CS311 : Computer Architecture Lab
Assignment 0 : Familiarizing with Java

Rushikesh Dixit 180030037
Shreyas Sathe 180010033
Ritwik Singh 180010028

September 4, 2020

Abstract

In this experiment, we studied that how the values of Sensor Duty Cycle Probability and Border Width can affect the time for Infiltrator to cross the Border.

1 Problem Statement

Original Statement : Consider the scenario where one country, called the defending country (DC), wishes to defend its border against another country, called the attacking country (AC), whose aim is to send an infiltrator to cross the border and enter DC's land. DC decides to deploy a wireless sensor network along the border. If a sensor detects an infiltration attempt, DC can then send its troops to counter the infiltration with "fire and fury" (Trump style!). Quite obviously, the infiltrator would like to enter DC's land without triggering any sensors.

In brief : To find the time taken by the infiltrator to cross the border for different combinations of width and probability of each sensor to be ON.

2 Simulation

Java was used to simulate the scenario since it is a object-oriented programming language. In this scenario, separate classes for Border, Sensor, Infiltrator and Clock to modularise the solution. An Experiment Class was also created to store the value of average time taken for a finite number of iterations for specific values of Sensor Duty Cycling Probability and Border Width. A detailed description of all the Classes involved are given below.

2.1 Border

Border Class stores the physical shape of the Border. In current declaration, it stores the Border Width value for further use. Within the class, there are two methods. First is changeWidth method which takes an integer newWidth as an input and changes the current value of width to newWidth. Other is getWidth method which returns the current value of the width in use.

2.2 Clock

Clock Class stores the current value of time passed when the Infiltrator has started moving. It stores the current time value for further use. Within the class, there are 3 methods.First is resetTime method which resets time value to zero. Third method incrementTime10 increments the current value of time by 10 units.

2.3 Experiment

Experiment Class stores the dynamic state of 4 cells of Sensor grid that the Infiltrator needs to care about viz. CurrentCell, FrontCenterCell, FrontLeftCell and FrontRightCell. It also stores the current state of the Infiltrator. This class has a method doExperiment which takes 3 inputs viz. number of iterations, experimental Sensor P-value and experimental width and returns the average time taken to cross border successfully.

2.4 Infiltrator

Infiltrator Class stores the current state of infiltrator whether standing (0) or moving (1). It has two methods viz. getInfiltratorState method which returns current state of Infiltrator and the other one is changeInfiltratorState which changes the current state of Infiltrator to other state.

2.5 Sensor

Sensor Class stores the current state of sensor and initialise the state randomly based on duty cycling probability provided. It has methods viz. intialiseState which updates the state of sensor, getSensorState returns current sensor state, getPvalue returns currently used P-value and changePvalue which helps change Sensor P-value.

2.6 Main

Main Class contains driver code to do experiments.

3 Methodology

Initially, the Infiltrator is presumed to be standing behind the First Row of the Border i.e at Attacking Country (AC) Area.

As soon as the Infiltrator tries entering the Border, the Clock starts from Zero. For entering the First Row of the Border, the Infiltrator has to take care only about 3 Sensor Cells : Front Center Cell, Front Left Cell and Front Right Cell. As soon as any of the three sensors change to OFF state, Infiltrator is able to move forward one step to reach the First Row.

Now, since the Infiltrator is in First Row, he has to consider the states of 4 Sensor Cells : Current Cell on which he is standing and Front Center Cell, Front Left Cell and Front Right Cell. As soon as the a situation arrives, where Current Cell state is OFF and any of the 3 Front Cells is in OFF state, only then the Infiltrator can move forward. This method keeps working till the Infiltrator reaches the Last Row of the Border.

Now, since the Infiltrator is standing in the Last Row, he only has to move 1 step forward in order to enter Defending Country (DC) area. For doing so, he needs to keep track only on his Current Cell to be in OFF state, as soon as Current Cell changes to OFF state, Infiltrator moves one step forward and reaches DC. Here, the process ends. And the output comes out as the Time Taken to cross Border.

Similarly, we gather Average Time Taken values for 1000 iterations for specific pair of Sensor P-value and Border Width.

4 Results and Graphical Aspects

Shown below are the results of the experiment involving 1000 iterations for each specific pair of p-value and border width.

5 Conclusion

Key conclusions include :-
1. Average Time Taken increases linearly with Border Width value.
2. Till P-value of about 0.7, average time taken is almost constant but after 0.7, the average time taken value starts rising exponentially with increase in Sensor P-value.

6 Steps to reproduce Whole Experiment

1. Extract the ZIP package.
2. cd to 'ca_assignment_0' folder.
3. Run command **$ javac *.java**
4. To run series of experiments with P-value from 0.00 to 0.95 and Width value from 5 to 100, run command **$ java -cp . Main**
5. To run an experiment with command-line input as number of iterations, P-value, Width, run command **$ java -cp . Experiment numExp pvalue width**
e.g. **$ java -cp . Experiment 1000 57.3 32**

7 Attachments

GitHub Repository for this assignment.

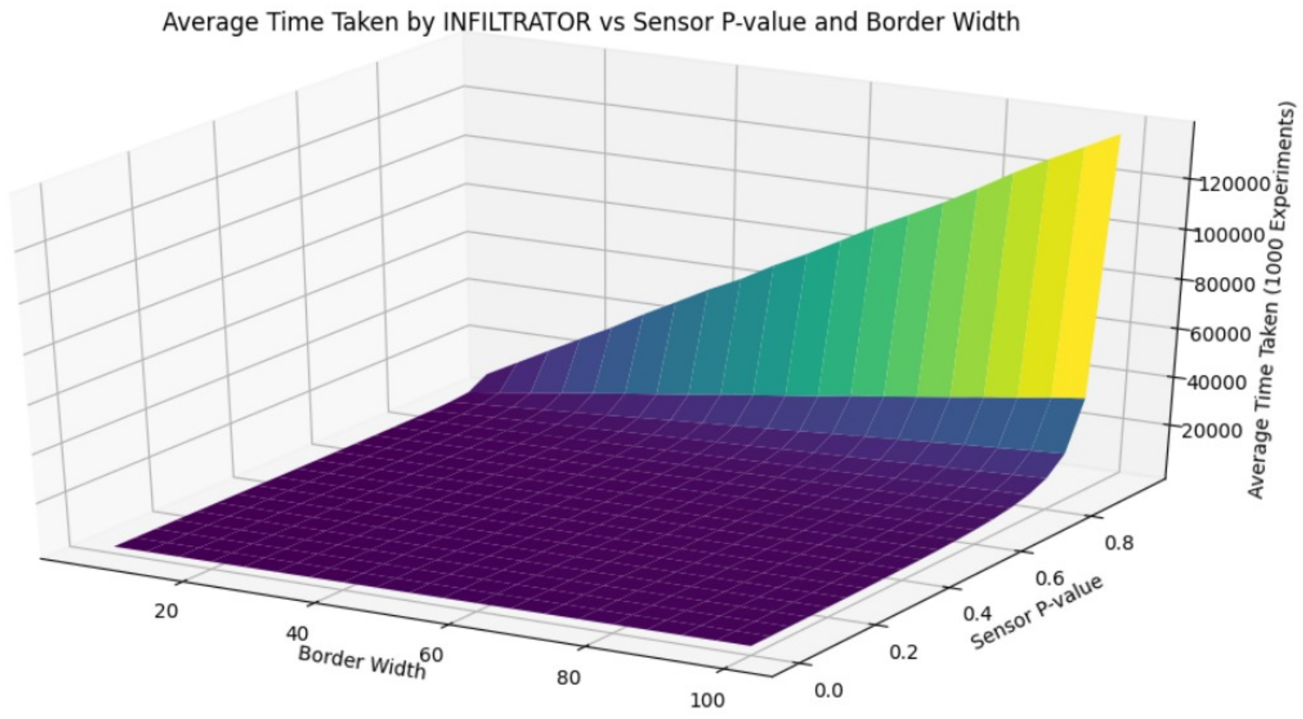

Figure 1: Figure shows changes in average time taken (1000 iterations) for specific values of Sensor P-value and Border Width. Varied the value of probability from 0 to 0.95 in steps of 0.05 and the value of width from 5 to 100 in steps of 5.