Shreyyas Vanarase

Christian Tuchez

CX 4230 Computer Simulation

Prof. Vuduc

Mini Project 3: GT Evacuation Simulation

Part A: Conceptual Model

Introduction

Agent based models and queuing models can be used to simulate interesting scenarios. The purpose of this project is to develop and test an evacuation plan for the individuals at Georgia Tech. To do this, we must develop and use a discrete event simulator based on a queuing network conceptual model. As one can imagine, in an emergency situation, unless controlled, individuals will try to leave all at once resulting in severe traffic along the roads to the highway. We assume that individuals leave from Georgia Tech's West campus and exit the campus by entering the highway through three main routes: North Avenue, Fifth Street, and Tenth Street. The simulation is stochastic, meaning that randomness will be utilized to simulate unpredictable elements of the system such as the number of vehicles required to evacuate. In addition, select intersections within the Georgia Tech road system have a police officer present who will direct traffic according to a set of predefined rules. In the following sections, we present our conceptual model for this simulation and identify the high level software architecture of the application. We will also provide highlights of our software implementation, analyze the biasness our random number generator, and provide a bound for the amount of time required to evacuate campus.

Conceptual Model

This project runs a full agent body simulation of vehicles evacuating the GaTech campus during an emergency. The conceptual model of this simulation combines agent-based models and queuing models. In the context of the agent based model, our agents are called vehicles, the smallest unit of our simulation. Vehicles represent the individuals trying to leave GaTech and are initially located in the parking lots of West campus. In the context of the queuing model, parking lots and intersections are servers and roads are queues. Vehicles run on objects called Roads. Each road knows its direction and the maximum number of vehicles that can be situated on a road at a given time. Roads themselves are connected by objects called intersections. Intersections handle how vehicles move from one road to the next.

When the simulation begins, vehicles will flow out of the parking lots and enter the queue. Each vehicle will move at the same speed and will only move if there is no vehicle stopped in front of it. Upon reaching an intersection without a police officer, a vehicle will move in FCFS (first Come First Served) fashion. That is, if there is no police officer present, the vehicle that arrived at the intersection the earliest will be the first to move forward. If there is a police officer present then the vehicle will obey the rules imposed by the officer. The police officer checks if there are three empty spots on a road and if there are three cars at the front of another road in an intersection attempting to leave campus. If these conditions are met then the officer moves three cars at a time from one road to another. If there are less than three cars or spots on either the

source road or destination road, respectively, then the usual FCFS behavior takes place. Once the vehicles have crossed the interstate they exit the simulation.

Stochastics

This project utilized *drnglib* to obtain access to Intel's Digital Random Number Generator (Beccario 2013). Intel's random number generator retrieves cryptographically secure random numbers directly from the CPU using the rdrand instruction. Each value is calculated based on the thermal entropy of the silicon within the internal hardware (Mechalas 2014). This library allowed us to stochastically determine the number of vehicles in each parking lot. For each parking lot, upper and lower bounds limit the maximum and minimum number of cars generated, respectively.

Map

Figure 1 shows the Georgia Tech road system we have used in our simulation. This map has been coarsened by removing the extraneous geographical obstacles to allow us to better run experiments and perform analyses. To provide some intuition for the functionality of the map, we discuss its key features below.

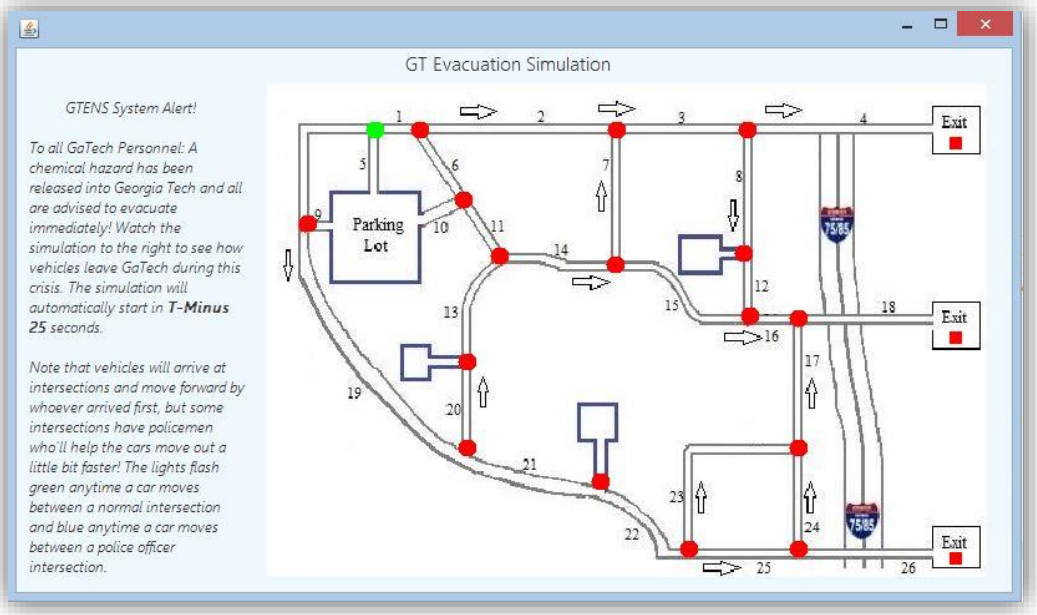

Figure 1: Simplified Georgia Tech Road System

Initially, vehicles are created by the car maker and are stored within queues inside of the parking lots. The four parking lots in our simulation are represented by the four blue squares on the map. Upon leaving the parking lot, vehicles will move toward one of the three exits along one-way roads. Each road has a maximum capacity proportional to its length on the map. If a street connecting a parking lot to the road system is fully occupied then the next vehicle in queue to leave the parking lot will wait until an empty spot has appeared on the road and then will move onto the road. Cars move along the roads until they reach intersections, designated by the red circles on the map. The intersection will turn green to notify that a vehicle has crossed from one road to the next. If a police is present at an intersection, the intersection will light up blue to notify a police has directed vehicles from one road to another. When vehicles reach the exit, the red squares underneath the exit signs will light up green to notify that a vehicle has exited. These functionalities allow a user to visualize vehicle movement. Note that on this map, intersections are checked one by one to see if a vehicle has passed through it or not. As a result, multiple intersections will not light up at the same time.

Part B: Software Architecture and Implementation Highlights

Software Architecture

The simulation has been developed in an object oriented framework through the Java programming language. Before delving into the class structure, **Figure 2** below presents a high level software architecture. Our internal software determines vehicle functionality and road/intersection functionality. Cameron Beccario's DRNG library complements our software by providing access to Intel's DRNG (Becarrio 2013). By using this library, we can simulate the behavior of a random number of cars that evacuate the campus. We integrated all these features into our Swing-based GUI as shown in **Figure 3** to the right.

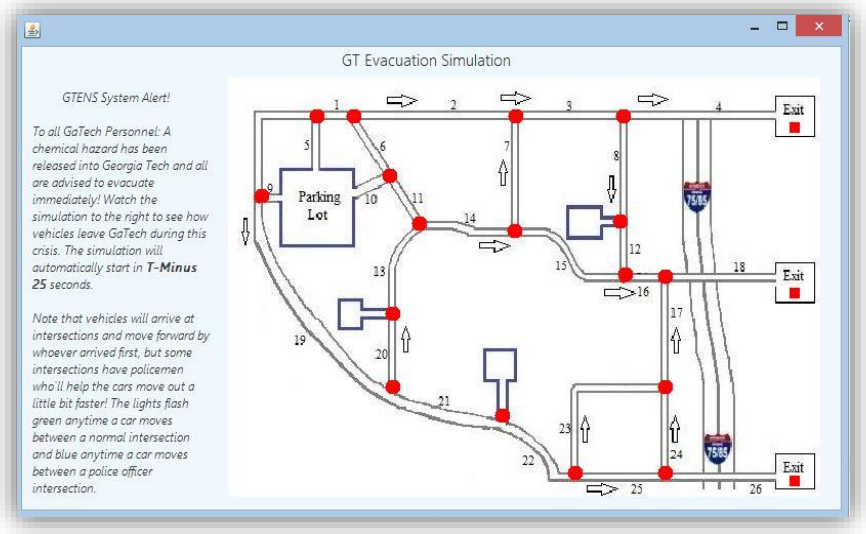

Figure 3: GaTech Evacuation Simulation GUI

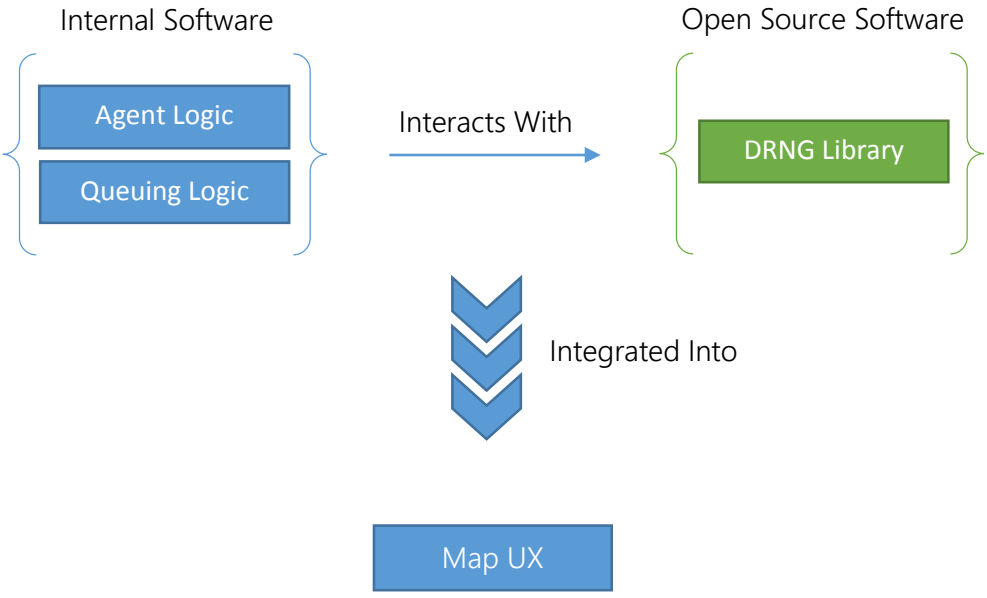

Figure 2: Software Architecture Diagram

Software Implementation Highlights

Figure 3 below depicts the important classes in our code and highlights the relevant attributes and methods that drive the main functionality of the simulation.

Class	Description	Attributes	Methods
Simulation Driver	The main driver of the simulation. It generates the map network (roads, intersections, and exits) and runs the simulation for a set number of iterations.	*NUMBER_OF_ITERATIONS:* The number of iterations to run	*main()*: Runs the simulation
Vehicle	The individual vehicles that travel the roads.	*carID:* Unique identifier for each vehicle. *timeStamp:* Distinguishes arrival into the queues	Getters/Setters
Road	The physical representation of the queue.	*numVehiclesAllowed:* Maximum number of vehicles allowed on the queue. *direction:* Specifies in which direction the road traffic moves. *roadName:* Unique identifier for each road.	*addVehicle():* Adds a vehicle to a road. *moveCarsForward():* Moves the vehicles on the road forward by one location.
Intersection	The server that moves cars between roads.	*hasPoliceOfficer*: States whether or not the intersection has a police officer. *travelable*: An array of roads a vehicle can drive into at an intersection based on direction.	updateQueues(): Contains logic to check if a vehicle should move from one road to another. *moveVehicleBetweenRoads():* Moves a single vehicle from a source road to a destination road. *moveThreeVehiclesBetweenRoads():* Moves three vehicles between a source road and a destination road if a police officer is present.
MegaParkingLot	A large parking lot connected to three roads.	*road5WaitingQueue*: An internal queue of vehicles waiting to exit the parking lot to road 5.	*updateRoadQueue():* Vehicle exits the parking lot onto the road.
FunSizedParkingLot	A small parking lot connected to a single road.	*roadWaitingQueue*: An internal queue of vehicles waiting to exit the parking lot to some road.	*updateRoadQueue():* Vehicle exits the parking lot onto the road.
Exit	The point of exit for vehicles.	*road:* The road the exit is connected to (e.g. North Ave, 10th Street, or 5th Street).	*removeVehicle():* Removes the vehicle from the simulation system when it enters the exit.
Tesla	The official car maker; it generates cars and assigns them to parking lots.	*UPPERBOUNDS:* Largest number of cars to create for a certain parking lot. *LOWERBOUNDS:* Smallest number of cars to create for a certain parking lot.	*makeAndAssignCars():* Makes a number of vehicles that is determined by the random number generator. *assignCars():* Assigns the vehicles created to the different parking lots.
DRNG	The random number generator; generates cryptographically secure random numbers based on thermal noise in the silicon of the CPU.	*digitalRandom*: Creates a digital random number generator	*getRandomInt():* Returns a random integer within an upper and lower bound.
GeorgiaTechMap	The simulation GUI.	*intersectionList:* List of intersections *exitList:* List of exits	*start():* Begins the simulation. *initialize():* Makes the panel that contains the GaTech Map GUI
ImagePanel	The map resides on this panel; handles drawing the map and changing the colors of intersections/exits during simulation state change.	*intersectionList:* List of intersections *exitList:* List of exits	*paintComponent():* Updates the GUI to represent the latest change in simulation state.

Figure 3: Software Implementation Highlights

Part C: Analysis and Conclusions

Chi Squared Data Analysis

As mentioned before, the purpose of this simulation is to model the behavior of individuals evacuating Georgia Tech. For this simulation since we have stochastically determined the number of vehicles that leave campus during the evacuation, we should first analyze our random number generator actually provides *random* values. By varying the number of cars created in our simulation, evacuation prevention analysts will be better knowledgeable about the *time* required to exit campus during an emergency in general cases. This leads us into our second point of analysis: to provide an estimate for the amount of time required to evacuate campus.

We begin by analyzing the Intel Digital Random Number Generator (DRNG) used to generate our random numbers. This generator uses the entropy inherent to the thermodynamic properties of a processor to generate random numbers. Intel claims this generator creates highly unpredictable, statistically independent sequences of uniformly distributed integers (Mechalas 2014). We tested the DRNG using the Chi Square test using the formula (1.1) below. We generated 100 integers with upper bounds of 10, 30, and 50 for a total of 300 integers.

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{1.1}$$

The results obtained are displayed in **Figure 4** and specified in **Figure 5** below. As we can see the values we observed for chi squared are quite close to those of the theoretical chi square. The theoretical chi squared values are based on the degrees of freedom and the probability there is a larger value is 50%. Since each of the observed values are close to this expected value at 50% with degrees of freedom = |V| -1, the generator can be described as *unbiased* for each of these ceiling values. Since the DRNG is unbiased, it directly implies that there is no bias towards repeating sequences or integers and therefore, it's a good random number generator. Note that the ceiling values are simply the degrees of freedom.

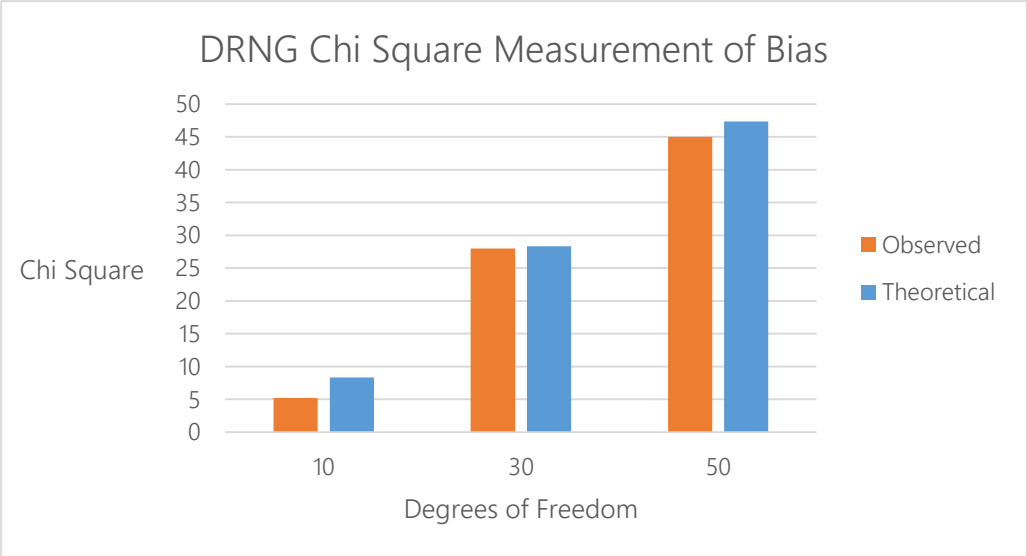

Figure 4: DRNG Chi Square Measurement of Bias

Degrees of Freedom, V	*Observed χ^2*	*Theoretical χ^2*
10	*5.2*	*8.34283*
30	*27.99*	*28.3361*
50	*45*	*47.3350*

Figure 5: Analysis of χ^2

Confidence Intervals Analysis

Having analyzed that our random number generator is quite unbiased, let's consider how long it will take for the campus to evacuate through our simulation. We gathered data for 4 trials:

1) Number of Vehicles ~ 150, with Police Officer Guidance
2) Number of Vehicles ~ 150, without Police Officer Guidance
3) Number of Vehicles ~ 30, with Police Officer Guidance
4) Number of Vehicles ~ 30, without Police Officer Guidance

Figure 6 below shows us the results for these 4 cases. In the first case, we set our upper and lower bounds for each of the parking lots so that we generated about 30 vehicles per simulation. Over 30 trials, we saw that the simulation generated approximately 159 vehicles on average, each iteration about 172 seconds to complete, and our standard deviation was 20.6 seconds. Thus, for about 150 vehicles it takes approximately 3 minutes to evacuate the campus. With some further arithmetic, we can even bound the amount of time required to evacuate campus. Using a *t* distribution and a certain confidence level, we calculated an interval to enumerate this amount of time. Hence, we are 99.9% confident that it will take 150 vehicles between $2.6281 < \mu < 3.0847$ minutes to evacuate Georgia Tech campus. Now, this was with the guidance of 6 police officer placed throughout campus. Without the police officers, we can instantly see the amount of time to evacuate campus increase by a factor of about 25%. The average time rose to 209 seconds from the 172 seconds. Now, we would be 99.9% confident that it would take 150 vehicles to $3.2979 < \mu < 3.6988$ minutes to evacuate campus.

Running tests on a smaller number of vehicles displayed the same change. Without police officers, the amount of time increased by 25% again. Thus, we see through this data that the amount of time required to evacuate a set number of vehicles is nearly proportional. For example, increasing the number of vehicles by a factor of 5, also increased the amount of time required to evacuate by approximately a factor of 5.

	Number Of Vehicles	*Amount of Time (s)*	*Standard Deviation*	*Lower Bound (min)*	*Upper Bound (min)*
With Police Officer	*158.52*	*171.39*	*20.58*	*2.63*	*3.08*
Without Police Officer	*160.61*	*209.90*	*18.07*	*3.30*	*3.70*
With Police Officer	*30.23*	*36.45*	*10.07*	*0.50*	*0.72*
Without Police Officer	*31.19*	*45.61*	*11.18*	*0.64*	*0.88*

Figure 6: Confidence Interval Data

Now, in reality there is a much higher number of vehicles that must evacuate GaTech. Since there are an order of thousands of vehicles, our simulation can only go so far to provide estimates for the upper bound on the amount of time required to evacuate. Additionally, there are other factors that have a strong role in this estimate. For example, the accuracy of the map (there may be roads unspecified in this simulation that are actually available for use in daily life), the use of double lane roads vs single lane roads and similarly one-way roads vs double way roads, size of cars (some cars may take up more space than others and so the number of cars allowed on each road at a time is maximized by the size of each car) can all vary this amount of time.

However, that is not to say that we cannot use this simulation to provide fruitful insights. One main point to surely note is that adding police officers to moderate traffic is clearly a better option than letting individuals always follow the FCFS principle. Based on our data, adding police officers has significantly reduced the amount of time required to evacuate even for small instances of number of vehicles on the road.

Conclusion

As we have seen, this simulation provides us the ability to view the behavior of individuals leaving Georgia Tech campus during an emergency. We understand how the simulation maintains a sense of randomness through its DRNG and how it provides us with insight into the amount of time required to evacuate individuals off campus. Through the χ^2 test, we showed that DRNG is quite unbiased towards any value and that it actually does provides unpredictable, statistically independent sequences of uniformly distributed integers. Finally we provided a bound for amount of time to evacuate within a 99.9% level of confidence for a set number of cars. Additionally, regardless of the number of vehicles on the road, placing police officers at the intersections of high volume traffic will surely decrease the amount of time required to evacuate.

Works Cited

Beccario, Cameron. "Cambecc/drnglib." *GitHub*. 9 Feb. 2013. Web. 1 Mar. 2015. <https://github.com/cambecc/drnglib>.

Mechalas, John. "Intel® Digital Random Number Generator (DRNG) Software Implementation Guide." *Intel® Digital Random Number Generator (DRNG) Software Implementation Guide*. 15 May 2014. Web. 2 Mar. 2015. <https://software.intel.com/en-us/articles/intel-digital-random-number-generator-drng-software-implementation-guide>.