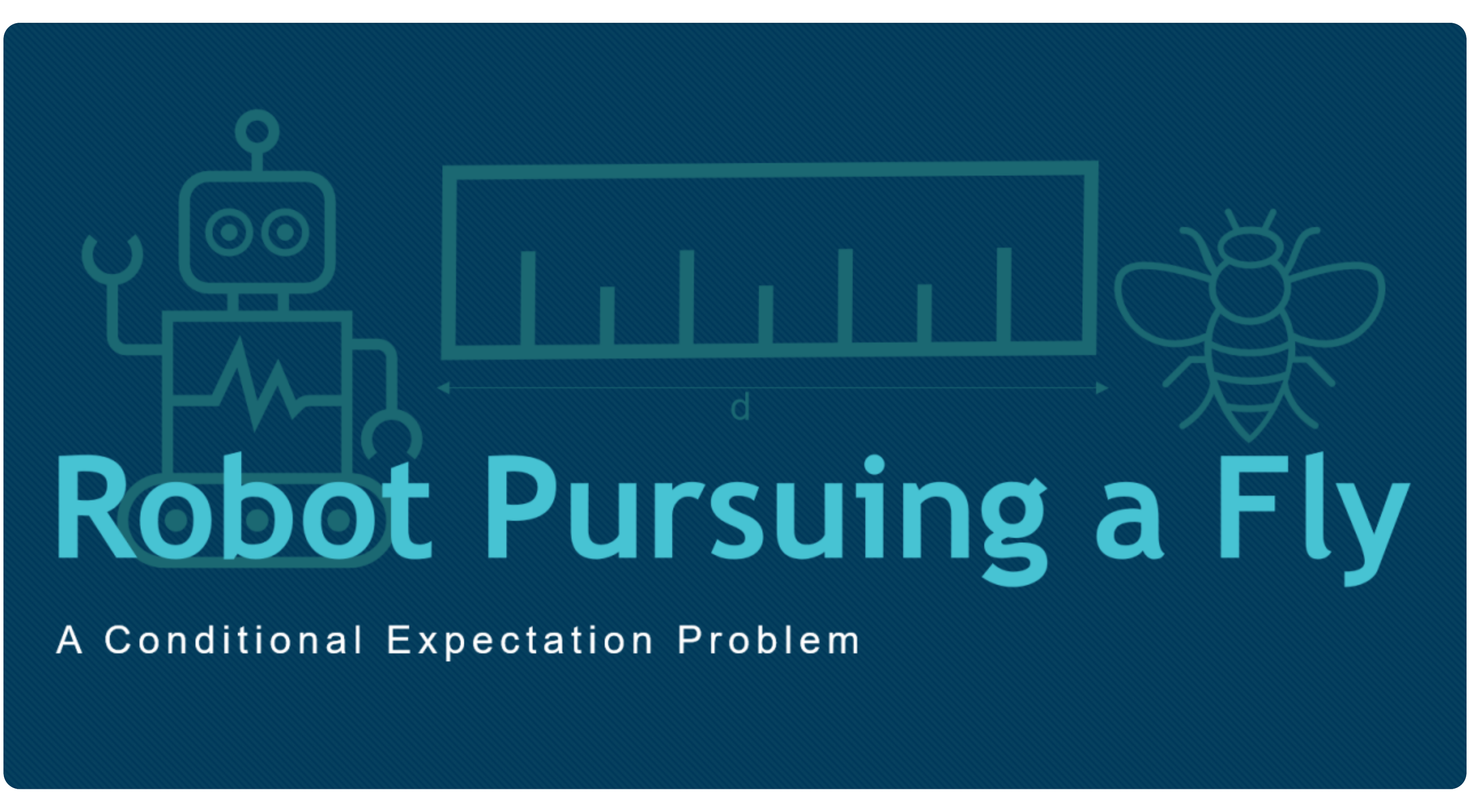
Pursuit of Expected Value

Published on November 17, 2021 Edit article | View stats




Nabeel Khan
MS in Data Science at SUNY Buffalo | IIT Roorkee | Machine Learning Engineer with 3+ years of experience & seeking internship for Summer'22
1 article

Problem Statement

- A Robot and a fly move along a straight line
- At each second, the fly moves a unit step to the right or to the left with equal probability ***p***, and stays where it is with probability ***1-2p***, and the robot moves a unit towards the fly with the probability of ***q***, or stays still (with the probability of ***1-q***) regardless of the fly movement.
- The Robot and the fly start ***d*** units apart, where ***d*** is a random variable taking integer values from 10 to 19 (with equal probabilities).
- Experiment ends when the robot and the fly land on the same spot, or when they cross each other**.**
- Find the **Expected Value** of Time (Et) for a robot to catch fly, for ***p*** = [.1, .2, .3, .4, .5] & ***q*** = 1.5***p***

Pure Analytical Approach:

For initial distance ***d*** there are 6 cases possible after 1 unit of time.

Case	Robot movement	Fly movement	Distance	Probability
1	Still	Left	d-1	(1-q)p
2	Still	Right	d+1	(1-q)p
3	Still	Still	d	(1-q)(1-2p)
4	Right	Left	d-2	qp
5	Right	Right	d	qp
6	Right	Still	d-1	q(1-2p)

Using Conditional Expectation we can break the expected value with the above cases

$$E[T|A_d] = P(Robot\ still, Fly\ moves\ Left)E[T|A_d \cap B_{d-1}] + P(Robot\ still, Fly\ moves\ Right)E[T|A_d \cap B_{d+1}] + P(Robot\ still, Fly\ Still)E[T|A_d \cap B_d] + P(Robot\ moves\ right, Fly\ moves\ Left)E[T|A_d \cap B_{d-2}] + P(Robot\ moves\ right, Fly\ moves\ Right)E[T|A_d \cap B_d] + P(Robot\ moves\ right, Fly\ Still)E[T|A_d \cap B_{d-1}]$$

where,

A_d: *The event that* ***initially*** *the robot and the object are* **d** *units apart*
B_d: *The event that* ***after one second*** *the robot and object are* **d** *units apart*

The above equation can simplified and rephrased as the equation below (with E[T|Ad] = Ed, and q = 1.5p)

$$E_{d+1} = \frac{1 + (2.5p - 4.5p^2)E_{d-1} + (4.5p^2 - 3.5p)E_d + 1.5p^2 E_{d-2}}{p(1.5p - 1)}$$

This seems great, as the Expected value for d+1 is dependent on expected values for d, d-1, and d-2, and we should be able to use this to everything starting with the base case E[1].

But this equation is only valid for d ≥ 1, as E[0] (bot and fly at same spot) and E[-1] (bot and fly crossing each other) are the minimum possible values. Hence we can't use this equation to find E[1] (and other expected values). **This is a big setback for the Pure analytical approach.**

Pure Simulation Approach

Using below python script, it is possible to achieve satisfactory results.

```
import numpy as np

# Final Expected Values for p = 0.1 to .5
final_exp_vals = []

for p in [.1, .2, .3, .4, .5]:
    bot_mvmts = np.array([0,1])
    fly_mvmts = np.array([-1,0,1])
    fly_mvmt_probs = np.array([p, 1-2*p, p])
    bot_mvmt_probs = np.array([1-1.5*p, 1.5*p])
    time_taken = []

    # Running Experiment for 10000 times, and averaging for expected values
    for sim_count in range(10000):
        time_count = 0
        bot_position = 0

        # Randomly placing fly for d = 10 to d = 19
        fly_position = np.random.choice(range(10,20))
        while bot_position < fly_position:
            bot_position += np.random.choice(bot_mvmts, 1,p=bot_mvmt_probs)[0]
            fly_position +=  np.random.choice(fly_mvmts, 1,p=[p, 1-2*p, p])[0]
            time_count += 1
        time_taken.append(time_count)
    final_exp_vals.append(sum(time_taken)/len(time_taken))
```

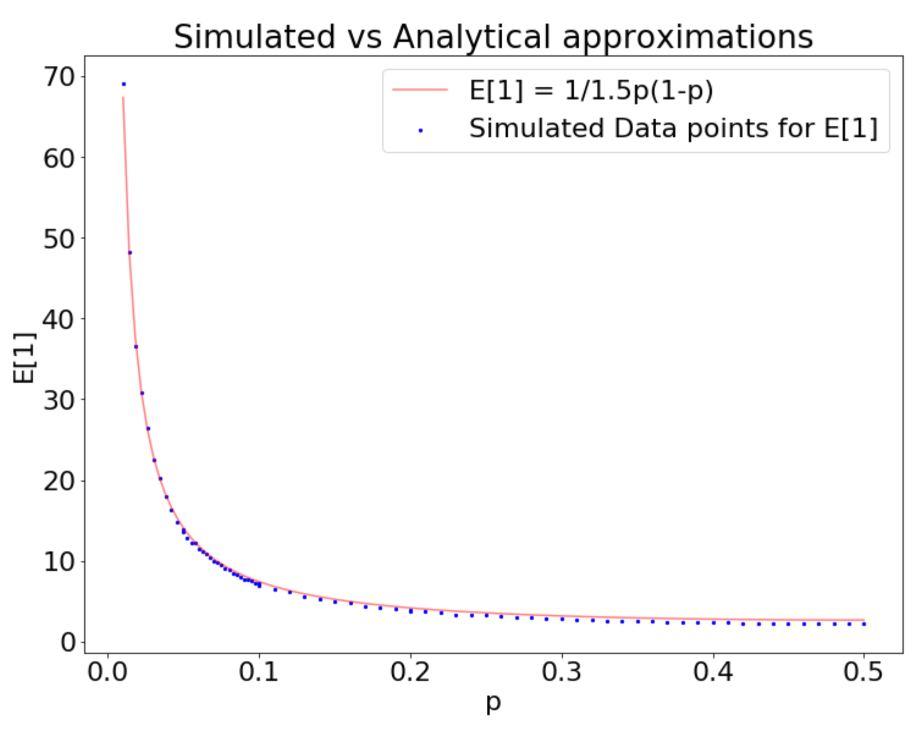
p	Simulated Expected Value
0.10	96.7419
0.20	48.9432
0.30	32.7241
0.40	24.5996
0.50	19.8819

Simulation is not a scalable approach as it is compute intensive, and its not extremely precise.

Mixed Approach

Since we have approximate results from simulation we can try different approaches and know their effectiveness.

By calculating E[1] for extreme cases i.e., ***p*** = 0 and ***q*** = 0, we can guess approximate value of E[1] to be 1/(q(1-p)) (or 1/(1.5p(1-p)))



With this plot we can confirm that 1/(q(1-p)) is very close to E[1]. We should be able to use this E[1] in the analytical equation to get the required expected values.

The analytical equation calculated previously

$$E_{d+1} = \frac{1 + (2.5p - 4.5p^2)E_{d-1} + (4.5p^2 - 3.5p)E_d + 1.5p^2 E_{d-2}}{p(1.5p - 1)}$$

However when we use E[1] from 1/(1.5p(1-p)), we do not get same expected values as the simulated values for larger distances. For larger distances expected values essentially explode and are much larger than simulated values.

This tells us that: E[1] = 1/(1.5p(1-p)) - error term.

The Expected values for larger distances were very sensitive to E[1] towards both +∞ and -∞**.** We can exploit this behavior to calculate the **error term with very high precision (15-20 decimal places)** without any simulation.

```
p_error_terms = []
for p_om in np.linspace(.1,.5, 5):
    q_m = 1.5*p_om
    error_term = 0
    for nth_place in range(1,20):
        next_it = True
        for var_term in range(9,-1,-1):
            if next_it:
                e1 = 1/(q_m*(1-p_om)) - (error_term + var_term/(10**(nth_place)))
                e2 = (1 +(3*p_om*q_m - 2*p_om -q_m)*e1)/(p_om*(q_m-1))
                e3 = (1 +(3*p_om*q_m - 2*p_om -q_m)*e2 + \
                     (p_om + q_m -3*p_om*q_m)*e1)/(p_om*(q_m-1))
                exp_val_2 = [e1,e2,e3]
                for n in range(4,50):
                    e_t = (1 +(3*p_om*q_m - 2*p_om - q_m)*exp_val_2[-1] \
                           + (p_om + q_m -3*p_om*q_m)*exp_val_2[-2] \
                           + (p_om*q_m)*exp_val_2[-3])/(p_om*(q_m-1))
                    exp_val_2.append(e_t)
                if exp_val_2[-1] > 0:
                    error_term += var_term/(10**(nth_place))
                    next_it = False
    p_error_terms.append(error_term)
```

Using the above script we can estimate E[1] with high precision and then use the analytical approach to get final expected values.

Doing so we get:

p	Analytical Expected Value	Simulated Expected Value
0.10	97.082	96.7419
0.20	48.767	48.9432
0.30	32.678	32.7241
0.40	24.651	24.5996
0.50	19.856	19.8819

These should be the actual Expected Value of Time (Et)

To get more precision we can find the equation for **the error term** by using the below p vs error term plot.