

Graph Fleas

Ting Gong, Nick VanderLaan, and Nikhil Shankar

June 11, 2018

Abstract

On a connected graph, we consider the motion of a “flea” under rules assigned to each vertex. Given an initial set of states and rules on a graph, will a flea ever return to its initial position, with all states in their initial conditions? What changes with two fleas?

1 Introduction

A flea lives on the graph, hopping from vertex to vertex along the edges. A state, assigned to each vertex determines the edge along which the flea leaves a vertex, depending on the edge along which it entered the vertex. The flea, on the other hand, changes the state of a vertex as it is leaving it, according to some rule applicable to that vertex.

We will discuss the motion of fleas moving on a polygon with vertices occupying one of two “states, ” taken from a subgroup of the group of permutations of the edges having that vertex as an endpoint, which in this case is S_2 , the symmetric group of two elements. We can think of a polygon with n vertices as the n -cycle, or the connected graph with n nodes and n edges, where each node is connected to exactly two other nodes by edges. We assign to each node a set of states given by a subgroup of $S_2 = \langle 1, s : s^2 = 1 \rangle$.

For this problem, we assign to each vertex of a graph the state $s \in S_2$ initially. In this way we associate to each vertex of the graph an ordered triple $(v_i, x, n) \in V \times S_2 \times \mathbb{Z}_{\geq 0}$ where $v_i \in V$ the set of vertices. In this way a flea coming in by one edge leaves according to some rule dependant on the state of the vertex it is leaving, and a flea leaving a vertex acts on the state of the vertex (v_i, x) by means of $f \cdot (v_i, x, n) \rightsquigarrow (v_i, sx, n)$ with f an action on S_2 . We adopt the notation of $1 = \times, s = \checkmark$, which will be used throughout the paper.

From there we explore the behaviour of two fleas on an n -cycle, and a “Cops and Robbers” game that could be played out by two fleas, both cases yeilding complex behaviour.

2 A Flea Walk on a Triangle

We will begin by exploring the example of one flea in the case $n = 3$ (a triangle) with all vertices initialized to the “✓” state. The “✓” will not change the direction the flea is moving, however the “×” state will. For example, if the flea is moving clockwise, and lands on a “✓”, its next step will still be in the clockwise direction; however if the flea instead lands on a “×”, its next step will be in the counter-clockwise direction. When the flea is leaving a vertex, the state of that vertex is changed to the opposite state. If the vertices are labeled $v_0, v_1, v_2 \in V$, and the flea begins at v_1 moving towards v_2 , its motion is described by the following table (the position of the flea is denoted by a dot):

# steps	Vertex			Direction of next hop
	v_0	v_1	v_2	
0	✓	✓	✓	→
1	×	✓	✓	→
2	×	×	✓	→
3	×	×	×	→
4	✓	×	×	←
5	✓	×	✓	←
6	×	×	✓	→
7	×	✓	✓	←
8	✓	✓	✓	→

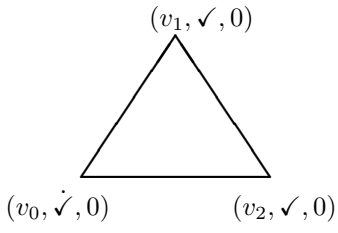


Table 1: A semi-cycle of a triangle

Notice that after 8 steps, the vertex states have all returned to their initial condition.

Definition. Let φ denote the number of moves a flea takes in order to return the state on every vertex to its initial condition, ignoring the final position of the flea. This will be called a **semi-cycle**.

Further notice that, in this example, the flea is not on its initial vertex, instead it occupies v_2 . Finally, notice that after one semi-cycle, the flea is still moving the the same direction it started moving. So, by symmetry, we expect that after three total semi-cycles, the flea will be at its starting position and all vertices will be in the “✓” state.

Definition. The number of moves a flea must make to return to its initial position with all vertices in their initial conditions will be referred to as a **complete cycle**, and denoted Ω .

After φ many moves, we can think of the resultant graph as a rotation of the original graph, due to the flea’s location at a vertex that is not necessarily the same as the vertex she started on.

Definition. We introduce the number k to be the least positive integer such that $\Omega = \varphi \cdot k$. In the case of the triangle, $k = 3$.

3 Extention to n-sided Polygon

Solving a flea walk by hand quickly becomes tedious and mistake-prone. To attack this problem we used computational techniques to learn about specific cases. The following is output from our program up to the 12-cycle. This data gave us interesting information and the ability to form and pursue the conjectures we prove in the following section. For example, it seems that given vertices labeled v_0, \dots, v_{n-1} labeled in the clockwise direction, for a flea starting on v_0 , initially traveling clockwise (denoted $+1$), the position of a flea after a semi-cycle is always: v_{n-2} . We also observe the semi-cycle length $\varphi = 4n - 4$, and the direction of the flea does not change after a semi-cycle.

n	φ	Location after φ	Direction after φ
3	8	v_1	$+1$
4	12	v_2	$+1$
5	16	v_3	$+1$
6	20	v_4	$+1$
7	24	v_5	$+1$
8	28	v_6	$+1$
9	32	v_7	$+1$
10	36	v_8	$+1$
11	40	v_9	$+1$
12	44	v_{10}	$+1$

Table 2: Semi-cycle data for small n

3.1 General Theory

To extend these ideas to an n -sided polygon, we would like to introduce a mathematical tool. For a n -gon with initial conditions as before, after n moves we find ourselves in the configuration of the table below. From there we count the number of steps to resolve the rightmost n entries to checkmarks by way of traversing the graph. Table 3 (below) depicts the semicylce of a 4-cycle, and we can see that it also contains the semi-cycle for the 3-gon at row $n + 5$.

It turns out that after the shortest walk that reverts the nodes to all checks will not return the flea to its initial position. One then asks how many such walks must a flea take to be ensured to return home, leaving the initial state of the graph as it was before she left. The above table indicates that the flea effectively moves $n - 2$ places every semi-cylce (despite taking φ steps), equivalent to taking 2 steps in the opposite direction. Two questions remain.

Move	v_{n-1}	v_0	v_1	v_2	\dots	v_{n-2}
0	✓	✓	✓	✓	\dots	✓
\vdots	\vdots			\vdots		\vdots
n	×	×	×	×	\dots	×
$n+1$	×	✓	×	×	\dots	×
$n+2$	✓	✓	×	×	\dots	×
$n+3$	✓	×	×	×	\dots	×
$n+4$	✓	×	✓	×	\dots	×
$n+5$	✓	✓	✓	×	\dots	×
$n+6$	✓	✓	×	×	\dots	×
$n+7$	✓	✓	×	✓	\dots	×
$n+8$	✓	✓	✓	✓	\dots	×
\vdots				\vdots		\ddots

Table 3: Flea motion for a general n

1. How many moves must a flea make to complete a semi-cycle?
2. How many semi-cycles must a flea complete to return to her initial position?

3.2 Number of Semi-cycles in a Complete Cycle

An answer to the second question is given by the following lemma:

Lemma 1. $(n-2)k \equiv 0 \pmod{n} \iff k = n \text{ if } n \text{ odd, or } k = \frac{n}{2} \text{ for } n \text{ even, where } n \geq 3. \text{ We are interested in the shortest semi-cycle, so we add the restriction } 0 < k \leq n.$

Proof. The position of a flea after traversing a semi-cycle is effectively $n-2$ steps away from where she began, so we calculate the number of trips of length $n-2$ that must be taken to return to the initial position.

We want the minimal $k \geq 1$ such that $(n-2)k \equiv 0 \pmod{n}$

$$(n-2)k \equiv nk - 2k \pmod{n} \quad (1)$$

$$\equiv -2k \pmod{n} \quad (2)$$

$$\implies -2k \equiv 0 \pmod{n} \quad (3)$$

$$\therefore 2k \equiv 0 \pmod{n} \quad (4)$$

and indeed landing at the $n-1$ st position traveling clockwise is equivalent to landing at the second position moving in the reverse direction initially. Now, note that $2k \equiv 0 \pmod{n} \iff n|2k$ and that $0 < k \leq n$. This gives the following possible candidates, $2k = n, 2n \implies k = \frac{n}{2}, n$. Since we want the smallest $k \in \mathbb{N}$ which satisfies the above relationship, it is clear that when n is odd $k = n$ and when n is even $k = \frac{n}{2}$. \square

3.3 Length of Semi-cycle

To answer the question how long a semi-cycle φ is, we offer the following lemma:

Lemma 2. *The length of a semi-cycle, $\varphi = 4n - 4$*

Proof. We will refer to the table of solving an n -gon as we count the number of steps. There are an initial n steps as the flea makes the first trip around the n -gon with no \times 's in its way.

Then notice, the flea is at the center of a trio of \times 's, and in the following six moves it changes two of the three \times s into the \checkmark states and moves to a new trio to the right of these \checkmark states. In those six steps the flea did not leave the trio of \times 's, and thus by symmetry the flea will continue to build two \checkmark states every six moves as it travels rightwards.

Case 1: If n is odd, then $n = 2m + 1$ and the pattern of creating two \checkmark every six moves will continue for the next $6(m - 1)$ steps and the flea will reach a position of

$$\checkmark \dots \checkmark \times \dot{\times} \times$$

Where, since the flea came from the \times to its left, the semi-cycle will be over in five more moves. So in this case there were the initial n steps, the proceeding $6(m - 1)$ steps, and then the final five steps:

$$n + 6(m - 1) + 5 = 8m = 4n - 4$$

Case 2: If n is even, then $n = 2m$ and the pattern of creating two \checkmark every six moves will continue for for the next $6(m - 1)$ moves. This gives a position of

$$\checkmark \dots \checkmark \checkmark \times \dot{\times}$$

At which point in two moves the flea will complete a semi-cycle. In a similar fashion, this implies the number of steps are:

$$n + 6(m - 1) + 2 = 8m - 4 = 4n - 4$$

This proves our claim that the length of the semi-cycle is $\varphi = 4n - 4$. \square

3.4 Direction of Flea at End of Semi-cycle

Lemma 3. *On an n -cycle with the usual initial configuration of every vertex occupying state \checkmark , the direction the flea is traveling at the beginning of a semi-cycle is the same as it is at the end of a semi-cycle.*

Proof. It suffices to show that there are an even number of $\dot{\times}$ within each semi-cycle. Referencing Table 3, we need to count the number of $\dot{\times}$ at which the flea's direction reverses. So fix $n \geq 3$. We will encounter 2 $\dot{\times}$'s in the first $n + 2$ moves. From there on, every 3 moves we encounter exactly 2 $\dot{\times}$'s. And

$$4n - 4 - (n + 2) = 3n - 6 = (n - 2) \cdot 3$$

So the number of $\dot{\times}$ we encounter is given by

$$\text{number of } \dot{\times} = 2 + 2(n - 2)$$

which is even, therefore after completing a semi-cycle, the flea is traveling in the same direction it began. \square

4 Two Fleas on an n -gon

One might ask how the length of a semi-cycle differs if the graph starts with 2 fleas, and one flea moves, changing the graph, after which the other flea moves. This turns out to be a complex problem.

4.1 Fleas Positioned Adjacently on a Polygon

We begin by examining the first semicycle on various n -gon. Initially we start with a n -cycle with vertices labeled v_0, \dots, v_{n-1} in the clockwise direction, placing the fleas on vertices v_0 and v_1 , with every vertex occupying the initial state of \checkmark . We record the number of moves that must be made in order to return the graph to a state of all \checkmark as it was initially in.

Remark. So as to keep numbers smaller, a move refers to one motion of each flea.

n			3	4	5	6	7	8
φ			5	8	14	20	56	32
n	9	10	11	12	13	14	15	16
φ	38	44	260	56	62	68	936	80
n	17	18	19	20	21	22	23	24
φ	86	92	794	104	110	116	7648	128
n	25	26	27	28	29	30	31	32
φ	134	140	331214	152	158	164	6452	176
n	33	34	35	36	37	38	39	40
φ	182	188	4235186	200	206	212	17353500	224
n	41	42	43	44	45	46	47	48
φ	230	236	134844006	248	254	260	762406088	272

Table 4: Semi-cycle length for small n , standard initial configuration with fleas positioned adjacently

We can immediately see that this is not linear like the 1 flea case. Running code to compute the semi-cycle length for the n up to 30 we find

It is apparent that certain n -gons have much longer semi-cycle than others. Particularly for $n \equiv 3 \pmod{4}$, the length of a semi-cycle is far longer than for any other n . We also note that even among $n \equiv 3 \pmod{4}$, the sequence of φ 's

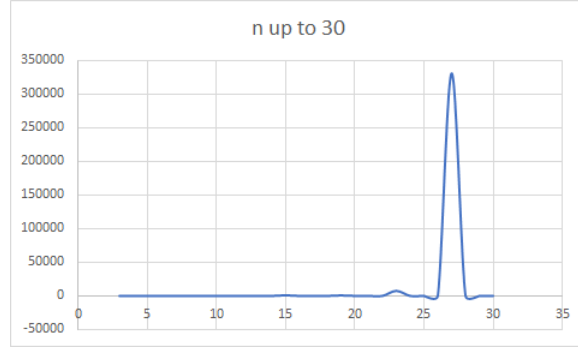


Figure 1: Subcycle length of small n -gons. $3 < n \leq 30$

is not strictly increasing. For $n = 27$, $\varphi = 331214$, while for $n = 31$, $\varphi = 6453$, whereas $n = 28, 29, 30$ have $\varphi = 152, 158, 164$ respectively.

Interestingly, if we only look at polygons with a number of sides not congruent to 3 mod 4, the story changes completely. Then the data suggests that

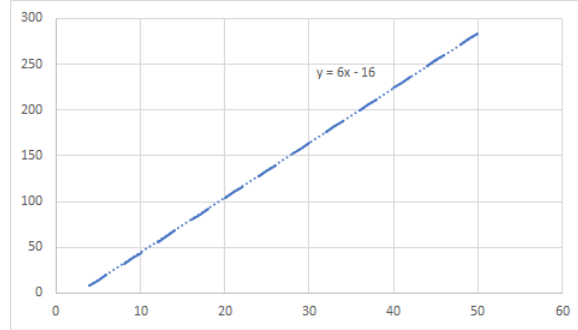


Figure 2: $n \not\equiv 3 \pmod{4}$

$\varphi = 6n - 16$, which is a linear relation much like the 1 flea setting.

Conjecture: For $n \equiv 0, 1, 2 \pmod{4}$, φ is linear, specifically $\varphi = 6n - 16$.

4.2 Reaching a Configuration of all \times 's

In the one-flea setup explored in Section 3, the number of moves it took to reach an n -gon where every node occupies the state \times starting from an initial configuration of all \checkmark 's was n . It seems natural to extend this question to the two-flea problem, where we provide the following observations:

Lemma 4. *The number of \times 's after a move on a two-flea n -gon always changes by an even number.*

Corrolarry to Lemma 4: For odd n , an n -gon with initial configuration of all \checkmark with two fleas will never reach a configuration where every vertex has the state \times .

Proof. We consider the possible positions of the fleas at a given time.

1. Both flea a and flea b occupy the same vertex (v_i, \checkmark) : a leaves the vertex, updating it to (v_i, \times) . b leaves the vertex, updating it to (v_i, \checkmark) . A net increase of $0 \times$'s.
2. Both flea a and flea b occupy the same vertex (v_i, \times) : a leaves the vertex, updating it to (v_i, \checkmark) . b leaves the vertex, updating it to (v_i, \times) . A net increase of $0 \times$'s.
3. Flea a occupies (v_i, \times) , flea b occupies (v_j, \times) , $i \neq j$: a moves, sending $(v_i, \times) \rightarrow (v_i, \checkmark)$, then b moves, sending $(v_j, \times) \rightarrow (v_j, \checkmark)$. A net decrease of $2 \times$'s.
4. Flea a occupies (v_i, \checkmark) , flea b occupies (v_j, \checkmark) , $i \neq j$: a moves, sending $(v_i, \checkmark) \rightarrow (v_i, \times)$, then b moves, sending $(v_j, \checkmark) \rightarrow (v_j, \times)$. A net increase of $2 \times$'s.
5. Flea a occupies (v_i, \checkmark) , flea b occupies (v_j, \times) , $i \neq j$: a moves, sending $(v_i, \checkmark) \rightarrow (v_i, \times)$, then b moves, sending $(v_j, \times) \rightarrow (v_j, \checkmark)$. A net increase of $0 \times$'s.
6. Flea a occupies (v_i, \times) , flea b occupies (v_j, \checkmark) , $i \neq j$: a moves, sending $(v_i, \times) \rightarrow (v_i, \checkmark)$, then b moves, sending $(v_j, \checkmark) \rightarrow (v_j, \times)$. A net increase of $0 \times$'s.

□

5 The Cop and Robber Fleas

Consider again the case of two fleas on a polygon, but label one a “cop” and the other a “robber”. If the cop lands on a vertex in the \checkmark state, it moves towards the robber; and if the robber lands on a vertex in the \checkmark it moves away from the cop. Of course, if either land on a vertex in the \times state, they must move in the direction opposite that they would have on a \checkmark .

Some natural questions are: will the cop flea always catch the robber flea? Where is the best location for the robber flea to commit his “crime” so he can avoid the cop for the longest? Can we determine how many steps it will take to catch the robber flea given initial conditions?

Remark. “Move towards” and “move away”, meaning minimizing and maximizing the shortest path between the cop and robber, can be ill-defined on a general graph. Thus, we will only consider the case of an odd number of sides polygon.

Remark. Notice that on a complete graph the cop flea is guarenteed to win as soon as he reaches a vertex in the \checkmark state.

5.1 Computational Evidence

Using an expanded version of our previous computer programs, we find that the cop is not guaranteed to catch the robber on even a polygon of size seven! There is not a clear general pattern as to when a cop will catch the robber, and how long it will take. However, we were able to find the following smaller patterns of cases where the robber evades the cop indefinitely:

n	d	φ
7	2	11
9	3	13
11	4	15
13	5	17
15	6	19
\vdots	\vdots	\vdots

Table 5: Given an n sided polygon, and an initial distance of d between the cop and robber, the two will enter a semi-cycle of length φ which repeats forever. Note that φ counts only the cops' steps.

There is some structure here. On an n -gon where $n = 2m + 1$, if we choose a distance $d = m - 1$ the cop and robber fleas will never cross paths, and will go through cycles of length $\varphi = 2(m + 2) + 1$. However, if you consider the larger set of data for which the cop does not catch the robber in the Appendix, it is still an open question whether a similar general rule exists.

Furthermore, the length of time it takes the cop to catch a robber varies in strange ways and it is unclear what processes are taking place.

6 Appendix

6.1 Cops and Robbers Flea Data

The Robber Gets Away							
d	2	3	4	5	6	7	8
n	7	9	11	13	15	17	19
	20	25	15	17	17	23	27
	23	41	31	19	23	49	31
	31	81	39	29	33	53	63
	39	93	71	43	43	77	71
	47		95	59	51		97
	49			69	55		99
	67			73	95		
	71			81			
	79						

Table 6: Given a distance d , this table lists the n -sided polygons on which the robber is not caught for $2 \leq n \leq 100$.

6.2 Pseudo-Code

We made liberal use of the `NetworkX` python package, which allows a user to generate and manipulate graphs. A pseudo-code outline of our program is below:

```

function BUILDGRAPH( $n$ )
    Make polygon graph of size  $n$ 
    Initialize vertices to chosen initial conditions
    return Graph
end function
function SEMICYCLECHECK(Graph, Initial Conditions)
    if Current Graph State does not match initial conditions then
        return False
    else
        return True
    end if
end function
function TRAVERSEGRAPH(Graph)
    Initialize flea with a position, direction and step counter
    Take first step: ▷ This is so the While loop executes
        change state of previous vertex
        update position, direction, and step counter
    while SEMICYCLECHECK(Graph) returns True do
        Take a step:

```

```

        Change state of previous vertex
        update position, direction, and step counter
    end while
    return position, and step counter
end function
function MAIN
    Graph ← BUILDGRAPH(n)
     $x, \varphi \leftarrow$  TRAVERSEGRAPH(Graph)
end function

```