Computer Architecture Assignment 0

Om Patil (200010036)
Hrishikesh Pable (200010037)

August 2022

1 Problem 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. Quite obviously, the infiltrator would like to enter DC's land without triggering any sensors.

2 Structure of Program

The program is divided into the following classes:

2.1 Class Clock

The clock class keeps track of the discrete time in the simulation. It has the class variable time, which denotes the world time, and is first set to zero in the constructor.

2.1.1 incrimentTime ()

This function increments time by 'increment' number of units when called.

2.2 Class Sensor

This class emulates the sensors on the border, and keeps track of the probability of the sensor being on (int p), whether the sensor exists on the grid or not (boolean exist), and whether it is On or Off (boolean isOn).

The constructor sets the values of the booleans p and exists. It also calls the runDutyCycle function once.

2.2.1 runDutyCycle()

This function samples a random float value between 0 and 1, and checks if the value is lesser than the threshold p, in which case it sets the isOn variable, else it is reset.

2.3 Class Border

This class emulates the border between AC and DC, which has a width = width and is infinitely long. Each cell of the border contains a sensor. It is implemented as a 3*3 grid which keeps on changing as the infiltrator keeps on moving, and simulates the entire border.

The border is characterised by width (of border), the intruder_progress (how far he has reached), booleans for whether intruder is detected or not and whether he crossed the border

2.3.1 runDutyCycle()

This function runs the runDutyCycle function of every sensor on the grid.

2.3.2 moveIntruder()

This function takes in the change in x and y coordinates in the form of x and y parameters, and modifies the 3*3 grid to create the effect of moving the intruder across the border.

2.3.3 updateView()

This function updates the neighbour_status array (which keeps track of sensors which actually exist among the 9) and the On_sensors array which keeps track of the sensors that are turned On currenly.

2.4 Class Infiltrator

This is a simple class that emulates the Infiltrator. It consists only the NextMove() function which is as follows:

2.4.1 NextMove()

This function is the logic which infiltrator uses to move on the grid. It checks the status of the 9 sensors around it, and accordingly moves the infiltrator

2.5 Class Simulation

This class creates objects of the other classes and handles the main loop of the program, using the run function

2.5.1 run()

This function runs the actual simulation by using the other class objects and following the problem structure.

3 Observations

The Time taken to get across the border for the infiltrator is linear in width and exponential in the duty cycle p.

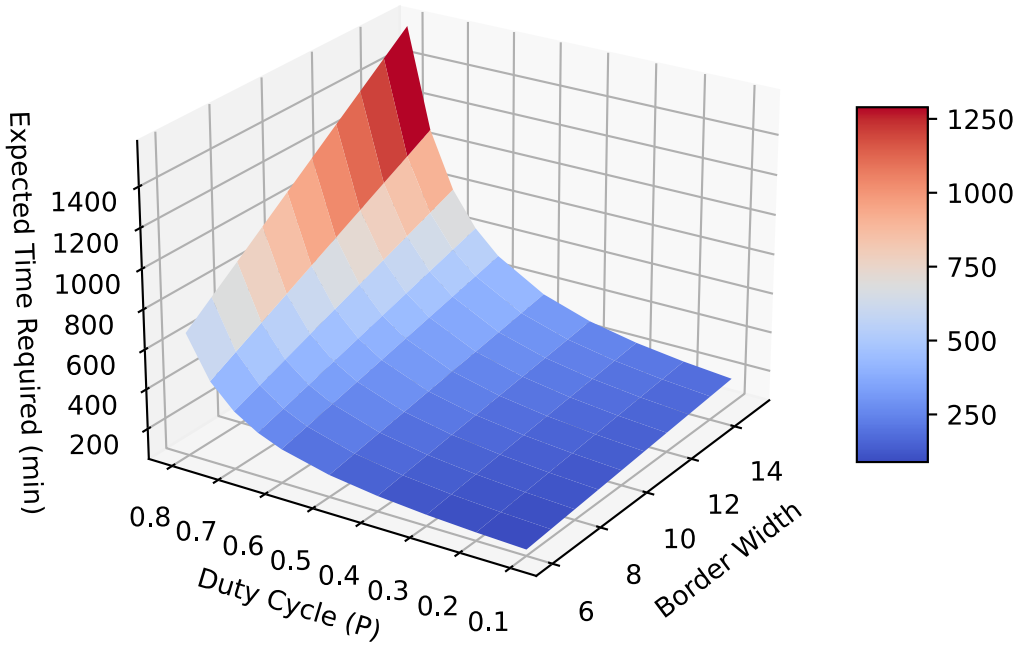

Figure 1: plot of Time taken vs p vs width