be two cops C_1 and C_2. Let them both start at a vertex say v, while the robber R starts at some vertex w. If, $v = w$, then we are done. If not, then the cops can form a strategy to capture R. The cop C_1 will keep chasing R to capture it by moving to its neighbour in the shortest path between its vertex and that of the R, while C_2 stays put in v. The robber, can either stay put or move away from C_1. At each turn it stays put the cop C_1 can decrease the distance by moving closer to it. At each turn, it moves away from C_1 it comes closer to C_2. Thus, after a finite number of moves, it will either be a neighbour of C_1 or C_2 or both, which will ensure its capture by a cop. □

3 Butterfly Networks

Butterfly Networks are a special class of networks that are defined inductively for every non-negative integer([4]). We denote the *nth* Butterfly Network as B_n and say that it is the *nth level* or $n - dimensional$ Butterfly Network. The definition is as follows:

- At level 0, we have B_0 which is a graph with single vertex and no self-loops. We also say that the *set of external vertices* of B_0, denoted by $Ext(B_0)$ is that very single vertex.

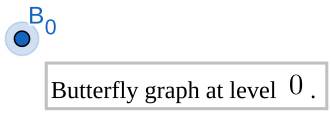

Figure 4: B_0

- The set V of nodes (vertices) of an n-level butterfly network ($n \geq 1$) correspond to pairs $[w, i]$, where i is the dimension or level of a node ($0 \leq i \leq n$), and w is an n-bit binary number that denotes the row of the node. Two nodes $[w, i]$ and $[w', i']$ are linked by an edge if and only if $i' = i + 1$ and either, (i) w and w' are identical, or (ii) w and w' differ in precisely the i^{th} bit. In figure 5 we have B_3.

The external vertices of B_n, $Ext(B_n)$ are precisely the vertices of level 0, the vertices $v = [w, 0]$. Since w is an n-bit binary number so it can have exactly 2^n many different values. Thus, $|Ext(B_n)| = 2^n$ and $|V(B_n)| = 2^n(n + 1)$, for all $n \geq 0$.

3.1 Construction of B_{n+1} from B_n

We can construct B_1 from B_0 as follows:

- We take two copies of B_0, labelling one of them as $[0, 1]$) and the other as $[1, 1]$ as shown in figure 6
- We then join a new vertex labelled as $[0, 0]$ to $[0, 1]$ and to $[1, 1]$ as shown in 7
- And at last we join another new vertex labelled as $[1, 0]$ to $[1, 1]$ and to $[0, 1]$

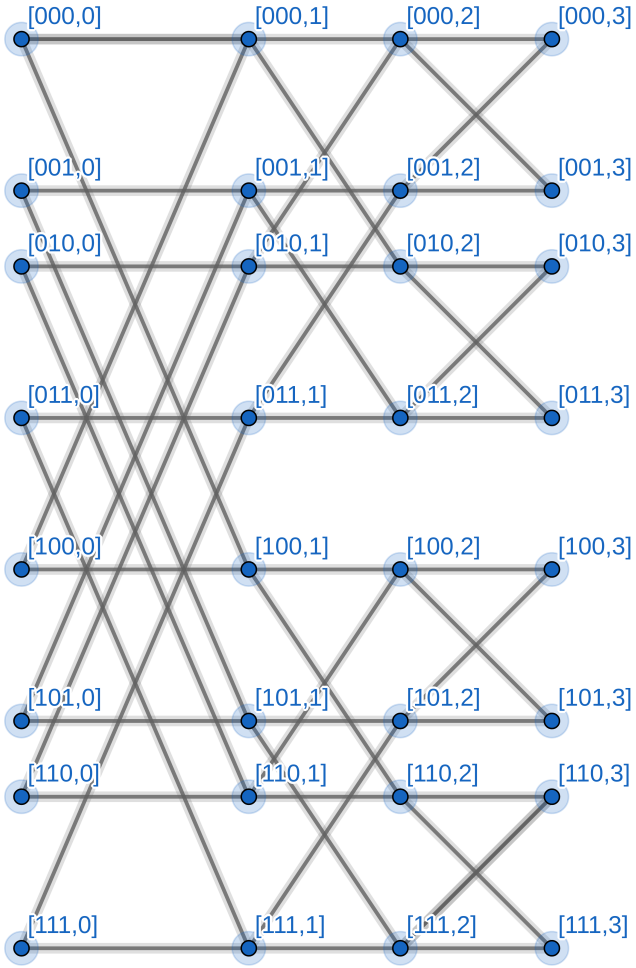

Figure 5: Butterfly network of $3rd$ level, B_3

Figure 6: Two copies of B_0

It is clear, the above graph is B_1. In general, we can construct B_{n+1} from B_n, where $n \geq 1$ as follows:

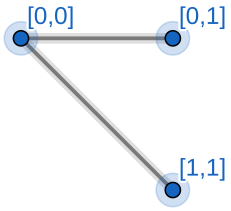

Figure 7: Adding a vertex $[0, 0]$

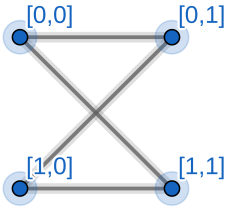

Figure 8: B_1 constructed from two copies of B_0

- We take two copies of B_n, say B' and B''

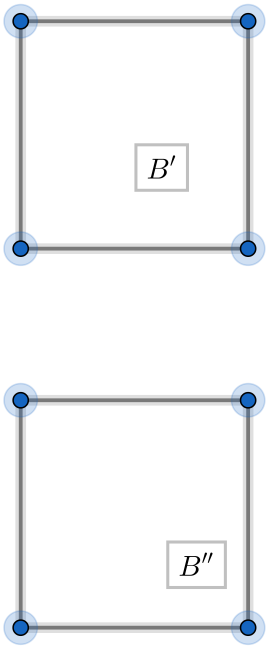

Figure 9: Two copies of B_n

- We increase the level number of each vertex in both B' and B'' by 1
- We add a 0 at the beginning of the row number of each vertex in B', making it $n+1$-bit
- We add a 1 at the beginning of the row number of each vertex in B'', making it $n+1$-bit
- We bring in 2^{n+1} new vertices labelled $[w, i]$ with $i = 0$ and w being an $(n+1)$-bit binary number ranging from 0 to $2^{n+1} - 1$.
- We join a new vertex $[w, i]$ to another vertex $[w', i']$ from either B' or B'' iff $i' = i + 1$ and either, (i) w and w' are identical, or (ii) w and w' differ in precisely the i bit.

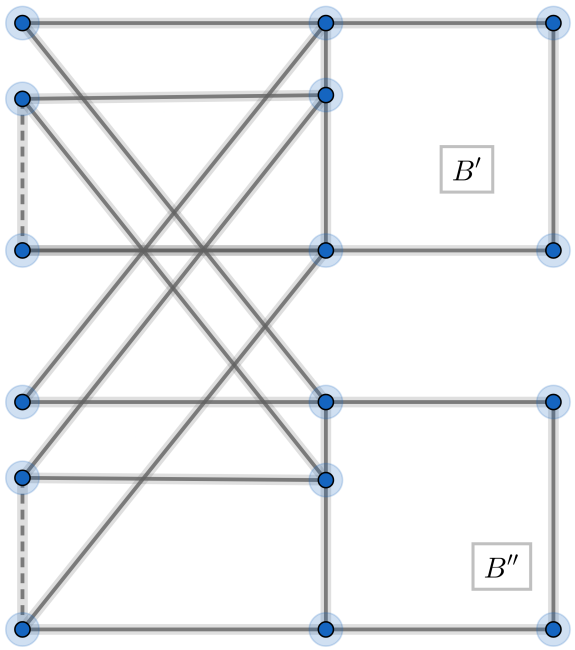

Figure 10: Construction of B_{n+1} from two copies of B_n

We claim that this new graph as shown in figure 10 is B_{n+1}. If we take two vertices $[w, i]$ and $[w', i']$, both from either B' or B'' then are joined by the rules of butterfly networks. If one is in B' and the other is in B'', then they will not have an edge between them by our very construction. And by the last step in our construction, we can see that the new vertices are joined to vertices in B' and B'' by the rules of butterfly network. Thus, the

above construction gives us a butterfly network. As, for the level, then we can see that the levels of nodes range from 0 to $n+1$ and the row numbers are $n+1$-bit binary numbers, so we have B_{n+1}. The external vertices of this graph are precisely the ones of level 0.

Thus, from the above construction, we see that any butterfly network B_n, $n \geq 1$ can be constructed from two copies of B_{n-1}. In fact, for any $n \geq 1$, we can construct B_n straight from B_0. We have the following theorem:

Theorem 4. *For any butterfly network B_n, the copies of B_k, with $0 \leq k \leq n$, within B_n is 2^{n-k}.*

Proof. We will prove this by induction on n.

Base Case: When $n = 0$. Thus, k can only have the value of 0. It is clear that B_0 will only have one copy of itself within itself.

Induction Hypothesis: Let our statement be true when $n = m$, for some $m \in \mathbb{N}$.

Inductive Step: We consider the case when $n = m + 1$. If $k = m + 1$, then as B_{m+1} will have only one copy of itself within itself, so the claim is true in this case.

If $0 \leq k \leq m$, then by Induction hypothesis each B_k will have 2^{m-k} many copies within B_m. As there are two copies of B_m within B_{m+1} by construction, so each of those B_k will have 2^{m+1-k} many copies in B_{m+1}, thus proving our statement. □

Corresponding vertices: As, B_n, $n \geq 1$ contains copies of $B_0, \ldots,$ B_{n-1}, the vertices which are not in $Ext(B_n)$, will have *corresponding copies*. So, if there is a vertex with label $[w, i]$, where $i \neq 0$ and $i \neq n$, then it does not belong to $Ext(B_n)$, and its corresponding copies are all those vertices $[w', i]$, which would have the exact same row number as itself in an isolated copy of B_{n-i}, that is, w and w' are identical after the first i bits. As for vertices of level n in B_n, then they are essentially the copies of B_0 in B_n and are thus corresponding to each other. We have the following theorem.

Theorem 5. *In B_n, the corresponding vertices in the sets of external vertices of the copies of B_{n-1} have the same neighbours in $Ext(B_n)$.*

Proof. Let two vertices $[w, 1]$ and $[w', 1]$ be corresponding vertices in the sets of external of the copies of B_{n-1}. So, by our construction, w and w' are identical except for the 0^{th} bit. Consider, $[w, 1]$. By our construction its neighbours in $Ext(B_{n+1}$ are exactly two, one of them being $[w, 0]$ and the other being $[w', 0]$. Similarly, the neighbours of $[w', 1]$ are exactly two in $Ext(B_{n+1}$, one of them being $[w, 0]$ and the other being $[w', 0]$. Thus proved. □

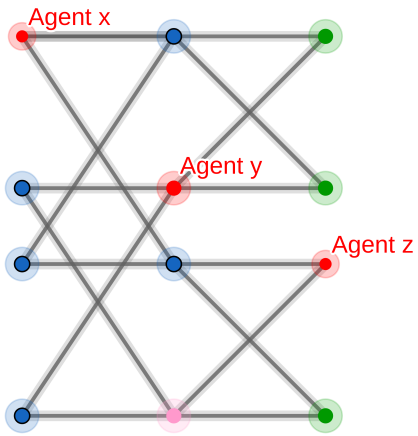

Figure 11: Agents and their images in B_2

3.2 Cops and Robber game in Butterfly Networks: (Our Main Contribution)

We have two cops C_1 and C_2, and a robber R. Let,

$v(C_i)$:= Current position of the cop C_i.

$v(R)$:= Current position of the robber R.

Observation: We observe that irrespective of the value of $v(C_i)$ or $v(R)$ in a graph B_n, $n \geq 0$ it is bound to lie in a vertex $v \in Ext(B_k)$, where $0 \leq k \leq n$, where $n - k$ is the level of the vertex. If $k = n$, it is in $Ext(B_n)$ or else in $Ext(B_k)$ for some $k < n$. This is because, we are constructing B_n incrementally from B_0, B_1,....,B_{n-1}. Thus, each vertex in B_n, lies in an external vertex of some B_i, where $0 \leq i \leq n$.

We use the above observation to define certain terms for our agents. (An agent is either a cop or a robber). For an agent x, $v(x)$ denotes its current position.

Container graph: Let $v(x)$ belong to $Ext(B_k)$, such that $0 \leq k \leq n$

Let's consider j, such that $0 \leq j < k$, then we notice that $v(x)$ does not belong to any vertex of any of the copies of B_j within B_n. Also, if $k < n$ and we consider j, such that $k < j \leq n$, then while $v(x)$ may or may not belong to a particular copy of B_j, it surely does not belong to the sets of external vertices of any of the copies of B_j. Thus, we see that $v(x)$ and the copy of B_k, such that $v(x) \in Ext(B_k)$ have a relationship. We call this particular copy of B_k to be the *container graph* of the agent x and k to be its *container level*. In particular, if $v(x) = [w, i]$, then $k = n - i$. Thus, in figure 11, the container level of agent x is 2, since $v(x) \in Ext(B_2)$, for agent y it is 1, since it belongs

to the external vertices of a copy of B_1 and similarly for agent z it is 0.

Image Positions: Let $v(x)$ have its container level k, such that $0 \leq k < n$.

We note that as $k < n$, then by theorem 4, B_k will have multiple copies within B_n. We denote the container graph of $v(x)$ to be B', which is a copy of B_k. As $v(x) \in Ext(B')$, then the other copies of $Ext(B_k)$ each have their vertices corresponding to $v(x)$. We define *set of image positions of x*, $I(x)$, to be precisely those vertices,i.e., $I(x) := \bigcup_{copies\ of\ B_k}\{v \in Ext(B_k) | v(x) \neq v,\ and\ v\ is\ corresponding\ to\ v(x)\}$. So in figure 11, $I(x) = \emptyset$, whereas $I(y)$ is a singleton set marked pink and $I(z)$ contains three vertices marked green. If a cop C_i manages to occupy vertex belonging to $I(R)$, then we say that *an image of R has been captured by* C_i.

Effective Positions: For an agent x, the *set of effective positions*, $Eff(x)$ is defined as,

$Eff(x) := \{v(x)\} \cup I(x)$.

Effective distance: Let C_i be a cop. The distance $d(C_i, R)$ is the actual distance between $v(C_i)$ and $v(R)$, defined in the usual way. We, define the *effective distance between* C_i *and* R, $d_E(C_i, R)$, as:

$d_E(C_i, R) := min\{d(v(C_i), w) | w \in Eff(R)\}$.

Thus, $d_E(C_i, R) \leq d(C_i, R)$. Also, note that $d_E(C_i, R) = 0$, iff $v(C_i)$ and $v(R)$ are corresponding to each other or $v(C_i) = v(R)$, whereas $d(C_i, R) = 0$, iff $v(C_i) = v(R)$. If, $d_E(C_i, R) = 0$, then we say that *an effective position of R has been captured by* C_i.

We have the following algorithm to ensure capture of effective position by one cop:

Theorem 6. *Algorithm 1 ensures that the cop C_i will capture one of the effective positions of the robber R, i.e., either R itself or one of its images.*

Proof. We notice that the algorithm moves the cop C_i closer to the closest effective position of the robber R. This results in the following:

- If R moves out of its current container graph to a bigger one then it increases the effective distance by 1. The cop on its turn nullifies it by moving closer to the robber's current effective position closest to itself.
- If R moves from its current container graph into something smaller, then it decreases the effective distance. The cop on its turn decreases the distance even further. If the effective distance reaches nil, then we are done.
- If R on its turn stays put, then too the cop decreases the effective distance by moving closer to the former's closest effective position to itself.

Algorithm 1 Capture Effective Position of Robber R

Require: Positions of the Cop C_i and the robber R and the graph under consideration, B_m

Ensure: C_i captures either R or one of its image positions

Initially cop C_i should be place in vertex $[w, m]$, where w is the m-bit binary representation of 0

while $d_E(C_i, R) \neq 0$ **do**

Move C_i to the closest effective position of R on its turn

Update the positions of all agents

end while

if $d(C_i, R) = 0$ **then**

Report the capture and terminate the game

end if

if $d_E(C_i, R) = 0$ but $d(C_i, R) \neq 0$ **then**

Report capture of image

end if

- As our graph is finite, so the robber can't increase the effective distance indefinitely and thus it will keep decreasing until it drops to nil.

□

Theorem 7. *By Algorithm 1, if R is not captured, then it can never happen that the dimension of $v(C_i)$ is less than that of $v(R)$, without R getting its image captured before that.*

Proof. Initially, we see that in the graph B_n, C_i begins at $[w, n]$, where w is the n-bit representation of 0. Thus, if R begins at any of the vertices of dimension n its image gets captured at the very beginning, since all vertices of dimension n are corresponding to each other. On, the other hand if R begins at a vertex of dimension less than n, and then it tries to move to vertices of higher dimensions, it is actually decreasing the value of $d_E(C_i, R)$. If, it stays put then to C_i will keep decreasing the value of $d_E(C_i, R)$ by theorem 6. Thus, before it can move to vertices of dimensions higher than that of $v(C_i)$, its image will get captured, if it does not get captured itself. □

Theorem 8. *If an image position of R has been captured by C_i in B_n by using Algorithm 1, then the cop can either capture R or keep its image captured.*

Proof. Lets say that an image position of R has been captured by C_i, such that $v(R) = [w, i]$ and $v(C_i) = [w', i]$. Then, the cop can do the following:

- If R moves from $[w, i]$ to $[w, i-1]$, (if $0 \leq i-1 \leq n$), then C_i will move from $[w', i]$ to $[w', i-1]$
- If R moves from $[w, i]$ to $[w, i+1]$, (if $0 \leq i+1 \leq n$), then C_i will move from $[w', i]$ to $[w', i+1]$, if $[w, i+1]$ and $[w', i]$ are not adjacent
- If R moves from $[w, i]$ to $[w, i+1]$, (if $0 \leq i+1 \leq n$), then C_i will capture R, if $[w, i+1]$ and $[w', i]$ are adjacent
- These similar moves are applied even when the value of either w or i changes. Thus, C_i always keeps an image of R captured or captures R.

□

Degree of Freedom: Suppose the cop C_i has captured one of the images of R (not R itself). We then define the *degree of freedom of R*. The *degree of freedom of R*, denoted by $D(R)$, is the largest value k, where $0 \leq k \leq n$, such that R is in one of the copies of B_k but C_i does not belong to the exact same copy of B_k. If $D(R) = k$, then the copy of B_k such that R is in B_k, is called the *freedom graph of R*.

Observation: If $D(R) = k$, then while C_i may not be in R's freedom graph B_k, it can be in any of the other copies of B_k and it is obviously in the copy of B_{k+1} containing R.

Theorem 9. *If $D(R) = k$, then R cannot move to $Ext(B_{k+1})$.*

Proof. We notice that if R wants to move to $Ext(B_{k+1})$, then it must first come to an external vertex of its freedom graph B_k. Then by theorem 8, C_i too will move to the corresponding vertex in its copy of B_k. Also, we note as shown in figure 12 that the vertex of R will have exactly two neighbours in $Ext(B_{k+1})$, which are precisely the two neighbours of the vertex of C_i from $Ext(B_{k+1})$, by theorem 5. Thus, any movement of R from its current position to a vertex of $Ext(B_{k+1})$, will result in its capture. □

Corollary 9.1. *If $D(R) = k$, then it can't move out of its freedom graph B_k.*

Proof. This follows from theorem 9. □

Corollary 9.2. *If $D(R) \leq k$, then it can't move out of its freedom graph.*

Proof. This too follows from the theorem 9 above. □

Theorem 10. *Let C_1 and C_2 be two cops with B_n, where $n \geq 2$, being our graph in consideration. Suppose, C_1 has captured an image of R, thus making its degree of freedom to be k, such that $0 \leq k < n$. Then, using C_2 in Algorithm 1, either we can capture R or decrease $D(R)$ even further.*

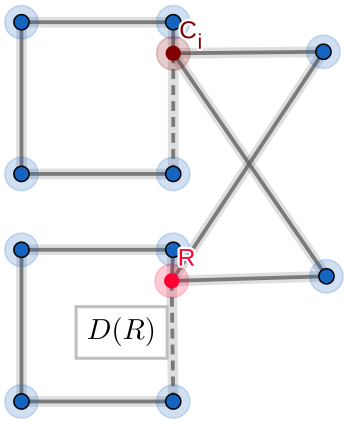

Figure 12: When the robber R tries to escape from $D(R)$

Proof. Let $D(R) = k$, due to C_1, such that $0 \leq k < n$. While R is moving within its freedom graph B_k we bring C_2 to the $top - right$ vertex of this copy of B_k,i.e, the vertex $[w, n]$, such that w is all zeroes after the first $n - k$ bits. By, Corollary 9.1, R should not go out of its freedom graph B_k to avoid getting captured by C_1. Thus, we might as well, consider the graph to be the freedom graph of R. We, now repeat the algorithm 1 with C_2, R and B_k. Thus, by the theorem 6 already proved, either C_2 will capture R or reduce $D(R)$ to k', such that $0 \leq k < k'$. □

Corollary 10.1. *$D(R)$ can be reduced to 0.*

Proof. By the last theorem, we can alternatively use C_1 and C_2 to keep decreasing the value of D_n until it becomes zero. □

Corollary 10.2. *It is possible to capture the robber R using only two cops.*

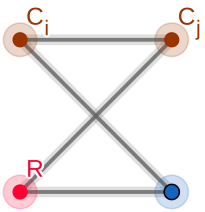

Figure 13: When $D(R) = 0$ and the robber can be captured

Proof. Once, $D(R) = 0$, due to say C_i, we bring C_j, as shown in figure 13 ($j \neq i$), to one of the neighbours of R. On its next turn if R stays put then it will get captured by C_j. If R moves to its neighbour unoccupied by C_j, then it becomes adjacent to C_i, thus getting captured on the latter's next turn. □

We thus have the following algorithm for capturing the robber R using two cops C_1 and C_2

Algorithm 2 Capture the Robber R

Require: Positions of the Cops C_1 and C_2 and the robber R and the graph under consideration, B_n

Ensure: R get captured

while R has not been captured **do**

Alternatively use C_1 and C_2 in Algorithm 1 to decrease the degree of freedom to zero

When degree of freedom of R is zero we use one of the cops to capture it

end while

Report capture and terminate the game

Theorem 11. *The Cop number of a Butterfly Network B_n, is one if $n = 0$, else it is two.*

Proof. If $n = 0$, then we can capture the robber R by using only one cop as it is only a single-vertex no edge graph.

If $n = 1$, then we have a four-cycle whose cop number is known to be two.

If $n \geq 2$, then as it is will have copies of the four cycle, so one cop is not sufficient for the capture, and by Corollary 10.2, two cops are sufficient. Thus in this case, the cop number is two as well.

Therefore, the theorem has been proved. □

Time Analysis: Before we move on to give a time analysis of the algorithm to capture the robber in a butterfly network we need the following theorem.

Theorem 12. *In a butterfly network B_n, $n \geq 0$, the largest possible value of the $d_E(C_i, R)$ in Algorithm 1, is n.*

Proof. We will prove this by induction on n on the initial value of $d_E(C_i, R)$.

Base Case: When $n = 0$, it is obvious the distance as well as the effective distance between the robber R and the cop C_i is zero.

Induction Hypothesis: Let the claim be true when $n = m$, for some $m \in \mathbb{N}$.

Inductive Step: Let $n = m + 1$. Initial position of C_i is in $[w, m+1]$, where w is the $m+1$-bit representation of 0.

Case i: R **is not in** $Ext(B_{m+1})$. In this case if both R and C_i are in the same copy of B_m, we can apply the induction hypothesis to deduce that the maximum value of $d_E(C_I, R)$ is $m \leq m + 1$. If they are in different copies then we consider the image of R closest to C_i to be actual R, in which case too, by our induction hypothesis, the maximum value of $d_E(C_i, R)$ should be $m \leq m + 1$.

Case ii: R **is in** $Ext(B_{m+1})$. In this case the actual distance is the effective distance, and the maximum value of actual distance possible is $m+1$, since all the vertices of $Ext(B_{m+1}$ are at a distance of $m+1$ from $[w, m+1]$.

Now, as our algorithm proceeds, by theorem 6, the value of $d_E(C_i, R)$ keeps decreasing. Thus, proved. □

So, if the cops begin at $[w, n]$ in B_n, where w is the n-bit binary representation of 0 and the robber R at $[w', 0]$, where w' is the n-bit binary representation of $2^n - 1$, we will have the maximum effective distance possible by theorem 12, which is n. As, algorithm 1 will cause one cop say C_1 to move closer to the effective position of R, the time needed to capture either R or its image is at most n. Upon the image capture of R by C_1, we need to bring C_2 to the copy of B_{n-1} where R is located. This, will take $n - 1$ steps to cover the maximum effective distance within B_{n-1}, then one more to move to a vertex in $Ext(B_n)$, one more to move from there to the copy of B_{n-1} where R has been restricted to. Finally, in order to move to $[w, n-1]$, where w is the $n-$bit binary number starting with 1 and followed by all 0's there needs to be at most n more steps. Thus, in order for a cop C_1 to capture the image of R in B_n and for C_2 to reach the position where the algorithm 1 can be applied again we need atmost:

$n + (n-1) + 1 + 1 + n = 3n + 1 \leq 4n$.

Thus, to keep decreasing the value of $D(R)$ until it reaches 0 and gets captured we need at most:

$4(n + \overline{n-1} + \ldots\ldots + 1 + 1) = 2n(n+1) + 4 = 2n^2 + 2n + 4 = O(n^2)$.

So, the time complexity to capture the robber R using two cops in a butterfly network B_n is $O(n^2)$.

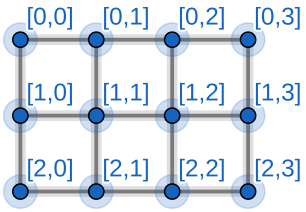

Figure 14: A 3×4 grid

4 Grid Graphs

There is another class of networks called *grid graphs*. We will begin our discussion with *rectangular grid graphs*.

Definition of rectangular grid graph: Formally an $m \times n$ rectangular grid $G_{m,n}$ is a graph whose vertices are arranged in m rows and n columns, where each vertex $v_{i,j}$, $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$ is connected to the four vertices $v_{i-1,j}$, $v_{i+1,j}$, $v_{i,j-1}$, $v_{i,j+1}$, whenever these indices stay inside the closed intervals $[0, m-1]$ and $[0, n-1]$ respectively. Then the vertices can be divided into three sets, namely: *corner vertices*, where both the subscripts i and j have the values 0 or $m-1$, and 0 or $n-1$, respectively; *border vertices*, where one of the subscripts i and j has the value 0 or $m-1$, or 0 or $n-1$, respectively; *internal vertices*, i.e. all the others. Corner, border, and internal vertices have two, three, and four neighbors each.

4.1 Cops and Robber Game in Rectangular Grids[6]

If a rectangular grid is a path then we know that the cop number is one. If it is a four-cycle then we know that the cop number is two. Our claim is that an arbitrary rectangular grid $G_{m,n}$ which is not a path also has cop number two as well. This was proved in [6] . Before we present the proof, we observe the following, with R being the robber and C_1, C_2 being the two cops:

Coordinates of an agent x: Let $v_{i,j}$ be the position of an agent x, then we say that (i, j) are the *coordinates of agent* x.

Horizontal Distance between C_k and R: Let (i, j) be the coordinates of a cop C_k and (i', j') be the coordinates of R. Then we say that the horizontal distance between cop C_k and R, denoted by $d_h(C_k, R)$ is defined as follows:

$d_h(C_k, R) := |j - j'|$.

Vertical Distance between C_k and R: Let (i, j) be the coordinates of a cop C_k and (i', j') be the coordinates of R. Then we say that the vertical

distance between cop C_k and R, denoted by $d_v(C_k, R)$ is defined as follows:
$d_v(C_k, R) := |i - i'|$.

Effective Distance between C_k and R: The *effective distance between C_k and R*, denoted by $d_E(C_k, R)$, is defined as follows:
$d_E(C_k, R) := max\{d_h(C_k, R), d_v(C_k, R)\}$.

Cornering of R by a cop C_k: We say that the robber R has been *cornered by cop C_k* if either of the following happens:

- The coordinate of R is $(0, 0)$ whereas that of C_k is $(1, 1)$
- The coordinate of R is $(m-1, 0)$ whereas that of C_k is $(m-2, 1)$
- The coordinate of R is $(0, n-1)$ whereas that of C_k is $(1, n-2)$
- The coordinate of R is $(m-1, n-1)$ whereas that of C_k is $(m-2, n-2)$

Idea behind the algorithm: Our algorithm will use one of the cops, say C_1 to corner the robber. If R moves to one of its neighbours then it will get captured by C_1, thus its forced to remain still on the corner. Then the other cop, say C_2 will occupy one of the two neighbours of R. Now, if on its turn R moves to its unoccupied neighbour it will get captured by C_1 otherwise by C_2.

Theorem 13. *The above algorithm is correct, that is, the cop C_1 either captures the robber R or corners it.*

Proof. Without loss of generality, let the effective distance be initially in the horizontal direction. Our algorithm will cause C_1 to decrease it by moving in the horizontal direction. This goes on until the effective distance is in the vertical direction. For, R to evade capture it must try to increase the distance in the direction of the effective distance. As our graph is finite so R will reach one of the border vertices. Once, the direction of effective distance changes so will the movement of C_1. This, will cause R to move in the extreme of the border it is in already and thus get cornered by C_1. □

Corollary 13.1. *It is possible in a rectangular grid $G_{m,n}$ for two cops to capture R.*

Proof. Once, cornered by C_1, C_2 can come to occupy one of the neighbours of vertex of R. If, on its turn R moves to its unoccupied neighbour it gets captured by C_1 else by C_2 as shown in figure 15. □

Theorem 14. *The cop number of a rectangular grid $G_{n,m}$ which is not a path is 2.*

Algorithm 3 Corner the Robber R

Require: Position of the robber R and the graph under consideration, $G_{n,m}$
Ensure: Either C_1 captures R or C_1 corners R

Initially the position of both C_1 and C_2 is $(0, 0)$, while R can start from anywhere
while R is un-captured and R is not cornered by C_1 **do**
  Calculate $d_h(C_1, R), d_v(C_1, R)$ and $d_E(C_1, R)$
  if Effective distance is the same as the horizontal distance **then**
    Move the cop C_1 from its position, say $v_{i,j}$, to $v_{i,j+1}$
    Update the positions of all agents
    if C_1 is adjacent to R **then**
      Let C_1 capture R
      Report capture and terminate the game
    end if
  else
    Move the cop C_1 from its position, say $v_{i,j}$, to $v_{i+1,j}$
    Update the positions of all agents
    if C_1 is adjacent to R **then**
      Let C_1 capture R
      Report capture and terminate the game
    end if
    Update the positions of all agents
  end if
end while

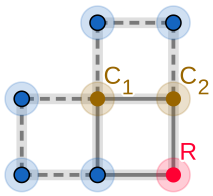

Figure 15: Capture after getting cornered in a rectangular grid

Proof. If we have one cop, then as a rectangular grid which is not a path will have in it four-cycles so the robber can simply evade capture by maintaining a distance of two in a cycle.

Also by corollary 13.1, we have already shown that two cops can capture the robber, thus the cop number of the graph class in 2. □

Time Analysis of Algorithm 3: The maximum value of effective distance is d, where $d = max\{n-1, m-1\}$. Also, the maximum actual distance possible between a cop and the robber is d', where $d' = n+m-2$. So, for the cop C_1 in our algorithm the time needed to make the effective distance to be 1 is $O(m+n)$. After that in order to corner R, C_1 needs $O(d)$ time to corner R. Once, R is cornered, C_2 will take $O(m+n)$ time to reach the desired position. Thus, the capture takes place in $O(m+n+d+m+n) = O(m+n)$ time.

4.2 Solid Grid Graphs: A Work in Progress

The two-dimensional integer grid G^∞ is an infinite graph with vertex set of all points of the Euclidean plane with integer coordinates. In this graph, there is an edge between any two vertices of unit distance. For a vertex v of this graph, let v_x and v_y denote x and y coordinates of its corresponding point.

A *grid graph* is a finite vertex-induced subgraph of the two-dimensional integer grid. In a grid graph, each vertex has degree of at most four. We notice that the rectangular grid graphs we discussed before are precisely the grid graphs $G(n, m)$, whose vertex set is $V := \{v | 0 \leq v_x \leq n-1, 0 \leq v_y \leq m-1\}$.

A *solid grid graph* is a grid graph without holes, i.e., all interior faces are four cycles. In the next few sections we will try to determine the cop number of partial grids in general. ***However, the reader is requested to keep in mind that it is still a work in progress so concrete results in this regards may not be possible. In fact, some of the very claims are based on intuitive ideas rather than formal proofs.***

4.2.1 Decomposition of solid grid graphs

A solid grid graph is either a rectangular grid or made up of multiple rectangular grids. Since, the vertices of the solid grid have coordinates in $\mathbb{R}^2$, so one can define the height, $h(G)$ and the width $w(G)$ of the graph. Essentially, $h(G) :=$ *difference between maximum and minimum y-coordinates of the vertices in* G and $w(G) :=$ *difference between maximum and minimum*

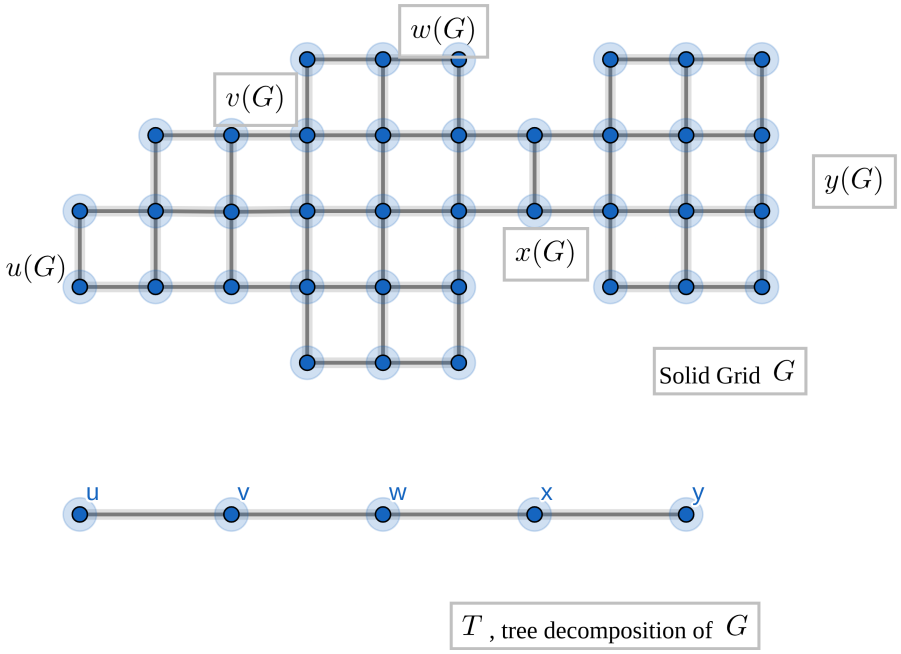

Figure 16: A solid grid G and its tree decomposition T

x-coordinates of the vertices in G. It seems possible that any solid grid G with $h(G) \leq w(G)$ can be decomposed to a graph T as follows:

- We start from the leftmost vertex in G, i.e., the one with the smallest x-coordinate
- From there we keep including vertices and the edges incident upon then as long as we can maintain the structure of a rectangular grid
- Once we see we can no more maintain rectangular property we stop
- We then begin with the border vertices of the above grid and do the same until all the vertices are covered
- For each rectangular grid formed with represent it with a node in T
- Two nodes in T are adjacent iff their corresponding grids share vertices

We do in a similar manner when $h(G) > w(G)$.

If u is node in T, then its corresponding rectangular grid in G is denoted by $u(G)$. We show a decomposition of a solid grid in figure 16. ***Note the following claims although labelled as lemma are actually claims based on intuition. The "proofs" too are statements of intuitive observations, as those are all works in progress.***

Lemma 15. *T is connected.*

Proof. Let $u, v \in V(T)$. If their respective grids in G, say $u(G)$ and $v(G)$ share vertices then we are done. If not then there must be some finitely many rectangular grids in between them. By our construction this will give arise to a path in T. Thus, T is connected. □

Lemma 16. *T is acyclic.*

Proof. If possible, let T be cyclic. Let the largest cycle in T be u_1—u_2—.....—u_k—u_1. Then the path u_1–u_k will have their rectangular grid in G placed side by side. However, it is not possible for the rectangular grid $u_1(G)$ to share any vertices with $u_k(G)$ as this will create holes in G, which is not possible. Thus, T can have no cycles. □

4.2.2 Cops and Robber games in solid grid graphs

We have already seen that the cop number of rectangular grid is 2. We think it is possible for solid grids as well that the cop number is 2 in general. Suppose for a solid grid G, the decomposition T happens to be a path as in 16. Then, we begin with both the cops in the top left corner of the leftmost rectangular grid, say u. If the robber is in that grid we use the two cops to corner it as in the normal rectangular grid. Then, either the robber gets captured or moves on to the next rectangular grid, say v. Notice, in this case there is a unique vertex in the shared vertices between u and v which is closest to the robber. We use both our cops to capture that unique vertex. This is possible as that unique vertex, even if it changes position due to the robber's movement within v, will lie in a path, which is essentially the border vertices of u shared with v. Once, captured we can then make sure that one cop, say C_1, always keeps this unique vertex captured thus, preventing R's re-entry to u. This, reduces R's area of movement in T. In the meantime, C_2 can either use the usual cornering algorithm in v or move to the unique vertex in the border of v which is closest to robber if it has moved to the grid whose representative in T is the successor the representative of v in T. If while doing so, the cop C_2 becomes diagonally opposite to R, then we can corner it and hence capture it. Otherwise, we keep moving C_2 along the shortest path to R. It seems like this should allow C_2 at some point to be in a position to R like C_1. Thus, we can alternatively use C_1 and C_2 to finally corner R in the last vertex of the path T. What happens if T is non-path tree? Well, even in that case we can consider the unique path between the cops and R and do the same alogrithm.

Of course, as stated earlier, all the above assertions are intuitive ideas about a work on progress and we need to come up with formal rigorous proof.

References

[1] Anthony Bonato and Richard J. Nowakowski. *The Game of Cops and Robbers on Graphs*. American Mathematical Society,2010.

[2] S.Das and H.Gahlawat. *Variations of cops and robber games on grids*. Discrete Applied Mathematics, 2020.

[3] Fabrizio Luccio and Linda Pagli. Cops and robber on grids and tori,2017,ArXiv e-prints.
`https://arxiv.org/pdf/1708.08255.pdf`

[4] I.Rajasingh, P. Manuel, P. Natarajan,A. Jemilet and R. Sundara Rajan, 2015. Transmission in Butterfly Networks. The Computer Journal. 59. 10.1093/comjnl/bxv127.

[5] R. Nowakowski, P. Winkler, *Vertex-to-vertex pursuit in a graph*, Discrete Math. 43 (1983) 253–259.

[6] S. Neufeld, R. Nowakowski. *A game of cops and robbers played on products of graphs*. Discrete Math. 186 (1998) 253–268.

[7] D.B. West, *Introduction To Graph Theory*, second ed., Pearson Education, 2002.