Markov Models for the Tipsy Cop and Robber Game on Graph

Viktoriya Bardenova, Vincent Ciarcia, Erik Insko

Florida Gulf Coast University, Fort Myers, Florida
Email: vbardenova@fgcu.edu, vciarcia@fgcu.edu, einsko@fgcu.edu

Abstract

In this paper we analyze and model three open problems posed by Harris, Insko, Prieto-Langarica, Stoisavljevic, and Sullivan in 2020 concerning the tipsy cop and robber game on graphs. The three different scenarios we model account for different biological scenarios. The first scenario is when the cop and robber have a consistent tipsiness level through the duration of the game; the second is when the cop and robber sober up as a function of time; the third is when the cop and robber sober up as a function of the distance between them. Using Markov chains to model each scenario we calculate the probability of a game persisting through M rounds of the game and the expected game length given different starting positions and tipsiness levels for the cop and robber.

Keywords

Cops and Robbers, Graph Searching, Markov Chains

1. Introduction

The game of cops and robbers on graphs was introduced independently by Quilliot [1] and Nowakowski and Winkler [2]. In this game, a cop and a robber alternate turns moving from vertex to adjacent vertex on a connected graph G with the cop trying to catch the robber and the robber trying to evade the cop. In 2014, Komarov and Winkler studied a variation of the cop and robber game in which the robber is too inebriated to employ an escape strategy, and at each step he moves to a neighboring vertex chosen uniformly at random [3].

In 2020, Harris, Insko, Prieto-Langarica, Stoisavljevic, and Sullivan introduced another variant of the cop and robber game that they call the tipsy cop and drunken robber game on graphs [4]. Each round of this game consists of independent moves where the robber begins by moving uniformly and randomly on

the graph to an adjacent vertex from where he began, this is then followed by the cop moving to an adjacent vertex from where she began; since the cop is only tipsy, some percentage of her moves are random and some are intentionally directed toward the robber.

In this paper, we generalize the work of Harris *et al.* to analyze the tipsy cop and tipsy robber game. One inspiration for this study to model is the biological scenario illustrated in the YouTube video [5] https://www.youtube.com/watch?v=Z_mXDvZQ6dU where a neutrophil chases a bacteria cell moving in random directions. While the bacteria's movement seems mostly random, the neutrophil's movement appears slightly more purposeful but a little slower.

Harris *et al.* modeled the game by having the players alternate turns. In this paper we use a slightly different model of the game that was suggested to us by Dr. Florian Lehner. We let the cop and robber be any amount of tipsy and rather than assume that the players alternate turns and flip a coin to determine if the player moves randomly or not, we use a spinner wheel to determine the probability of whether the next move will be a sober cop move, a sober robber move, or a tipsy move by either player. We model this scenario on vertex-transitive graphs (both finite and infinite examples) and on non-vertex transitive graphs (friendship graphs) using the theory of Markov chains. Given a set of initial conditions on each player's tipsiness and their initial distance, the questions we consider are:

- What is the probability $P(i, j, \mathbf{M})$ that the game, beginning in state *i*, will be in state *j* after exactly $\mathbf{M}$ rounds?
- What is the probability $\mathfrak{G}_{\mathbf{M}}(d)$ that the game lasts at least $\mathbf{M}$ rounds if they start distance *d* away?
- What is the expected number $E(d)$ of rounds the game should last if they start distance *d* away?

For some of these graphs we are able to identify when the expected capture time is finite and when the robber is expected to escape.

In Section 2 we introduce our model of the tipsy cop and robber game on both vertex-transitive and non-vertex transitive graphs. In Section 3 we analyze the game on regular trees and compare it to the famous Gambler's ruin problem [6]. In Section 4 we present our general method for analyzing the game using Markov chains, and we then apply these methods to analyze the game on specific families of graphs in Sections 5-8. In Section 5 we analyze the game on cycle graphs. In Section 6 we analyze the game on Petersen graphs. In Section 7 we analyze the game on friendship graphs. In Section 8 we analyze the game on toroidal grids. In Section 9 we present a model for the game where players start off drunk and sober up as the game progresses, answering Question 6.1 from Harris *et al.* [4]. Finally, in Section 10 we present a model for the game where the players' tipsiness increases as a function of the distance between them, thus providing an answer to Question 6.6 from Harris *et al.* [4]. While we present our methods with specific examples from each family of graphs we analyze, we also in-

clude our SageMath (CoCalc) code in the Appendix so that any reader may adapt our methods to their own models.

2. Background and Introduction of Our Model

In this paper we model the tipsy cop and tipsy robber game on a graph G by first placing the cop on one vertex and the robber on another vertex on the graph G. Rather than require the players alternate turns as in previous models of the cops and robbers game, we allow for four possible outcomes in each round of the game: a sober robber move r, a sober cop move c, a tipsy robber move t_r, or a tipsy cop move t_c, where $c + r + t_c + t_r = 1$. The outcome of each round is assigned at random, perhaps by a probability spinner as depicted in **Figure 1**.

- r is the probability of a sober robber move (in yellow)
- c is the probability of a sober cop (in green)
- t_r is the probability of a tipsy move by the robber (in red)
- t_c is the probability of a tipsy move by the cop (in blue)

If the game is played on a non-vertex transitive graph the probability of transition from one state to another depends on the starting location of the players. One such graph is a friendship graph, where n copies of the cycle graph C_3 are joined by a common vertex (see **Figure 2** for an example of a friendship graph with 3 copies of the C_3 cycle graph).

A vertex-transitive graph is a graph, where every vertex has the same local environment, so that no vertex can be distinguished from any other based on the vertices and edges surrounding it. On a non-vertex-transitive graph a tipsy move by the cop is not necessarily equivalent to a tipsy move by the robber, as it may increase, keep the same, or decrease the distance between the players based on each player's starting state. For example, the first part of **Figure 2** gives us the possible movements of the cop (in blue) if she is in the center of a friendship graph with 3 triangles. If the cop makes a tipsy move to a green vertex (which accounts for $\frac{4}{6}$ of t_c moves), the distance between her and the robber (in red) increases to 2. If she moves tipsy to the vertex in orange the distance will not change ($\frac{1}{6}$ of t_c moves), and if she moves to the vertex in red (that move can be either $\frac{1}{6}$ of t_c moves or the only sober move) she will catch the robber (distance will be 0). On the other hand, if the cop's starting position is on an outer vertex (second part of **Figure 2**) there are only two possible outcomes, she moves closer to the robber (in black-either $\frac{1}{2}$ of t_c or the only sober move) or keeps the same distance (in orange, $\frac{1}{2}$ of t_c moves).

Similarly, the starting state of the robber determines the probability of moving to the next state. The first part of **Figure 3** depicts the possible movements of the robber (in red) if he is in the center of a friendship graph with 3 triangles. If the
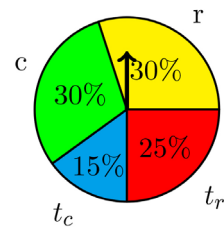
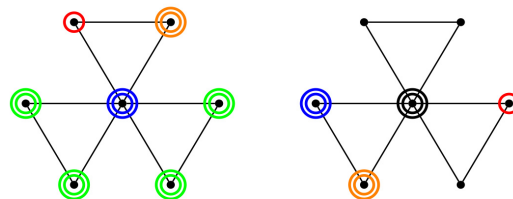**Figure 1.** Probability of each move based on spinner.

Figure 2. Possible cop moves (double circles) on a friendship graph with 3 triangles.
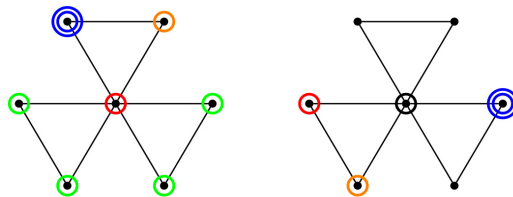

Figure 3. Possible robber moves (single circles) on a friendship graph with 3 triangles.

robber makes a sober or tipsy (which accounts for $\frac{4}{6}$ of t_r moves) move to a green vertex, the distance between him and the cop (in blue) increases to 2. If he moves tipsy ($\frac{1}{6}$ of t_r moves) to the vertex in orange the distance will not change, and if he moves to the vertex in blue (a different tipsy move $\frac{1}{6}$ of t_r moves) he will get caught (distance will be 0). On the other hand, if the robber's starting position is on an outer vertex (second part of **Figure 3**) there are only two possible outcomes, he moves closer to the cop (in black, $\frac{1}{6}$ of t_r moves) or keeps the same distance (in orange, $\frac{1}{6}$ of t_r moves or a sober move).

We will analyze the friendship graph game in more detail in Section 7.

The game is simpler to model on vertex-transitive graphs. A sober cop move will always decrease the distance between the two (as the cop is chasing the robber). A sober robber move will always increase the distance between the two, or if they are at a maximum distance from each other on a finite graph, he can decide to stay in the same place. Finally, a tipsy move by either player is a random move, and therefore may increase or decrease the distance between the two. When modeling the game on a vertex-transitive graph, a tipsy move by the cop is equivalent to a tipsy move by the robber, so we regroup all tipsy moves to-

gether. Every move in the game is guaranteed to fall in one of those three categories hence, $c + r + t = 1$ (**Figure 4**).

- r is the probability of a sober robber move (in yellow)
- c is the probability of a sober cop (in green)
- t is the probability of a tipsy move by either player (in red)

For example, if the game is played on an infinite path as depicted in **Figure 5**, the probability that the distance between the cop and robber increases by one in a round is equal to the probability of a sober robber move, since a sober robber will always run from the cop, plus one half of the probability of a tipsy move by either player, since half of all random moves will increase the distance between them. We will employ Markov chains and transformation matrices to model how the players move from one state to another on these graphs.

3. Regular Trees and the Gambler's Ruin Problem

On an infinite regular tree of degree Δ, the distance between the two players decreases with probability $\dfrac{1}{\Delta}$ and increases with probability $\dfrac{\Delta-1}{\Delta}$ when either player makes a tipsy move. When the cop makes a sober move, the distance always decreases, and when the robber makes a sober move the distance always increases. Hence, if we assume that the cop calls off the hunt when the distance between the players reaches a specified distance $d = \mathfrak{n}$, then the Markov chain in **Figure 6** models the game where the probability of the distance increasing is $p = t\left(\dfrac{\Delta-1}{\Delta}\right) + r$ and the probability of the distance decreasing is $1 - p = c + \dfrac{t}{\Delta}$.
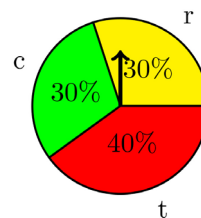

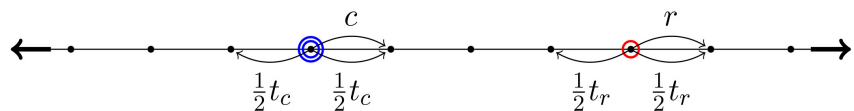**Figure 4.** Probability of each move based on spinner where tipsy moves are grouped together.

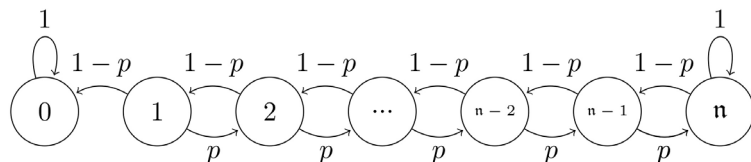**Figure 5.** Move of a tipsy cop (blue) and tipsy robber (red).

Figure 6. Markov chain on infinite regular trees.

Those familiar with the gambler's ruin problem will immediately realize that this Markov chain is the same as the gambler's ruin chain with a probability of the gambler winning a round given by $p = t\left(\frac{\Delta-1}{\Delta}\right) + r$ and the gambler losing a round given by $1 - p = c + \frac{t}{\Delta}$.

It is well-known that the expected game length of the gambler's ruin problem is given by the equation $E(d) = d(\mathfrak{n} - d)$ when $p = \frac{1}{2}$ ([7], p. 79) and

$$E(d) = -\frac{\mathfrak{n}}{1-2p} \frac{\left(\frac{1-p}{p}\right)^d - 1}{\left(\frac{1-p}{p}\right)^{\mathfrak{n}} - 1} + \frac{d}{1-2p} \tag{1}$$

when $p \neq \frac{1}{2}$ ([7], p. 79].

After finding a common denominator and factoring out $\frac{1}{1-2p}$ we can rewrite $E(d)$ using our notation as

$$E(d) = \left(\frac{1}{1-2p}\right)\left(d - \mathfrak{n} \cdot p^{\mathfrak{n}-d} \cdot \left(\frac{p^d - (1-p)^d}{p^{\mathfrak{n}} - (1-p)^{\mathfrak{n}}}\right)\right). \tag{2}$$

When $p < \frac{1}{2}$ (favoring cop success) then the right hand side of Equation (2) has a limit of d as $\mathfrak{n} \to \infty$. So if the cop never gives up, we get the formula

$$E(d) = \frac{d}{1-2p} = \frac{d\Delta}{\Delta - 2(1-c)(\Delta-1) - 2r} \tag{3}$$

$$= \frac{d\Delta}{(2c-1)\Delta + 2(1-c) - 2r}. \tag{4}$$

If $c > \frac{1}{2}$, then as $\Delta \to \infty$

$$E(d) = \frac{d}{2c-1}. \tag{5}$$

If $R(d)$ and $C(d)$ represent the probability of the robber or cop winning respectively when starting the game from distance d away from each other, then of course $R(d) + C(d) = 1$. Formulas for $R(d)$ and $C(d)$ can easily be derived from well-known formulas modeling the gambler's ruin problem. For instance, when $p = \frac{1}{2}$, the probability of the robber escaping $R(d)$ is the same as the probability of the gambler's success in the fair gambler's ruin problem $R(d) = \frac{d}{\mathfrak{n}}$ and when $p \neq \frac{1}{2}$, we have

$$R(d) = \frac{1 - \left(\dfrac{1-p}{p}\right)^d}{1 - \left(\dfrac{1-p}{p}\right)^{\mathfrak{n}}} \qquad (6)$$

is the same as the probability of the gambler's success in the unfair gambler's ruin problem ([6], Equation 1.2).

Substituting $p = t\dfrac{\Delta - 1}{\Delta} + r$ (transition probability to increase distance) and

$1 - p = c + \dfrac{t}{\Delta}$ (transition probability to decrease distance) we find $R(d)$ to be

$$R(d) = \frac{1 - \left(\dfrac{c + \dfrac{t}{\Delta}}{t\dfrac{\Delta - 1}{\Delta} + r}\right)^d}{1 - \left(\dfrac{c + \dfrac{t}{\Delta}}{t\dfrac{\Delta - 1}{\Delta} + r}\right)^{\mathfrak{n}}}. \qquad (7)$$

Numerical Examples

As an example, we consider the game on an infinite regular tree of degree $\Delta = 4$, where the proportion of moves follows the distribution of $c = 0.3$, $r = 0.4$, $t = 0.3$, and the cop calls off the chase if the robber reaches a distance of $\mathfrak{n} = 10$. Table 1 gives the expected game time $E(d)$, and the probability of the robber or cop winning from each possible starting distance *d*.

We observe that if this game begins at $d = 9$, then it has the lowest expected game time of any possible starting distance. This makes sense as 40% of the time the robber will escape in the first move, or with probability 22.5% either player will move drunkenly to allow the robber to escape in the first move; hence, the robber has a 66.5% chance of escaping in the first move if the game begins at $d = 9$.

Table 1. Expected game times and probabilities of cop or robber win.

Distance	Measure		
d	*E(d)*	*R(d)*	*C(d)*
1	12.10	0.4024	0.5976
2	17.76	0.6439	0.3561
3	19.55	0.7888	0.2112
4	19.03	0.8757	0.1243
5	17.11	0.9278	0.0722
6	14.37	0.9591	0.0409
7	11.12	0.9779	0.0221
8	7.567	0.9892	0.0108
9	3.838	0.9959	0.0041

4. General Matrix Method for Analyzing Markov Processes

In this section we describe how to use various matrix equations to find the critical data points of the game. These include, the probability of transitioning from one vertex to another in exactly $\mathbf{M}$ rounds, the probability the game lasts at least $\mathbf{M}$ rounds, and the expected game time. To calculate these values for a finite Markov chain we will use its probability matrix P, where $P_{i,j}$ is the probability of transitioning from state i to state j in one round of the Markov process. We will use these matrix methods repeatedly to analyze various families of graphs in the subsequent sections of this paper.

4.1. Calculating Probability of Moving from State *i* to State *j* in M Rounds

The probability of starting in state i and ending in state j in exactly $\mathbf{M}$ moves is

$$P(i, j, \mathbf{M}) = e_i \cdot P^{\mathbf{M}} \cdot e_j \tag{8}$$

where e_i and e_j are the row and column basis vectors respectively associated with state i and state j.

For example, given the Markov chain in **Figure 7**, the probability matrix for this Markov chain is

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1-p & 0 & p & 0 \\ 0 & 1-p & 0 & p \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{9}$$

Some probabilities of going from one state to another in $\mathbf{M}$ rounds are displayed in **Table 2**.
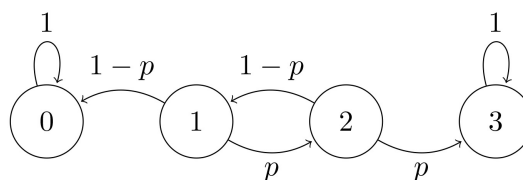
Figure 7. Sample for a Markov chain with four states.

Table 2. Multi-round state transition probabilities.

From state	to state	in $\mathbf{M}$ rounds	$P(i, j, \mathbf{M})$
$i = 3$	$j = 0$	$\mathbf{M} = 1$	$P(3,0,1) = 0$
$i = 3$	$j = 0$	$\mathbf{M} = 2$	$P(3,0,2) = 0$
$i = 3$	$j = 0$	$\mathbf{M} = 3$	$P(3,0,3) = 0$
$i = 2$	$j = 0$	$\mathbf{M} = 1$	$P(2,0,1) = 0$
$i = 2$	$j = 0$	$\mathbf{M} = 2$	$(1-p)^2$
$i = 2$	$j = 0$	$\mathbf{M} = 3$	$(1-p)^2 \cdot 1 + (1-p)^3 \cdot p$
$i = 2$	$j = 0$	$\mathbf{M} = 4$	$(1-p)^2 \cdot 1^2 + (1-p)^3 \cdot p$

Note that in this example, state 0 and state 3 are both absorbing states, and that it is possible when calculating $P(i,a,\mathbf{M})$ for an absorbing state *a*, that the robber reaches the absorbing state *a* in $m < \mathbf{M}$ moves, and then sits there for the remaining $\mathbf{M} - m$ moves.

4.2. Calculating the Probability $\mathfrak{G}_{\mathbf{M}}(d)$ That the Game Will Last at Least M Rounds

To calculate the probability that the cop will still be actively chasing the robber through $\mathbf{M}$ rounds of this game with a starting distance *d* between the players, we restrict attention to the matrix $T = P_{\text{transient}}$ modeling only the non-absorbing states of the Markov chain. Note that in this case, T is the matrix obtained from *P* by removing the columns and rows associated with any absorbing states in *P*. The probability of the game lasting at least $\mathbf{M}$ rounds of the game if the players start at distance *d* from each other is then given by the following product of matrices

$$\mathfrak{G}_{\mathbf{M}}(d) = e_d \cdot T^{\mathbf{M}} \cdot \mathbb{1} \tag{10}$$

where e_d denotes a standard basis row vector with 1 in column *d* and zero elsewhere, and $\mathbb{1}$ is the column vector with 1 in each entry.

4.3. Calculating Expectation *E*(*d*)

As the number of rounds goes to infinity the likelihood that the game is still going on goes to zero. If we sum all the $\mathfrak{G}_{\mathbf{M}}(d)$ probabilities for all $\mathbf{M}$ we can find the expected number of rounds the game should last. That is,

$$E(d) = e_d \cdot \left[I + T + T^2 + T^3 + \cdots + T^{\mathbf{M}} + \cdots \right] \cdot \mathbb{1} \tag{11}$$

$$= e_d \cdot \frac{1}{1-T} \cdot \mathbb{1} \tag{12}$$

$$= e_d \cdot (I-T)^{-1} \cdot \mathbb{1} \tag{13}$$

where *I* is the identity matrix.

5. Cycle Graphs

The study of cycle graphs breaks down into two families of cases based on if the number of vertices $\mathbf{n}$ is even or odd. We start by analyzing the case when $\mathbf{n}$ is even, and then explain how the odd case differs slightly.

For a cycle graph $C_{\mathbf{n}}$ with $\mathbf{n}$ nodes, where $\mathbf{n}$ is even we have the states where the robber and the cop are 0 (cop catches the robber or the robber is tipsy and stumbles upon the cop herself), $1, 2, 3, \cdots$ or $\frac{\mathbf{n}}{2}$ moves away from each other.

We assume that the cop always moves until they are at distance 0 away. Also, if they are at distance $\frac{\mathbf{n}}{2}$ away, and the robber is sober, he will stay at the same location since it is the furthest from the cop. The Markov chain for a cycle graph

with $\mathbf{n}$ nodes has $\dfrac{\mathbf{n}}{2}$ states as shown in **Figure 8**.

From this Markov chain we find the probability matrix P to be a tridiagonal $(\dfrac{\mathbf{n}}{2}+1 \times \dfrac{\mathbf{n}}{2}+1)$ matrix of the following form: $P_{1,1}=1$, $P_{\frac{\mathbf{n}}{2}+1,\frac{\mathbf{n}}{2}}=c+t$, $P_{\frac{\mathbf{n}}{2}+1,\frac{\mathbf{n}}{2}+1}=r$ and the upper and lower diagonal entries are $P_{k+1,k}=c+\dfrac{t}{2}$ for $1\le k\le \dfrac{\mathbf{n}}{2}-1$, and $P_{k+1,k+2}=r+\dfrac{t}{2}$ for $1\le k\le \dfrac{\mathbf{n}}{2}-1$, and finally $P_{i,j}=0$ otherwise.

The transition matrix T is derived by removing the first row and column corresponding to the absorbing state of P. Hence, T is an $(\dfrac{\mathbf{n}}{2}\times\dfrac{\mathbf{n}}{2})$ matrix with $T_{\frac{\mathbf{n}}{2},\frac{\mathbf{n}}{2}-1}=c+t$, $T_{\frac{\mathbf{n}}{2},\frac{\mathbf{n}}{2}}=r$, $T_{k+1,k}=c+\dfrac{t}{2}$ for $1\le k\le \dfrac{\mathbf{n}}{2}-2$, $T_{k,k+1}=r+\dfrac{t}{2}$ for $1\le k\le \dfrac{\mathbf{n}}{2}-1$, and $T_{i,j}=0$ otherwise.

When $\mathbf{n}$ is odd, the only difference in the Markov chain is that the probability of transitioning from state $\dfrac{\mathbf{n}}{2}$ to itself is $r+\dfrac{t}{2}$ and the probability of transitioning from $\dfrac{\mathbf{n}}{2}$ to $\dfrac{\mathbf{n}}{2}-1$ is $c+\dfrac{t}{2}$. Hence $P_{\frac{\mathbf{n}}{2}+1,\frac{\mathbf{n}}{2}}=c+\dfrac{t}{2}$, $P_{\frac{\mathbf{n}}{2}+1,\frac{\mathbf{n}}{2}+1}=r+\dfrac{t}{2}$, since the robber and cop move all the time.

We may now use our formulas involving T to find the probability that the game lasts at least $\mathbf{M}$ rounds as in Subsection 4.2 and the expected game time Subsection 4.3. It is also important to note that because our model does not guarantee turns alternate, and this graph is finite where the cop never calls off the chase, $C(d)=1$ for all values of d, c, r, t, and $\mathbf{n}$.

5.1. Numerical Examples

For cycle graph with $\mathbf{n}=6$ nodes we have the states where the robber and the cop are 0 (cop catches the robber or the robber is tipsy and stumbles upon the cop himself), 1, 2 or 3 moves away from each other as shown in **Figure 9**.

We assume that the cop always moves until they are at distance 0 away. Also, if they are at distance 3 away and the robber is sober he will stay at the same location since it is the furthest from the cop. We derive the Markov chain in **Figure 10** for a cycle graph with 6 nodes.

We use a stochastic matrix P to represent the transition probabilities of this system. The rows and columns in this matrix are indexed by the possible states
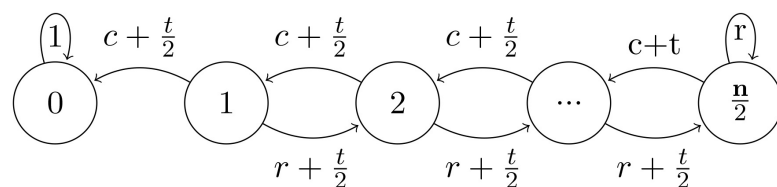
Figure 8. Markov chain for cycle graph with $\mathbf{n}$ nodes where $\mathbf{n}$ is even.
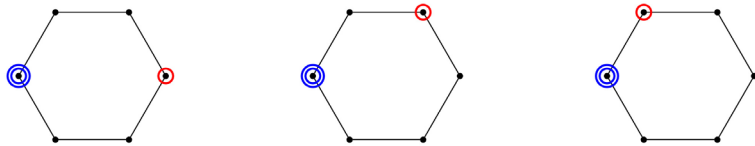
Figure 9. Distance between cop (blue double circle) and robber (red single circle) on a cycle C_6 graph.
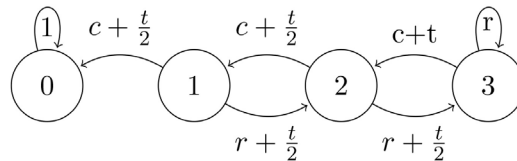
Figure 10. Markov chain on C_6 graph.

listed above, with the pre-transition state as the row and post-transition state as the column. Transition probability matrix

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ c+\dfrac{t}{2} & 0 & r+\dfrac{t}{2} & 0 \\ 0 & c+\dfrac{t}{2} & 0 & r+\dfrac{t}{2} \\ 0 & 0 & c+t & r \end{pmatrix}. \tag{14}$$

This transition probability matrix P can be restricted to a transient transition matrix T with absorbing state 0 removed

$$T = \begin{pmatrix} 0 & r+\dfrac{t}{2} & 0 \\ c+\dfrac{t}{2} & 0 & r+\dfrac{t}{2} \\ 0 & c+t & r \end{pmatrix}. \tag{15}$$

Using our general matrix method as in Section 4 we can solve the following numerical example.

5.2. Numerical Examples

We assume the tipsy moves by either player account for half of all moves $t = 0.5$ and $c + r = 1 - t = 0.5$. **Table 3** gives the probability the cop will still be chasing the robber after $\mathbf{M} = 7$ rounds provided the cop and the robber start at distance $d = 1, 2$ or 3 as the percentage of the sober moves allocated to the cop and the robber varies.

Table 3 also shows the game will last longest, and the chase has the largest probability of still continuing after 7 rounds, if the starting distance is 3 and the robber takes the maximum percentage of sober moves possible. The accompanying SageMath code for these computations is included in **Appendix A.1**, so the interested reader can adapt these calculations to model the game for any values of $\mathbf{M}, r, c, t$ they choose.

Table 3. Some survival rates and expected game durations on C_6.

Measure	Proportion of Sober moves $c+r=0.5$					
Robber	$r=0$	$r=0.1$	$r=0.2$	$r=0.3$	$r=0.4$	$r=0.5$
Cop	$c=0.5$	$c=0.4$	$c=0.3$	$c=0.2$	$c=0.1$	$c=0.0$
$\mathfrak{G}_7(1)$	0.020	0.073	0.169	0.307	0.473	0.649
$\mathfrak{G}_7(2)$	0.084	0.195	0.349	0.526	0.702	0.849
$\mathfrak{G}_7(3)$	0.084	0.222	0.402	0.59	0.759	0.887
$E(1)$	1.89	2.69	4.14	7.07	13.91	34
$E(2)$	3.56	4.83	6.98	11.04	19.86	44
$E(3)$	4.56	5.94	8.23	12.47	21.53	46

6. Petersen Graph

The Petersen graph is a vertex-transitive graph where the cop and robber can only be 0, 1, or 2 moves away from each other as illustrated in Figure 11.

We assume that the cop moves until eventually she captures the robber. Also, if robber is distance 2 away from the cop, and the robber is sober, he will stay at the same location since it is the furthest from the cop. Hence the Markov chain for the Petersen graph is as depicted in Figure 12.

We use a stochastic matrix P to represent the transition probabilities of this system. The rows and columns in this matrix are indexed by the possible states listed above, with the pre-transition state as the row and the post-transition state as the column.

$$P = \begin{pmatrix} 1 & 0 & 0 \\ c+\dfrac{t}{3} & 0 & r+\dfrac{2t}{3} \\ 0 & c+\dfrac{t}{3} & r+\dfrac{2t}{3} \end{pmatrix} \tag{16}$$

To calculate the probability of the game lasting at least **M** rounds and the expected number of rounds in the game, we remove the absorbing state 0 to get the restricted transition matrix

$$T = \begin{pmatrix} 0 & r+\dfrac{2t}{3} \\ c+\dfrac{t}{3} & r+\dfrac{2t}{3} \end{pmatrix} \tag{17}$$

We may now use our formulas involving T to find the probability that the game will last at least **M** rounds as in Subsection 4.2 and the expected number of rounds as in Subsection 4.3.

Numerical Examples

In Table 4 we assume the tipsy moves by either player account for half of all moves $t=0.5$ and $c+r=1-t=0.5$. We then calculate the probability the cop
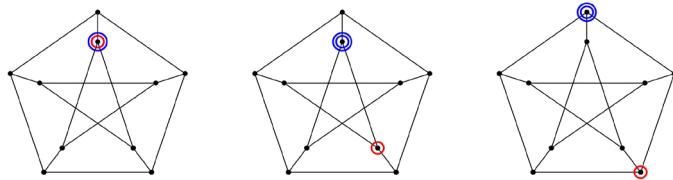
Figure 11. Distance between cop (blue double circle) and robber (red single circle) on a Petersen graph.
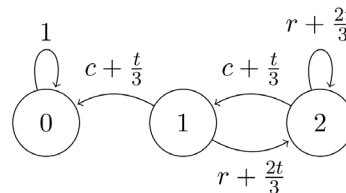
Figure 12. Markov chain for Petersen graph.

Table 4. Some survival rates and expected game durations on petersen Graph.

Measure	Proportion of Sober moves $c+r=0.5$					
Robber	$r=0$	$r=0.1$	$r=0.2$	$r=0.3$	$r=0.4$	$r=0.5$
Cop	$c=0.5$	$c=0.4$	$c=0.3$	$c=0.2$	$c=0.1$	$c=0.0$
$\mathfrak{G}_7(1)$	0.039	0.1	0.204	0.352	0.536	0.735
$\mathfrak{G}_7(2)$	0.078	0.175	0.318	0.496	0.687	0.86
$E(1)$	2.25	3.11	4.59	7.44	14.06	40
$E(2)$	3.75	4.87	6.73	10.17	17.81	42

will still be chasing the robber after $\mathbf{M}=7$ rounds based on their starting distances $d=1$ or 2 and on the percentage of the sober moves allocated to the cop and the robber.

We have published the accompanying SageMath code for these computations in **Appendix A.1**, so the interested reader can change these calculations to model the game for any values of $\mathbf{M}, r, c, t$ that they choose.

7. Friendship Graphs

As friendship graphs are not vertex-transitive, we cannot simply model the game on them with a Markov chain where each state is determined by the distance between the cop and robber, and we must treat tipsy cop moves and tipsy robber moves separately as the transition probabilities from state to state depend on who is moving. Hence, in this section the spinner we use has four distinct possible outcomes satisfying $r+c+t_r+t_c=1$.

The following notation will be used to model the tipsy cop and tipsy robber game on friendship graphs of n triangles. The states $1e$, $1cc$, and $1rc$ all refer to the cop and robber being distance 1 away from each other but, $1e$ means both players are on the same outer edge, $1cc$ means the cop is in the center, and $1rc$ means the robber is in the center as depicted in **Figure 13**.

Figure 13. All possible states of friendship graphs of $n = 2$ triangles.

Friendship graphs with $n = 2, 3$ and 4 triangles are shown in **Figure 14**. Since the cop never calls off the chase on this finite graph, the cop will win eventually $C(d) = 1$ even if the robber is completely sober throughout.

The Markov chain modeling the game on a friendship graph with n triangles is shown in **Figure 15**.

From this Markov chain we derive the transition probability matrix

$$P = \begin{pmatrix} r + \dfrac{t_c}{2} + \dfrac{t_r}{2} & c + \dfrac{t_c}{2} & \dfrac{t_r}{2} & 0 & 0 \\ t_c \cdot \dfrac{n-1}{n} & r + \dfrac{t_r}{2} & 0 & \dfrac{t_c}{2n} & c + \dfrac{t_c}{2n} + \dfrac{t_r}{2} \\ r + t_r \cdot \dfrac{n-1}{n} & 0 & \dfrac{t_c}{2} & \dfrac{t_r}{2n} & c + \dfrac{t_r}{2n} + \dfrac{t_c}{2} \\ 0 & \dfrac{t_c}{2} & r + \dfrac{t_r}{2} & 0 & c + \dfrac{t_c}{2} + \dfrac{t_r}{2} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

and transition matrix with absorbing state 0 removed

$$T = \begin{pmatrix} r + \dfrac{t_c}{2} + \dfrac{t_r}{2} & c + \dfrac{t_c}{2} & \dfrac{t_r}{2} & 0 \\ t_c \cdot \dfrac{n-1}{n} & r + \dfrac{t_r}{2} & 0 & \dfrac{t_c}{2n} \\ r + t_r \cdot \dfrac{n-1}{n} & 0 & \dfrac{t_c}{2} & \dfrac{t_r}{2n} \\ 0 & \dfrac{t_c}{2} & r + \dfrac{t_r}{2} & 0 \end{pmatrix}. \tag{19}$$

Sample Calculations

In this subsection we compute the expected game times and probability of the game lasting through $\mathbf{M} = 10$ rounds on a friendship graph with $n = 5$ cliques. **Table 5** summarizes the results. In these computations we assume that the cop and robber each get 50% of the moves so $r + t_r = c + t_c = 0.5$, but we vary the tipsiness of the cop and robber. The code for these computations is available in **Appendix A.4**.

8. Toroidal Grids

A toroidal grid is the Cartesian product of two cycle graphs $C_m \square C_n$ and represents a grid that can be embedded on the surface of a torus. Since our model does not guarantee turns alternate, and the cop never calls off the chase, the cop will eventually win $C(d) = 1$ on any toroidal grid.

The Markov chain for our model on a 7×7 toroidal grids is shown in **Figure 16**.
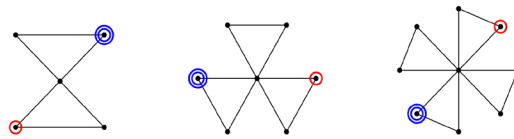
Figure 14. Examples of friendship graphs with $n = 2, 3, 4$ triangles with cop (blue) and robber (red).
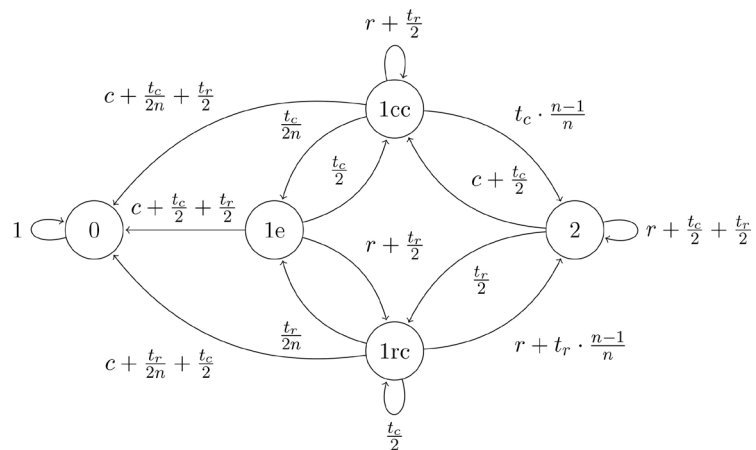
Figure 15. Markov chain for tipsy cop and tipsy robber game on friendship graphs of n cliques.
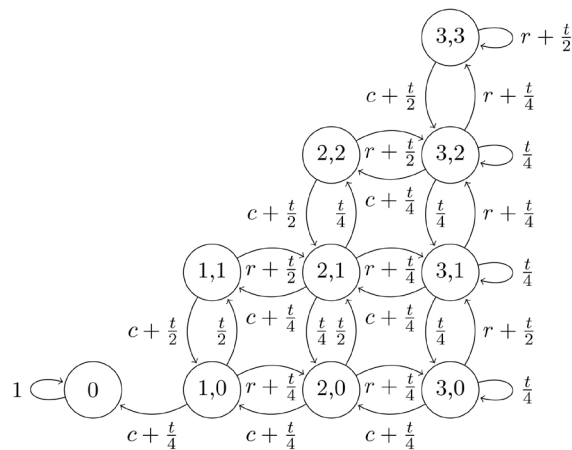
Figure 16. Markov chain for a 7×7 toroidal grid.

Table 5. Some survival rates and expected game durations on friendship graph.

Measure	Proportion of Tipsy moves								
Robber	$t_r = 0.1$			$t_r = 0.25$			$t_r = 0.4$		
Cop	$t_c = 0.1$	$t_c = 0.25$	$t_c = 0.4$	$t_c = 0.1$	$t_c = 0.25$	$t_c = 0.4$	$t_c = 0.1$	$t_c = 0.25$	$t_c = 0.4$
$\mathfrak{G}_{10}(2)$	0.0448	0.1836	0.4300	0.0252	0.1129	0.2836	0.0143	0.0691	0.1862
$\mathfrak{G}_{10}(1cc)$	0.0156	0.1043	0.3198	0.0076	0.0584	0.1948	0.0037	0.0326	0.1183
$\mathfrak{G}_{10}(1rc)$	0.0337	0.1275	0.2949	0.0192	0.0791	0.1960	0.0109	0.0486	0.1291
$\mathfrak{G}_{10}(1e)$	0.0229	0.0856	0.2167	0.0116	0.0473	0.1297	0.0056	0.0251	0.0752
$E(2)$	4.628	6.914	12.09	4.093	5.697	8.668	3.689	4.893	6.874

Continued

$E(1cc)$	2.540	4.505	9.344	2.160	3.543	6.333	1.878	2.914	4.769
$E(1rc)$	3.419	4.979	8.590	3.051	4.159	6.270	2.765	3.603	5.032
$E(1e)$	2.665	3.804	6.735	2.252	3.002	4.618	1.923	2.445	3.463

The number of states in our Markov chain model grows considerably as the number of nodes in $C_m \times C_n$ grows.

Therefore,

$$P = \begin{pmatrix}
r+\dfrac{t}{2} & c+\dfrac{t}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r+\dfrac{t}{4} & \dfrac{t}{4} & \dfrac{t}{4} & 0 & c+\dfrac{t}{4} & 0 & 0 & 0 & 0 & 0 \\
0 & r+\dfrac{t}{4} & \dfrac{t}{4} & \dfrac{t}{4} & 0 & c+\dfrac{t}{4} & 0 & 0 & 0 & 0 \\
0 & 0 & r+\dfrac{t}{2} & \dfrac{t}{4} & 0 & 0 & c+\dfrac{t}{4} & 0 & 0 & 0 \\
0 & r+\dfrac{t}{2} & 0 & 0 & 0 & c+\dfrac{t}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & r+\dfrac{t}{4} & 0 & \dfrac{t}{4} & 0 & \dfrac{t}{4} & c+\dfrac{t}{4} & 0 & 0 \\
0 & 0 & 0 & r+\dfrac{t}{4} & 0 & \dfrac{t}{2} & 0 & 0 & c+\dfrac{t}{4} & 0 \\
0 & 0 & 0 & 0 & 0 & r+\dfrac{t}{2} & 0 & 0 & c+\dfrac{t}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & r+\dfrac{t}{4} & \dfrac{t}{2} & 0 & c+\dfrac{t}{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \qquad (20)$$

and

$$T = \begin{pmatrix}
r+\dfrac{t}{2} & c+\dfrac{t}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
r+\dfrac{t}{4} & \dfrac{t}{4} & \dfrac{t}{4} & 0 & c+\dfrac{t}{4} & 0 & 0 & 0 & 0 \\
0 & r+\dfrac{t}{4} & \dfrac{t}{4} & \dfrac{t}{4} & 0 & c+\dfrac{t}{4} & 0 & 0 & 0 \\
0 & 0 & r+\dfrac{t}{2} & \dfrac{t}{4} & 0 & 0 & c+\dfrac{t}{4} & 0 & 0 \\
0 & r+\dfrac{t}{2} & 0 & 0 & 0 & c+\dfrac{t}{2} & 0 & 0 & 0 \\
0 & 0 & r+\dfrac{t}{4} & 0 & \dfrac{t}{4} & 0 & \dfrac{t}{4} & c+\dfrac{t}{4} & 0 \\
0 & 0 & 0 & r+\dfrac{t}{4} & 0 & \dfrac{t}{2} & 0 & 0 & c+\dfrac{t}{4} \\
0 & 0 & 0 & 0 & 0 & r+\dfrac{t}{2} & 0 & 0 & c+\dfrac{t}{2} \\
0 & 0 & 0 & 0 & 0 & 0 & r+\dfrac{t}{4} & \dfrac{t}{2} & 0
\end{pmatrix} \qquad (21)$$

Numerical Examples

In this section we model the game on a 7×7 Toroidal grid with the following proportion of moves $r = 0.4$, $c = 0.3$, $t = 0.3$. Table 6 shows the probability of the robber evading the cop through the first 50 rounds of the game as well as the expected game length based on each of the players' possible starting configurations. The SageMath code for these calculations is available to the interested reader in **Appendix A.2**.

9. Modeling the Game Where Players Sober up over Time

Harris *et al.* asked how to model a tipsy cop and drunk robber game when the cop is sobering up over time ([4], Question 6.1). In this section, we model the game where both players begin completely drunk and sober up as time passes. We define our probability of a tipsy move by either player t as a function of m rounds that have passed $t = f(m)$. Additionally, we set the proportion of sober cop moves to be $c = \dfrac{a}{b}(1-t)$ and the proportion of sober robber moves to be $r = \dfrac{b-a}{b}(1-t)$, where a and b are any desired integers that determine the proportion of sober moves that is assigned to the cop and to the robber. As we assume the players are sobering up as time passes, we choose to use a function f with the following properties:

$$f(1) = 1 \tag{22}$$

$$\lim_{m \to \infty} f(m) = 0 \tag{23}$$

with these assumptions, the probability of surviving $\mathbf{M}$ rounds, given an initial state d, is given by:

$$\mathfrak{G}_{\mathbf{M}}(d) = e_d \cdot \left(\prod_{m=1}^{\mathbf{M}} T_m \right) \mathbb{1} \tag{24}$$

The expected game time is given by the series:

$$E(d) = e_d \cdot \left(\sum_{n=1}^{\mathcal{N}=\infty} \left(\prod_{m=1}^{n-1} T_m \right) \right) \cdot \mathbb{1} \tag{25}$$

We conclude this section with some sample calculations in tables. The code for the calculations is attached in **Appendix A.3**, so the interested reader can change these calculations to model the game for any values of $a, b, t = f(m)$, or $\mathcal{N}$ they choose.

Table 6. Survival rates and expected durations on 7×7 toroidal grid.

Measure	Initial State d								
	(3,3)	(3,2)	(3,1)	(3,0)	(2,2)	(2,1)	(2,0)	(1,1)	(1,0)
$\mathfrak{G}_{50}(d)$	0.5480	0.5307	0.4942	0.4520	0.4958	0.4376	0.3731	0.3506	0.2331
$E(d)$	78.18	95.95	71.21	65.62	71.42	63.66	54.75	51.65	34.75

Numerical Examples

Assuming the game is played on a cycle graph of six nodes **Figure 9**, its accompanying Markov Chain is given in **Figure 10**.

Example 9.1.1. Let the tipsiness function be given by the formula $t = f(m) = \dfrac{4}{m+3}$ as it is an example of a function that follows the rules stated above. Then, **Table 7** shows the probability of the game lasting 5 rounds from varying starting positions and proportions of sober moves allotted to each player. It also shows estimates for expected game times from each starting position where we use

$$E(d) = e_d \cdot \left(\sum_{n=1}^{\mathcal{N}} \left(\prod_{m=1}^{n-1} T_m \right) \right) \mathbb{1} \qquad (26)$$

to approximate the infinite sum in Equation (25). We choose either $\mathcal{N} = 1000$ or $\mathcal{N} = 3000$ to approximate these values because these values of $\mathcal{N}$ are sufficiently large to approximate the expected game time accurately to at least five digits as we vary the proportion of sober moves allocated to the robber.

The expected game time for 80% was also calculated at $\mathcal{N} = 3000$, and found to be the same for at least the first five digits. It seems only at 90% and above is $\mathcal{N} = 1000$ not sufficiently large.

Example 9.1.2. Let tipsiness be the function given by the formula $t = f(m) = \dfrac{4}{2^m + 2}$, because it changes exponentially and follows the rules state above. Then, **Table 8** shows the probability of the game lasting 5 rounds from varying starting positions and proportions of sober moves allocated to each player. It also shows estimates for expected game times from each starting position where we use

$$E(d) = e_d \cdot \left(\sum_{n=1}^{\mathcal{N}} \left(\prod_{m=1}^{n-1} T_m \right) \right) \mathbb{1} \qquad (27)$$

Table 7. Some survival rates and expected game times on C_6 when players sober up over time.

Measure	Percentage of Sober Moves that are Robber Moves										
$t = \dfrac{4}{m+3}$	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_5(1)$	0.1393	0.1709	0.2023	0.2331	0.2630	0.2917	0.3189	0.3444	0.3681	0.3899	0.4095
$\mathfrak{G}_5(2)$	0.3667	0.4078	0.4484	0.4882	0.5270	0.5646	0.6008	0.6354	0.6684	0.6995	0.7286
$\mathfrak{G}_5(3)$	0.2786	0.3419	0.4047	0.4663	0.5260	0.5833	0.6378	0.6889	0.7363	0.7798	0.8190
$E(1)$	3.029*	3.245*	3.530*	3.922*	4.504*	5.456*	7.281*	11.83*	30.39*	242.8**	∞
$E(2)$	4.586*	4.962*	5.456*	6.140*	7.153*	8.812*	11.99*	19.92*	52.27*	422.9**	∞
$E(3)$	5.059*	5.491*	6.059*	6.845*	8.008*	9.913*	13.56*	22.67*	59.78*	484.7**	∞

*Sum calculated to $\mathcal{N} = 1000$, ** $\mathcal{N} = 3000$.

Table 8. Survival rates and expected duration from different starting states on C_6.

Measure	Percentage of Sober Moves that are Robber Moves										
$t = \dfrac{4}{2^m + 2}$	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_s(1)$	0.0381	0.0895	0.1435	0.1978	0.2505	0.3000	0.3449	0.384	0.4177	0.4445	0.4649
$\mathfrak{G}_s(2)$	0.1926	0.2723	0.3511	0.4275	0.5004	0.5688	0.6317	0.6885	0.7388	0.7820	0.8181
$\mathfrak{G}_s(3)$	0.0763	0.1790	0.2869	0.3956	0.5010	0.6000	0.6899	0.7688	0.8354	0.8890	0.9298
$E(1)$	2.546*	2.757*	3.060*	3.531*	4.329*	5.849*	9.228*	18.68*	57.75**	283.7**	∞
$E(2)$	3.753*	4.119*	4.647*	5.467*	6.858*	9.507*	15.40*	31.90*	100.2**	496.6**	∞
$E(3)$	4.093*	4.513*	5.120*	6.062*	7.659*	10.70*	17.46*	36.36*	114.5**	566.5**	∞

*Sum calculated to $\mathcal{N} = 500$, ** $\mathcal{N} = 1000$.

to approximate the infinite sum in Equation (25). We use either $\mathcal{N} = 500$ or $\mathcal{N} = 1000$ to approximate these values because these values of $\mathcal{N}$ are sufficiently large for their respective proportion of sober moves allocated to the robber to accurately approximate the expected game time to at least five digits.

The expected game time for 70% was also calculated at $\mathcal{N} = 1000$ and found to be the same for at least five digits. However, we are certain that for the expected game time at 90% $\mathcal{N} = 1000$ is not sufficiently large, yet trying to calculate the sum with any larger $\mathcal{N}$ causes the SageMath server to timeout.

10. Modeling the Game Where Tipsiness Is a Function of Distance

Mentioning that it may be more biologically realistic, Harris *et al.* also asked how to model a tipsy cop and drunk robber game where the players' tipsiness is determined by the distance between them ([4], Question 6.6). In this section, we model the tipsy cop and robber game where the players sober up as they get closer to each other. This models the scenario where the cop's ability to track the robber improves the closer they get, and the robber senses that the cop is on his trail so he moves more deliberately. The transition matrix T_δ represents the stochastic matrix, where both the cop and robber sober up as they get closer to each other and get more tipsy as the distance between the two increases. If $\mathfrak{n}$ is the maximum distance the two can be apart on a finite graph (or the distance at which the cop calls off the hunt), then we assume that the function $\delta(d)$ has the following properties:

$$\delta(1) = 0 \tag{28}$$

$$\lim_{d \to \mathfrak{n}} \delta(d) = 1 \tag{29}$$

Additionally, we set the proportion of sober cop moves to be $c = \dfrac{a}{b}(1-t)$ and the proportion of sober robber moves to be $r = \dfrac{b-a}{b}(1-t)$, where a and b are

any desired integers that determine what fraction of all sober moves is assigned to the cop and to the robber.

10.1. Linear Increase of Tipsiness

In this scenario, we choose to use the function $\delta(d) = \dfrac{d-1}{\mathfrak{n}}$, where d is the distance between the cop and the robber when they start the chase and $\mathfrak{n}$ is the maximum distance they can be apart; this value $\mathfrak{n}$ is specified to be either the radius of the graph on finite graphs, or the maximum specified distance before the cop calls off the chase on an infinite graph. The probability of the game lasting at least $\mathbf{M}$ rounds when starting in state d is

$$\mathfrak{G}_M(d) = e_d \cdot T_\delta^{\mathbf{M}} \cdot \mathbb{1}. \tag{30}$$

The expected game time is

$$E(d) = e_d \cdot (I - T_\delta)^{-1} \cdot \mathbb{1}. \tag{31}$$

10.2. Exponential Increase of Tipsiness

Now, we choose to use the function $\delta(d) = \dfrac{1 - 1.2^{1-d}}{1 + 1.2^{1-d}}$, where d is the distance between the cop and the robber when they start the chase. The probability of the game lasting through $\mathbf{M}$ rounds is given by:

$$\mathfrak{G}_M(d) = e_d \cdot T_\delta^{\mathbf{M}} \cdot \mathbb{1} \tag{32}$$

The expected game time is given by:

$$E(d) = e_d \cdot (I - T_\delta)^{-1} \cdot \mathbb{1} \tag{33}$$

We conclude this section with some sample calculations in tables. We have published the accompanying SageMath code for these computations in **Appendix A.5**, so the interested reader can change these calculations to model the game for any values of $\mathbf{n}, m, d, r$ that they choose.

10.3. Numerical Example on Cycle Graph

Given a cycle graph of $\mathbf{n} = 10$ nodes, we will have the possibilities of starting at distances 0 to 5 away with maximum distance $\mathfrak{n} = \mathbf{n}/2 = 5$. The accompanying Markov Chain is (again we assume that the cop always moves and the sober robber chooses not to move if they are furthest away from each other as noted in Section 5 - even nodes cycle graphs) (**Figure 17**).
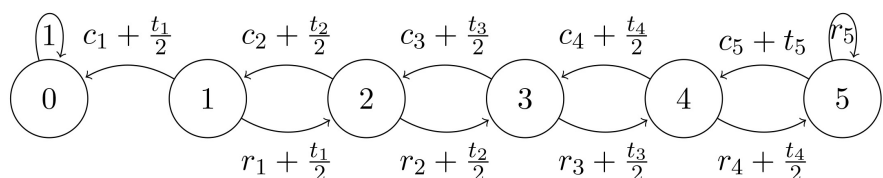

Figure 17. Markov chain for cycle graph C_{10}.

Based on the Markov chain we get the transformation matrix based on distance T_δ

$$T_\delta = \begin{pmatrix} 0 & r_1 + \frac{t_1}{2} & 0 & 0 & 0 \\ c_2 + \frac{t_2}{2} & 0 & r_2 + \frac{t_2}{2} & 0 & 0 \\ 0 & c_3 + \frac{t_3}{2} & 0 & r_3 + \frac{t_3}{2} & 0 \\ 0 & 0 & c_4 + \frac{t_4}{2} & 0 & r_4 + \frac{t_4}{2} \\ 0 & 0 & 0 & c_5 + t_5 & r_5 \end{pmatrix} \tag{34}$$

Note that for each distance d, $r_d + c_d + t_d = 1$. In our model we specify what percentage of all sober moves $(1 - t_d)$ are given to the robber.

The probability of surviving 20 rounds when starting at distance d apart $\mathfrak{G}_{20}(d)$ and the expected duration of the chase $E(d)$ using the linear growth of the tipsiness $t = \delta(d) = \frac{d-1}{n}$ is shown in Table 9.

Similarly, the probability of surviving 20 rounds when starting distance d apart $\mathfrak{G}_{20}(d)$ and the expected duration of the chase $E(d)$ using the exponential growth for tipsiness $t = \delta(d) = \frac{1 - 1.2^{1-d}}{1 + 1.2^{1-d}}$ are shown in Table 10.

Based on the results in the tables the chase is longer if the tipsiness is increasing exponentially and the percentage of all sober moves $(1 - t_d)$ are given to the robber is less than 50%. In the case that the percentage of all sober moves $(1 - t_d)$ are given to the robber is greater than 50% the chase is longer if the tipsiness is increasing linearly. Assuming the robber is given the choice, the robber should pick a strategy based on the percentage of sober moves assigned to him-if he gets to move more than 50% of the sober moves his tipsiness should increase

Table 9. Sample calculations on C_{10} when tipsiness increases linearly with distance.

Measure	Percentage of Sober Moves that are Robber Moves										
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_{20}(1)$	0	0.0001	0.0024	0.015	0.057	0.148	0.299	0.491	0.689	0.864	1
$\mathfrak{G}_{20}(2)$	6E−5	0.001	0.009	0.041	0.125	0.277	0.48	0.69	0.86	0.958	1
$\mathfrak{G}_{20}(3)$	0.0005	0.0049	0.025	0.085	0.208	0.391	0.599	0.784	0.912	0.978	1
$\mathfrak{G}_{20}(4)$	0.0011	0.0088	0.039	0.118	0.261	0.454	0.657	0.824	0.932	0.984	1
$\mathfrak{G}_{20}(5)$	0.0024	0.0139	0.053	0.144	0.295	0.489	0.683	0.839	0.938	0.985	1
$E(1)$	1	1.29	1.81	2.78	4.77	9.25	20.58	53.89	177.7	914.8	∞
$E(2)$	2.30	2.97	4.06	5.93	9.41	16.5	32.64	75.55	220.8	1015	∞
$E(3)$	4.02	5.04	6.59	9.1	13.45	21.75	39.64	85.2	234.7	1036	∞
$E(4)$	5.87	7.09	8.88	11.65	16.32	25.0	43.36	89.49	239.7	1042	∞
$E(5)$	6.87	8.14	9.97	12.79	17.51	26.25	44.68	90.88	241.2	1044	∞

Table 10. Sample calculations on C_{10} when tipsiness grows exponentially with distance.

Measure	Percentage of Sober Moves that are Robber Moves										
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_{20}(1)$	0	1.2E−5	0.0008	0.0093	0.049	0.152	0.325	0.532	0.726	0.881	1
$\mathfrak{G}_{20}(2)$	4E−8	6.1E−5	0.0025	0.025	0.109	0.287	0.529	0.755	0.906	0.978	1
$\mathfrak{G}_{20}(3)$	9E−7	0.0003	0.0072	0.0513	0.181	0.404	0.656	0.852	0.957	0.993	1
$\mathfrak{G}_{20}(4)$	1.9E−6	0.0006	0.012	0.0741	0.234	0.477	0.722	0.891	0.972	0.997	1
$\mathfrak{G}_{20}(5)$	8.2E−6	0.0012	0.018	0.0947	0.269	0.515	0.749	0.904	0.976	0.997	1
$E(1)$	1	1.27	1.72	2.59	4.52	9.58	25.7	90.6	466.9	4825	∞
$E(2)$	2.1	2.67	3.59	5.29	8.79	17.16	41.1	128	582.3	5360	∞
$E(3)$	3.33	4.19	5.56	7.93	12.52	22.74	50.1	144	615	5443	∞
$E(4)$	4.64	5.74	7.41	10.18	15.31	26.31	54.8	150.6	624.8	5459	∞
$E(5)$	5.64	6.82	8.58	11.46	16.73	27.89	56.6	152.7	627.2	5462	∞

linearly, otherwise (he gets to move 50% or less of all sober moves) he has a better chance of surviving if the tipsiness changes exponentially.

10.4. Infinite Regular Trees

If we play the game on an infinite regular tree, and the cop calls off the hunt if the robber reaches a distance of $\mathfrak{n}$ nodes away from the cop, then the matrix P is an $(\mathfrak{n}+1 \times \mathfrak{n}+1)$ tridiagonal matrix with $P_{1,1} = 1$, $P_{\mathfrak{n}+1,\mathfrak{n}+1} = 1$ and the upper and lower diagonal entries are $P_{k+1,k} = c_{k+1} + \dfrac{t_{k+1}}{\Delta}$ for $1 \le k \le \mathfrak{n}-1$, and

$P_{k+1,k+2} = t_{k+1}\dfrac{\Delta-1}{\Delta} + r_{k+1}$ for $1 \le k \le \mathfrak{n}-1$, and finally $P_{i,j} = 0$ otherwise. To obtain the matrix T we remove from P the first and last rows and columns corresponding to both absorbing states.

For example, on an infinite regular tree of degree Δ with maximum specified distance $\mathfrak{n} = 5$, the accompanying Markov Chain is in **Figure 18**. Based on the Markov chain we get the transformation matrix based on distance T_{δ}

$$T_{\delta} = \begin{pmatrix} 0 & r_1 + t_1\dfrac{\Delta-1}{\Delta} & 0 & 0 \\ c_2 + \dfrac{t_2}{\Delta} & 0 & r_2 + t_2\dfrac{\Delta-1}{\Delta} & 0 \\ 0 & c_3 + \dfrac{t_3}{\Delta} & 0 & r_3 + t_3\dfrac{\Delta-1}{\Delta} \\ 0 & 0 & c_4 + \dfrac{t_4}{\Delta} & 0 \end{pmatrix}. \tag{35}$$

Note that in this case we remove states 0 and 5 because they are absorbing. Similarly in the numerical example on cycle graphs, we specify what percentage of all sober moves $(1 - t_d)$ are given to the robber.

10.5. Numerical Example on Infinite Regular Tree with $\Delta = 4$ and Maximum Distance $\mathfrak{n} = 10$

The probability that the game continues for $\mathbf{M} = 30$ rounds when starting distance d apart $\mathfrak{G}_{30}(d)$ and the expected duration of the chase $E(d)$ when $\Delta = 4$ when tipsiness increases linearly $t = \delta(d) = \dfrac{d-1}{\mathfrak{n}}$ is shown in Table 11. Note that we assume the cop calls off the chase if the distance reaches $\mathfrak{n} = 10$.

Table 12 shows the probability of a game lasting 30 rounds when starting at distance d apart $\mathfrak{G}_{30}(d)$ and the expected duration of the chase $E(d)$ when the game takes place on a regular tree of degree $\Delta = 4$ and tipsiness is modeled
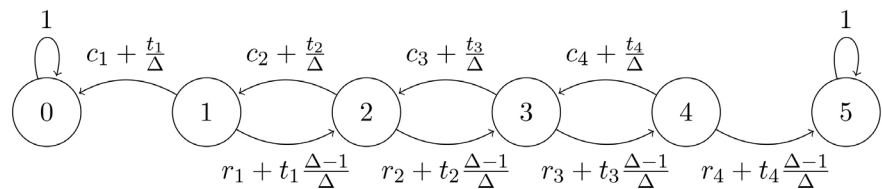

Figure 18. Markov chain for regular tree with maximum distance of $\mathfrak{n} = 5$.

Table 11. Numerical results on infinite regular tree when tipsiness increases linearly with distance.

Measure					Percentage of Sober Moves that are Robber Moves						
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_{30}(1)$	0	4.5E−5	0.0008	0.006	0.022	0.045	0.051	0.033	0.012	0.002	0.0002
$\mathfrak{G}_{30}(2)$	1.4E−5	0.0003	0.0031	0.016	0.046	0.073	0.067	0.034	0.009	0.0013	0.0001
$\mathfrak{G}_{30}(3)$	0.0002	0.0017	0.0099	0.035	0.077	0.099	0.077	0.034	0.008	0.0012	9.7E−5
$\mathfrak{G}_{30}(4)$	0.0007	0.0042	0.0174	0.048	0.083	0.088	0.057	0.021	0.004	0.0005	3.9E−5
$\mathfrak{G}_{30}(5)$	0.0026	0.0102	0.0303	0.063	0.089	0.08	0.046	0.016	0.003	0.0004	3.5E−5
$\mathfrak{G}_{30}(6)$	0.0041	0.0128	0.031	0.054	0.065	0.051	0.025	0.008	0.001	0.0002	1.3E−5
$\mathfrak{G}_{30}(7)$	0.0065	0.0161	0.0318	0.046	0.047	0.033	0.015	0.005	0.0009	0.0001	9.4E−6
$\mathfrak{G}_{30}(8)$	0.0045	0.0102	0.0182	0.024	0.022	0.014	0.006	0.0016	0.0003	3.3E−5	2.6E−6
$\mathfrak{G}_{30}(9)$	0.0027	0.0054	0.0086	0.01	0.009	0.005	0.002	0.0006	0.0001	1.4E−5	1.2E−6
$E(1)$	1	1.29	1.82	2.86	4.75	7.3	9.58	11	11.5	11.6	11.4
$E(2)$	2.2	2.88	4.09	6.21	9.38	12.6	14.3	14.2	13.2	11.8	10.4
$E(3)$	3.72	4.9	6.8	9.68	13.1	15.5	15.6	14.2	12.3	10.7	9.39
$E(4)$	5.64	7.29	9.62	12.5	15.15	16	14.8	12.8	10.9	9.38	8.28
$E(5)$	7.81	9.61	11.77	13.9	15.19	14.7	12.8	10.8	9.17	7.99	7.11
$E(6)$	9.54	10.97	12.37	13.4	13.4	12.1	10.3	8.63	7.41	6.53	5.87
$E(7)$	9.75	10.44	10.92	10.9	10.3	8.99	7.58	6.44	5.61	5.01	4.55
$E(8)$	7.8	7.86	7.75	7.37	6.67	5.76	4.91	4.25	3.78	3.42	3.14
$E(9)$	4.11	3.99	3.79	3.51	3.14	2.73	2.37	2.11	1.91	1.75	1.63

Table 12. Some survival times and expected game durations on infinite regular tree of degree 4 when tipsiness increases exponentially with distance.

Measure	Percentage of Sober Moves that are Robber Moves										
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
$\mathfrak{G}_{30}(1)$	0	0.0004	0.0016	0.0042	0.0085	0.014	0.019	0.021	0.02	0.017	0.014
$\mathfrak{G}_{30}(2)$	0.0014	0.0029	0.0056	0.0098	0.015	0.02	0.022	0.021	0.017	0.013	0.0089
$\mathfrak{G}_{30}(3)$	0.0055	0.0082	0.012	0.017	0.021	0.024	0.024	0.021	0.016	0.012	0.0085
$\mathfrak{G}_{30}(4)$	0.0066	0.0084	0.0107	0.013	0.015	0.015	0.014	0.012	0.009	0.007	0.0048
$\mathfrak{G}_{30}(5)$	0.0073	0.0086	0.0099	0.011	0.012	0.012	0.01	0.0087	0.007	0.005	0.0039
$\mathfrak{G}_{30}(6)$	0.0041	0.0046	0.005	0.0054	0.006	0.005	0.005	0.0039	0.003	0.002	0.0018
$\mathfrak{G}_{30}(7)$	0.0027	0.003	0.003	0.0033	0.003	0.003	0.0027	0.0023	0.0018	0.0015	0.0012
$\mathfrak{G}_{30}(8)$	0.001	0.001	0.001	0.0012	0.0011	0.001	0.0009	0.0008	0.0007	0.0005	0.0004
$\mathfrak{G}_{30}(9)$	0.004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0003	0.0003	0.0002	0.0002	0.0002
$E(1)$	1	1.47	2.22	3.32	4.8	6.6	8.55	10.5	12.1	13.5	14.5
$E(2)$	3.6	4.7	6.09	7.73	9.5	11.2	12.6	13.5	13.9	13.9	13.5
$E(3)$	7.41	8.53	9.71	10.9	12	12.8	13.2	13.3	13.1	12.8	12.4
$E(4)$	9.85	10.5	11	11.5	11.8	12.1	12.1	11.9	11.6	11.3	11
$E(5)$	9.88	10.1	10.2	10.3	10.4	10.3	10.2	10	9.81	9.6	9.4
$E(6)$	8.38	8.4	8.4	8.38	8.33	8.26	8.16	8.04	7.91	7.79	7.67
$E(7)$	6.33	6.31	6.28	6.25	6.21	6.15	6.09	6.02	5.96	5.89	5.83
$E(8)$	4.18	4.17	4.15	4.13	4.1	4.07	4.04	4.01	3.98	3.95	3.92
$E(9)$	2.07	2.06	2.05	2.05	2.04	2.03	2.02	2	1.99	1.98	1.97

using the exponential function $t_d = \delta(d) = \dfrac{1 - 1.2^{1-d}}{1 + 1.2^{1-d}}$. Again we assume the cop calls off the chase if the distance reaches $\mathfrak{n} = 10$.

Based on the results in the tables above, the robber should pick the strategy where the tipsiness changes exponentially, since it gives him a higher chance of survival and the chase lasts longer.

11. Future Directions and Open Questions

We end this paper by pointing out a few possible areas of future study and posing a few open questions: modeling the game on infinite grids is considerably more difficult than on infinite regular trees because there are multiple states for a given distance, and it is not immediately clear what strategy the cop should use to decrease distance or the robber should use to increase distance.

1) On an infinite grid, what are the optimal strategies for the cop and robber?

2) Who is more likely to win on an infinite grid, if both players employ their optimal strategies?

3) How will the game change if the cop and the robber do not take turns, but instead move simultaneously?

Acknowledgements

We would like to thank Drs. Pamela Harris, Florian Lehner, and Alicia Prieto-Langarica for many helpful and inspiring conversations and suggestions regarding this project. We also thank Drs. Katie Johnson and Shaun Sullivan for reading and providing helpful feedback regarding earlier drafts of this manuscript. This collaboration was supported by the FGCU Seidler Collaboration Fellowship.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Quilliot, A. (1986) Some Results about Pursuit Games on Metric Spaces Obtained through Graph Theory Techniques. *European Journal of Combinatorics*, **7**, 55-66. https://doi.org/10.1016/S0195-6698(86)80017-8

[2] Nowakowski, R. and Winkler, P. (1983) Vertex-to-Vertex Pursuit in a Graph. *Discrete Mathematics*, **43**, 235-239. https://doi.org/10.1016/0012-365X(83)90160-7

[3] Komarov, N. and Winkler, P. (2014) Catching the Drunk Robber on a Graph. *Electronic Journal of Combinatorics*, **21**, arXiv: 1305.4559. https://doi.org/10.37236/3398

[4] Pamela, H., Erik, I., Alicia, P.-L., Rade, S. and Shaun, S. (2020) Tipsy Cop and Drunken Robber: A Variant of the Cop and Robber Game on Graphs. https://arxiv.org/pdf/2004.00606.pdf

[5] Immiflex Immune System (2013) Neutrophil Phagocytosis—White Blood Cell Eats Staphylococcus Aureus Bacteria. https://www.youtube.com/watch?v=Z_mXDvZQ6dU

[6] El-Shehawey, M.A. (2009) On the Gambler's Ruin Problem for a Finite Markov Chain. *Statistics & Probability Letters*, **79**, 1590-1595. https://doi.org/10.1016/j.spl.2009.03.029

[7] Dunbar, S. (2019) Mathematical Modeling in Economics and Finance: Probability, Stochastic Processes, and Differential Equations. *AMS/MAA Textbooks*, **49**, 78-84.

Appendix

A. SageMath Code

The code described in the manuscript is provided in this appendix.

A.1. SageMath Code for Petersen Graph

```
var ('c,t,M,i,p')
###################################################
def  check_rt(r,t):
    check=1
    if r<0:
        check=0
        print('r cannot be negative')
    if t<0:
        check=0
        print('t cannot be negative')
    if t+r>1:
        check=0
        print('r+t cannot be greater than 1')
    return check
###################################################
def define_P(n=4): #default n=4 can be changed later
    L=[]
    for   i in range(n+1):
        L.append([])
        for   j in range(n+1):
            L[i].append(0)
    L[0][0]=1
    L[−1][−1]=1
    for   i in range(1,n):
        L[i][i+1]=r+(2*t/3)
    for   i in range(1,n):
        L[i][i−1] =c+t/3
    L[n][n]=r+(2*t/3)
    L[n][n−1]=c+t/3
    M=matrix(L)
    return M
###################################################
def  check_define_P(n=4):
    if check_rt(r,t) ==1:
    return define_P(n)
###################################################
M=7 #number of rounds
r=.30
```

```
t=.40
c=1−t−r
n=10 #default n=4 can be changed here
check_define_P(n)
##################################################
def define_T(n):
    P=define_P(n)
    T=P.submatrix(1,1,n,n)
    return T
##################################################
P=define_P(n)
print(P)
print()
T=define_T(n)
print(T)
print()
I = identity_matrix(n)
print(I)
one=[]
for i in range(n):
    one.append(1)
N=vector(one)
print(N)
print('c=',c,'r=',r)
for d in range(1,n+1):
    e_d = I.column(d−1)
    print('d=',d,'n=',n,'M=',M)
    G = e_d*(T^M)*N
    print('Surv. prob. of M rounds from')
    print('distance', d, 'is', G)
    E = e_d*((I−T)^(−1))*N
    print('Expected number of rounds from')
    print('distance', d, 'is', E)
    print()
```

A.2. SageMath Code for 7 × 7 Toroidal Grid

```
var('r,t,c')
r=0.4
# Proportion of moves that are sober robber moves.
c=0.3
# Proportion of moves that are sober cop moves.
t=0.3
# Proportion of moves that are tipsy moves by
# either player.
```

```
M=50
# Number of rounds you want the robber to survive.
###################################################
if c+r+t != 1:                                     #
    print('Make sure your probabilities sum to 1:') #
    print('Sum = ' +'t+r+c' + ' = 1 ?')            #
###################################################
else:
    v=vector([1,1,1,1,1,1,1,1,1])
    a=vector([1,0,0,0,0,0,0,0,0]) #starting state (3,3)
    b=vector([0,1,0,0,0,0,0,0,0]) #starting state (3,2)
    d=vector([0,0,1,0,0,0,0,0,0]) #starting state (3,1)
    e=vector([0,0,0,1,0,0,0,0,0]) #starting state (3,0)
    f=vector([0,0,0,0,1,0,0,0,0]) #starting state (2,2)
    g=vector([0,0,0,0,0,1,0,0,0]) #starting state (2,1)
    h=vector([0,0,0,0,0,0,1,0,0]) #starting state (2,0)
    i=vector([0,0,0,0,0,0,0,1,0]) #starting state (1,1)
    j=vector([0,0,0,0,0,0,0,0,1]) #starting state (1,0)
    T=matrix([[r+t/2,c+t/2,0,0,0,0,0,0,0,0],
            [r+t/4,t/4,t/4,0,c+t/4,0,0,0,0],
            [0,r+t/4,t/4,t/4,0,c+t/4,0,0,0],
            [0,0,r+t/2,t/4,0,0,c+t/4,0,0],
            [0,r+t/2,0,0,0,c+t/2,0,0,0],
            [0,0,r+t/4,0,t/4,0,t/4,c+t/4,0],
            [0,0,0,r+t/4,0,t/2,0,0,c+t/4],
            [0,0,0,0,0,r+t/2,0,0,c+t/2],
            [0,0,0,0,0,0,r+t/4,t/2,0]])
    G=vector((T^M)*v)
    E=vector((1−T).inverse()*v)
    print('G_'+'M'+'(3,3)=',G*a)
    print('G_'+'M'+'(3,2)=',G*b)
    print('G_'+'M'+'(3,1)=',G*d)
    print('G_'+'M'+'(3,0)=',G*e)
    print('G_'+'M'+'(2,2)=',G*f)
    print('G_'+'M'+'(2,1)=',G*g)
    print('G_'+'M'+'(2,0)=',G*h)
    print('G_'+'M'+'(1,1)=',G*i)
    print('G_'+'M'+'(1,0)=',G*j)
    print('E(3,3)=',E*a)
    print('E(3,2)=',E*b)
    print('E(3,1)=',E*d)
    print('E(3,0)=',E*e)
    print('E(2,2)=',E*f)
```

88

```
print('E(2,1)=',E*g)
print('E(2,0)=',E*h)
print('E(1,1)=',E*i)
print('E(1,0)=',E*j)
```

A.3. SageMath Code for Cycle Graphs Sobering up Over Time

```
from sage.calculus.calculus import symbolic_sum
var ('t,M,N,i,p')
M=5 #number of rounds you want the robber to survive
N=1000 #Nth partial sum of infinite sum.
# Higher is more accurate but takes longer.
k=9 #number of nodes in cycle graph
p=0.51 #proportion of sober moves allocated to robber
##################################################
# if p > 0.8 then N should be at at least 3000.        #
# otherwise N=1000 is sufficient.                      #
##################################################
If   k%2==0:
        n=(k/2)
else:n=(k−1)/2
def define_P(n,p,t):
    L=[]
    for   i in range(n+1):
        L.append([])
        for   j in range(n+1):
            L[i].append(0)
    L[0][0]=1
    L[−1][−1]=1
    for   i in range(1,n):
        L[i][i+1]=p*(1−t) +t/2
        # transition probability to increase distance
        L[i][i−1]=(1−p)*(1−t) + t/2
        # transition probability to decrease distance
    if k%2==0: #i f cycgraph i s even
        L[n][n]= p*(1−t)
        L[n][n−1]=(1−p)*(1−t)+t
    else: #if cycle graph is odd
        L[n][n]=p*(1−t)+t/2
        L[n][n−1]=(1−p)*(1−t)+t/2
    M=matrix(L)
    return M
##################################################
def  define_T(n,p,t):
    P=define_P(n,p,t)
```

```
            T=P.submatrix(1,1,n,n)
            return T
####################################################
# ouptut: n length vector of all ones
####################################################
def one_vector(n):
    L=[]
    for   i in range(n):
        L.append(1)
    return vector(L)
####################################################
def position_vector(n,d):
    L=[]
    for   i in range(n):
        L.append(0)
    L[d]=1
    return vector(L)
####################################################
T=define_T(n,p,t)
#print T
#Uncomment the line above to see Transition Matrix
print('Calculating...')
x=vector(prod(define_T(n,p,4/(3+m))
for m in range (1,M+1))*one_vector(n))
y=vector(sum(prod(define_T(n,p,4/(3+m))
for m in range (1,q))
for q in range (1,N+1))*one_vector(n))
print('done:')
for  i in range (0,n):
    print('G_'+'M'+'('+'i+1'+')=',x*position_vector(n,i))
    print('E('+'i+1'+')=',y*position_vector(n,i))
####################################################
```

A.4. SageMath Code for Friendship Graphs

```
var('r,tr,tc,n,c')
tr=.4 #Probability of a tipsy robber move
tc=.4 #Probability of a tipsy cop move
r=.1 #Probability of a sober robber move
c=.1 #Probability of a sober cop move
n=6 #Number of triangles
M=5 #Number of rounds the game lasts
####################################################
if   tc+tr+r+c != 1:                              #
    print('Make sure your probabilities sum to 1:')   #
```

```
            print('Sum = '+'tc+tr+r+c' + ' = 1 ?')                    #
##################################################
else:
    T=matrix([[r+tc/2+tr/2,c+tc/2,tr/2,0],
              [tc*((n−1)/n),r+tr/2,0,tc/(2*n)],
              [r+tr*((n−1)/n),0,tc/2,tr/(2*n)],
              [0,tc/2,r+tr/2,0]])
    v=vector([1,1,1,1])
    a=vector([1,0,0,0]) #starting state 2
    b=vector([0,1,0,0]) #starting state 1cc
    c=vector([0,0,1,0]) #starting state 1rc
    d=vector([0,0,0,1]) #starting state 1e
    E=vector((matrix.identity(4)−T).inverse()*v)
##################################################
    print('G_'+'M'+'(2)=',a*T^M*v)
    print('G_'+'M'+'(1cc)=',b*T^M*v)
    print('G_'+'M'+'(1rc)=',c*T^M*v)
    print('G_'+'M'+'(1e)=',d*T^M*v)
    print('E(2)=',E*a)
    print('E(1cc)=',E*b)
    print('E(1rc)=',E*c)
    print('E(1e)=',E*d)
```

A.5. SageMath Code for Regular Tree with Exponential Change of Tipsiness as a Function of Distance

```
var('c,t,M,i,p')
M=30 #number of rounds you want the robber to survive
#c=1−t−c #proportion of sober cop moves
#r—proportion of sober robber moves
#t—proportion of tipsy moves, changes with distance
#D—distance between cop and robber when they start
n=10#maximum distance between cop and robber
delta=4
p=1 #proportion of sober moves allocated to robber
##################################################
# input: n is maximum distance between cop and robber
# before calling off chase # input: p is proportion of sober moves allocated to
# robber
# output: transition matrix (including 2 absorbing
# states)
##################################################
def  define_P(n,p):
    L=[]
    for   i in range(n+1):
```

```
                    L.append([])
                    for  j in range(n+1):
                        L[i].append(0)
        T=[0]
        R=[0]
        C=[0]
        L[0][0]=1
        L[−1][−1]=1
        for  D in range(1,n+1):
            s=1 − (1.2)^(1−D) q=1 + (1.2)^(1−D)
            T.append(s/q) #tipsiness based on distance
            R.append(p*(1−T[D]))
            # Percentage of sober moves are robber moves
            # as a function of t based on distance
            C.append(1−T[D]−R[D])
        print('T=',T)
        print('R=',R)
        print('C=',C)
        for  i in range(1,n):
            L[i][i+1]=R[i]+(T[i]*((delta − 1)/delta))
            # transition probability to increase distance
            L[i][i−1] = C[i]+T[i]*(1/delta)
            # transition probability to decrease distance
        L[n][n−1]=0
        L[n][n]=1
        M=matrix(L)
        return M
####################################################
####################################################
# input: n is maximum distance between cop and robber
# before calling off chase
# input: p is proportion of sober moves allocated to
# robber
# output: transition matrix (without absorbing states)
###################################################
def  define_T(n,p):
    P=define_P(n,p)
    T=P.submatrix(1,1,n−1,n−1)
    return T
####################################################
####################################################
# input: n the maximum distance between cop and robber
# before calling off chase
```

```
# ouptut: n−2 length vector of all ones
####################################################
def  one_vector(n):
    L=[]
    for   i in range(n−1):
        L.append(1)
    return vector(L)
####################################################
####################################################
# input: n the maximum distance between cop and robber
# before calling off chase
# input: d such that d+1 is starting distance
# between cop and robber
####################################################
def  position_vector(n,d):
    L=[]
    for   i in range(n−1):
        L.append(0)
    L[d]=1
    return vector(L)
####################################################
P=define_P(n,p)
T=define_T(n,p)
one= one_vector(n)
print() print('G_d is probability of surviving', M,)
print('rounds starting at distance d.')
print('E_d is expected game length from distance d:')
print()
for   d in range(n−1):
    print('starting distance d=', d+1)
    e=position_vector(n,d)
    G_d=e*(T^M)*one
    print('G_d=', float(G_d))
    E_d=e*((1−T).inverse())*one
    print('E_d=', float(E_d))
    print()
```