ASSIGNMENT 0

Sachin Kumar(MC22BT019), Amogh R (CS22BT005)

1 Introduction

In the described scenario, a defending country (DC) aims to secure its border against an attacking country (AC) by deploying a wireless sensor network. The wireless sensors, strategically placed along the border, detect infiltration attempts, enabling the defending country to respond effectively. Key components include the deployment of sensors, real-time communication between them, sophisticated detection algorithms, and a rapid response mechanism involving the deployment of troops. Security measures are essential to prevent tampering, and continuous monitoring and improvement of the sensor network are crucial for its effectiveness. Ethical considerations and adherence to international laws are emphasized in the implementation of defense measures.

2 Elements of the Scenario

2.1 Border

- **Border Layout:** The border is represented as a long rectangular strip of land, with the length extending to infinity and discretized into a grid.
- **Grid Structure:** The rectangular strip is divided into a grid, with the width consisting of `W` cells. Each cell in the grid represents a discrete unit of the border.
- **Infinite Length:** While the length of the rectangle is considered infinite, the focus is on the discretized grid structure, which allows for a systematic representation of the border.
- **Geometric Configuration:** The discretization into a grid simplifies the analysis and deployment of a wireless sensor network for border defense, with each cell potentially hosting a sensor for detecting infiltration attempts.

2.2 Sensors

- **Single Sensor per Cell:** Each cell in the border grid is equipped with a motion sensor, capable of detecting moving infiltrators within its own cell.
- **Limited Sensing Range:** Sensor detection is confined to its assigned cell; it triggers only if an infiltrator is in motion within that specific cell.

- **Energy Conservation:** To extend battery life, sensors employ duty cycling, randomly deciding to be ON or OFF every 10 seconds. This decision is independent and made without communication.
- **Random Decision Making:** Sensors make decisions based on a coin flip with a probability p of heads. If heads, the sensor remains ON for the next 10 seconds; otherwise, it turns OFF.
- **Independent Operation:** Each sensor operates autonomously, with no communication with other sensors. The initial decision is made at time '0', and subsequent decisions are made independently.

2.3 Infiltrator:

- **Infiltrator's Movement:** The infiltrator moves in steps, with the ability to choose any of the 8 neighboring cells. Each movement takes 9 seconds.
- **Decision and Movement Cycle:** Every 10 seconds, the infiltrator spends 1 second studying neighboring cells and the next 9 seconds moving (if he decides to move).
- **Sensor Model Reminder:** The motion sensor model dictates that if the infiltrator decides to move from cell A to cell B, both cells' sensors must be OFF; otherwise, the infiltrator is caught.
- **Sensor Status during Movement:** For successful movement, both the departure (cell A) and destination (cell B) sensors must be OFF during the infiltrator's 9-second movement interval.
- **Risk of Detection:** If even one of the sensors in cells A or B is ON during the infiltrator's movement, it triggers the motion sensor and leads to the infiltrator being caught.

3 Objective

The main objective of this problem is to determine the time taken by the infiltrator to cross the border while varying the width of the border and the probability of the sensor being on.

4 Solution

- We define a `Clock` class where methods related to time, such as `getTime()` and `incrementClockTime()`, are defined to start the clock.
- Then, we define a `Sensor` class where a method `flipACoin()` is defined to handle the sensor's behavior. It mimics tossing a biased coin (with some probability), and turning the sensor on or off based on the bias of the coin.
- Next, we define a `Border` class that includes the `Sensor` within a 2D array.

- Following that, we define an `Infiltrator` class where numerous methods are defined. The infiltrator always starts from the point (`mid`,0), where `mid` is the X-value of the center of the finite border in the simulation. Ideally, this tries to mimic an infinitely long border.
  - `getMotionStatus()`
  - `getInfiltrationStatus()`
  - `changeMotionStatus(boolean status)`
  - `infiltrationSuccess(boolean hasCrossed)`
  - The method `studySurrounding(Border border)` allows the infiltrator to gather information about the surrounding sensors.
  - The method `moveInfiltrator(int pos)` enables the infiltrator to move safely without being caught.

5 Plots

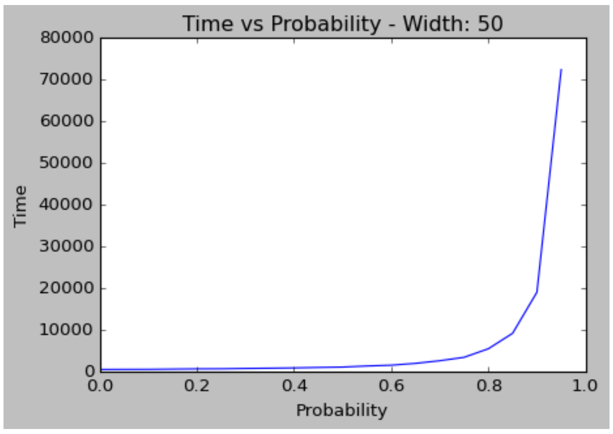

Figure 1: Average Time taken to cross border vs Probability of sensor turning on

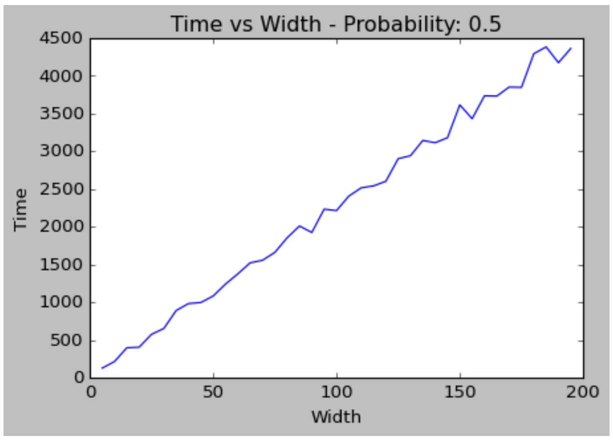

Figure 2: Average Time taken to cross border vs Width of border

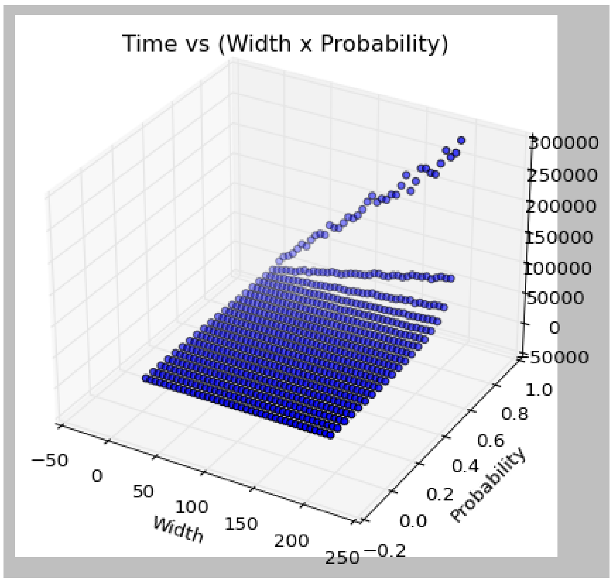

Figure 3: A 3D Plot of Time, Width and Probability

6 Observation

The following observations can be made from the data collected:

- The time taken to cross the border increases drastically as the probability of the sensor turning on reaches the 80% threshold.
- The time taken to cross the border increases linearly with the border width for a fixed probability of the sensor turning on. This is as expected because the average time taken for the infiltrator to make one move does not depend on his position, and rather depends on the probability of the sensors around him turning on/off (kept constant here).