Project Checkpoint: CSE 6730 Project 1

Chris Dunlap, Allen Koh, Matt May

Spring 2016

Contents

1 Problem Statement

The efficient evacuation of physical structures is an important problem with a long history of multi-disciplinary study [14]. For this study, we focus specifically on the problem of efficiently evacuating the area around Bobby Dodd Stadium in Atlanta, GA. Bobby Dodd Stadium, the home of the NCAA Division-I Georgia Tech Yellow Jackets football team, has a seating capacity of 55,000 individuals.

Figure 1: Bobby Stadium Stadium seating chart.

Our primary objective in this study is to minimize the pedestrian evacuation time of the stadium and its surrounding area as attendees leave the area following a home football game. We do this by taking a parametric approach, in which we explore the effects of road closures, strategically placed

guidance symbols (i.e., signs), and the “takeover" of certain intersections by law enforcement in promoting an optimal result.

For purposes of this study, we define the system under investigation (SUI) as a rectangular polygon (Fig. 2) surrounding the stadium. To be clear, the SUI is defined as the area inside the polygon but outside the stadium. As pedestrians exit the stadium, they enter the SUI, and as they cross the boundary of the polygon, they exit it.

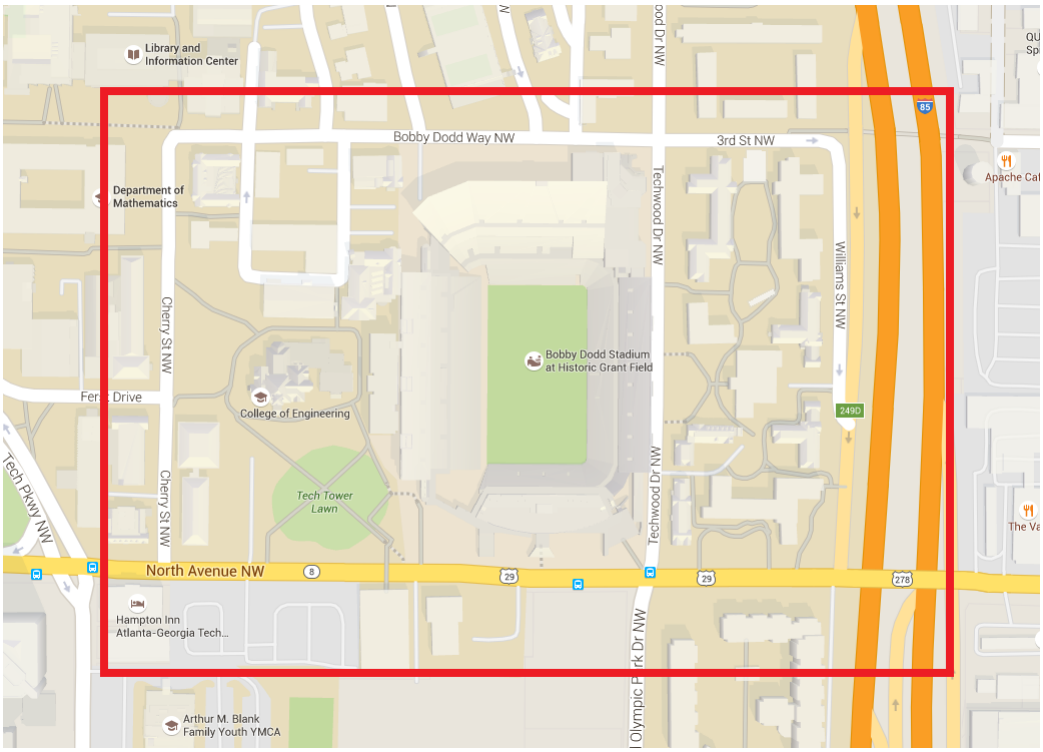

Figure 2: The system under investigation.

The stadium has 10 “gates," which serve as ingress (pre-game) and egress (post-game) sites. These serve as the entry points to our simulation. As pedestrians enter the SUI, they are assigned a destination and then proceed to their destination by way of pedestrian walking paths (i.e., sidewalks).

2 Related Work

A variety of techniques have been used to model pedestrian movement. Cellular automata, lattice gas, social force, fluid-dynamic, agent-based, game-theoretic, and animal experimentation-based approaches have been used [14]. As noted by Zheng, Zhong, and Liu [14], models often encounter or attempt to model several common phenomena: clogging, side-stepping, lane formation, and herding behavior, among them.

These phenomena manifest themselves in different ways and at different magnitudes depending on the system being modeled. In one of the early explorations of 2D cellular automata for simulating traffic flow in the related domain of vehicular traffic simulation, Biham and Middleton [1] note the presence of a sharp *jamming transition* in which all cars in the simulation transition from moving at maximal speed to being stuck. Similar effects were noted in simulations of bi-directional pedestrian movement, with Weifeng, Lizhong, and Weicheng [12] finding that as total pedestrian density increases, a critical value is reached at which the system transitions into a jammed state where only a few pedestrians are able to proceed.

In a slightly different take on the problem, Okazaki and Matsushita [11] modeled pedestrian evacuation using Coulomb's Law, with actors magnetically moving toward their goals and away from obstacles that would lead to collisions. In another application of equations of natural phenomena, Helbing [6] derived fluid dynamic equations for explaining the movement of pedestrian crowds, observing phenomena such as the development of lanes, jamming, and crossing. Helbing [8] further notes an explicit "faster-is-slower" effect, in which attempting to move faster results in a lower average speed of leaving at high pedestrian density in evacuation scenarios.

In a more recent study, Helbing et al. [7] use a "social force" model, which states that pedestrians operate in some sense automatically when reacting to obstacles and other pedestrians, applying strategies that have been learned to be most effective over time. Several suggestions are made to alleviate the three most common problems in pedestrian crowds: counterflows, bottlenecks, and intersecting flows. The presence of strategically placed obstacles are found to actually reduce these negative phenomena, leading to improved flow.

Building on previous cellular automata-based models, Burstedde et al. [3] introduce a *floor field*, a secondary grid of cells which underlies the main grid and acts as a substitute for pedestrian intelligence. These fields can be either

static or dynamic, and are capable of promoting the avoidance of jams, as well as simulating attractive effects, in which pedestrians are more likely to follow in the paths of pedestrians ahead of them.

Taking a more algorithmic approach, Fang et al. [5] have designed a modified ant colony optimization (ACO) algorithm for optimizing pedestrian evacuation, seeking to minimize evacuation time, evacuation distance, and congestion. Kemloh Wagoum, Seyfried, and Holl [9] utilize an observation principle approach in modeling pedestrian evacuation, in which pedestrians first observe their environment and then make a final decision on strategy based on obtained data.

There are several takeaways from the literature described above that have potential applicability to the model at hand. First, on the topic of modeling pedestrian walkways, several papers focused on either a single passageway, enclosed areas with obstacles (walls, pillars, etc.), or evacuation from a room with a single doorway. However, modeling walkways as a directed graph [5] seems feasible given the problem at hand, where there are multiple pathways one can take to get to a single destination. In papers where cellular automata was utilized, the median size of each cell on the walkway was 0.4m^2 [2, 3, 12].

Second, on the topic of pedestrian modeling, multiple walking speed techniques were used, ranging from a constant speed of 1 m/s [12] or 1.3 m/s [3], to using a distribution resembling a step function [2] or a Normal distribution [10]. In either distribution, the median walking speed was also approximately 1.3 m/s. For reaction to stimuli, 0.3 seconds was utilized [3], which was one time step in that particular simulation. Incorporation of opportunistic side-stepping was done [2], in addition to using least-cost algorithms for establishing the path a pedestrian would like to take [5]. In the context of cellular automata, analyzing a given pedestrian's cell's neighbors in either a 4-connected [12] or 8-connected fashion [3] to decide in which direction the pedestrian would like to walk for the next time step was utilized. This, coupled with a dynamic floor field [3] to factor in long-distance forces, could be utilized.

Finally, simulation execution practices were also gleaned. Care must be taken to do parallel-based updating to avoid sequential updating from interfering with underlying model execution [2]. The practice of using a deterministic model running with randomized initial conditions has precedent [1]. Finally, to achieve statistical significance, 20 runs per configuration has been used [2].

3 Conceptual Model

For our simulation, we utilize a 2-dimensional cellular automata (CA)-based approach. We construct a discrete time-stepped model, in which the simulation clock advances by a fixed interval every time step. We utilize a stochastic (probabilistic) approach, in which the output of each individual simulation run is a random variable. Thus, multiple simulation runs will be performed and statistical analysis will be used to provide confidence levels in our results.

3.1 Inputs

The primary input to our simulation model is a stream of football game attendees (i.e., pedestrians) exiting Bobby Dodd Stadium following a home football game. For purposes of modeling this input stream, we assume pedestrian interarrival time at the stadium gate boundary is a homogenous stochastic process that follows a *TODO: Insert* distribution. As pedestrians arrive at the gate boundary, they enter the SUI.

In addition, we select each pedestrian's target destination and walking speed at random. For selecting a target destination, we utilize *TODO: Allen?* For determining a walking speed for each pedestrian, we sample from a distribution previously referenced in the literature by Blue and Adler [2] (5% fast walkers (4 cells/time step), 90% standard (3 cells/time step), 5% slow (2 cells/time step)).

Pedestrians in our model can be considered a consumer entity class *Pedestrian* with attributes which will be set and updated during the course of our simulation. Table 1 provides an overview of the attributes of the *Pedestrian* class.

New instances of the *Pedestrian* class are created and initialized as pedestrians exit the stadium.

3.2 Output

For each simulation run, when all pedestrians were evacuated from the SUI we determined the simulation to be complete. As this is a stochastic simulation, the output results must be interpreted as samples from a random process.

At the conclusion of a series of simulation runs for a given set of parameters, we compute an average of the total egress duration across all runs from

that series. More formally, the average egress time E_{avg} for n simulation runs can be represented as:

$$E_{avg} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{1}$$

where d_i is the total egress duration of the ith simulation run. This output can be considered a derived scalar output variable (DSOV). We then define the optimal parameter strategy to be the set of parameters P that give the the minimum value of E_{avg}.

3.3 Content

3.3.1 Approach

Pedestrians exit the stadium from one of four potential exits. To build the simulation space, we overlay a 2-dimensional cellular automata grid on a map of the SUI. We model this space as a weighted graph, a type of graph in which each graph edge is assigned a weight corresponding to the spatial distance between the two nodes the edge spans [13]. Note that in this section, the terms “cell" and “node" are used interchangeably.

As pedestrians exit the stadium (and thus enter the SUI), they are probabilistically assigned one of the graph’s destination nodes. When a pedestrian

Attribute	Description
Destination	*Node* object (see Table 2) corresponding to the final destination of the pedestrian.
Speed	Walking speed, formulated in grid cells traversed per time step.
Current	*Node* object corresponding to the current location of the pedestrian.
EgressComplete	A boolean value. This value is false when the pedestrian has not yet egressed, and true when egress is complete.

Table 1: Attributes of the *Pedestrian* entity class.

reaches a destination node, that constitutes them exiting the SUI. Destination nodes are chosen based on their spatial meaning when overlaid on a map of the SUI, and constitute one of three possible classes: on-campus housing (dormitories), parking lots, or the North Avenue Metropolitan Atlanta Rapid Transit Authority (MARTA) station.

Each node in the graph is a member of a *Node* class, which has attributes as shown in Table 2.

Attribute	Description
XCoordinate	X-coordinate of the node on the 2-dimensional grid.
YCoordinate	Y-coordinate of the node on the 2-dimensional grid. (Note: for purposes of this simulation, the y-axis is inverted.)
Paths	A dictionary (hash map) relating each possible destination to the NodeID of the next node in the shortest path as determined by Dijkstra's algorithm.
Neighbors	Array of *Node* objects corresponding to nodes that are one edge away from the node.
Available	Boolean value indicating whether the node is available for a pedestrian (true) or not available (false).
Type	Enumeration of the type of node, corresponding to its spatial significance. The Type can be one of four possibilities: sidewalk, road, SUI entrance, or SUI exit.

Table 2: Attributes of the *Node* class.

As part of the simulation initialization, the node and edge information is populated. For each node in the graph, we pre-compute the least-cost path to every possible destination node N_d. In general, we can compute the shortest path between any given node N_i and destination node N_d using Dijkstra's well-known algorithm [4]. For each node in the graph, we compute the least-

cost path to each possible destination node. A hash map *Paths* attribute of each node stores the next node in the least-cost path to each destination.

At each time step, a pedestrian occupying a given node N_i uses the *Paths* attribute to identify the next desired cell to walk towards given their destination node N_d. This drives the pedestrian's instinct to move toward their destination, based strictly on the shortest distance walk from their current position to N_d.

This concept of pre-computing and saving shortest-path data is known as creating a static floor field (that is, a floor field that does not change as the simulation progresses). However, a given pedestrian must decide where to move in light of other pedestrians and other modeled factors. Such factors do change with time, and requires the use of a dynamic floor field [3].

The chosen implementation of the dynamic floor field is to be determined. However, the static floor field will return the next node desired based on the shortest path; due to the 8-connected nature of the nodes this corresponds to a specific direction of travel spatially (for example, North or North East). Thus, it is reasonable to also consider directions adjacent to the "shortest path" direction when deciding a pedestrian's next move (for example, a "shortest path" direction of East would yield adjacent directions of North East and South East).

It will likely occur that more than one pedestrian will decide to pursue the same node in a given time step. In such cases, a probabilistic approach is taken to resolve this conflict (since only one pedestrian can occupy a cell at a time), depending on their respective walking speeds. If the walking speeds are the same, each pedestrian will get an equal chance of getting the desired node. However, if there is a disparity, pedestrian(s) featuring higher walking speeds will have a higher chance of getting the desired node, and pedestrian(s) featuring lower walking speeds will have a lower chance of getting the desired node.

3.3.2 Parameters

TODO: Add parameters.
Walking speed distribution
Time step
Map Vertex file (contains meters/cell datum)
Map Edge file

In addition to a *Node* class, there is also an *Intersection* class, which has attributes as shown in Table 3.

Attribute	Description
Nodes	Array of *Node* objects corresponding to nodes that are part of this intersection.
Open	Boolean value indicating whether the intersection is open for pedestrian access (true) or not open (false).

Table 3: Attributes of the *Intersection* class.

To model intersections where vehicles are allowed to pass, the nodes in the intersections are assigned to a pre-defined intersection. The intersection is therefore made up of a group of nodes. To simulate the crosswalk opening and closing, the nodes will be either forced closed or available for pedestrians to occupy them. If the road is closed and vehicles are passing, then no nodes in the intersection will be available for pedestrians. The duration of intersection closure can be determined stochastically or by the number of pedestrians waiting for the crosswalk to open.

One parameter in the simulation that can be modified is determining which intersections will be completely closed to vehicles and always available for pedestrian use. To model this case, all nodes in that intersection will behave in the same way as the nodes that make up the sidewalk. There will be no need to create an intersection object for that intersection. On the other hand, if there is an intersection that is closed off for pedestrians and only available for vehicular traffic, then the initial creation of nodes will not include that intersection - there will be no nodes at that intersection as no pedestrian can ever occupy those nodes.

If there are pedestrians that need to cross an intersection, but the intersection is closed, then the pedestrians will not be able to advance to their next node. This will cause a list of pedestrians to stay in their current node until the crosswalk opens up (i.e., queue). Once that occurs, then the pedestrians can continue to their next nodes.

3.3.3 Random Numbers

To model the egress of the stadium, we take a stochastic (probabilistic) approach. In the context of a computer simulation, this necessitates the use of a pseudorandom number generator. For this study, we use the simple linear congruential approach. This generator is given a *seed* value, and then produces a uniformly distributed pseudorandom number. This value is then used to sample from a given probability distribution.

3.4 Assumptions and Simplifications

We make a number of assumptions and simplifications in constructing our simulation model.

Firstly, our simulation is simplified by focusing only on *pedestrian* traffic, avoiding the inclusion of vehicular traffic which could be significant in a football game evacuation scenario.

We also assume the stadium to be at or near full seating capacity of 55,000 individuals. While the mean value is somewhat lower than 55,000, the choice of 55,000 allows us to "stress test" our model by simulating the upper bound.

We assume that pedestrians stay on sidewalks and obey traffic lights, as the inclusion of other types of pedestrian behaviors would likely add complexity to the model without yielding much additional value. We also assume that main roads such as North Avenue remain open to vehicular traffic, avoiding a more unrealistic simulation that would be possible if all roads could be closed.

In the realm of our "updating rule" which specifies the next cell for a given pedestrian to travel to in the simulation, we assume that pedestrians are following a shortest-path approach. While logical, this may be an oversimplification, as pedestrians (particularly from the home team) may follow more indirect paths to their destinations following a victory.

We make the assumption that all pedestrians will be traveling to one of three possible classes of destinations - local housing, MARTA, or a parking facility. While an important simplification, it is likely true in reality that pedestrians may be traveling to other destinations as well, such as social venues. It is also likely that some percentage of pedestrians may not evacuate the area for an extended time period, as they sit idle while waiting for friends or planning their destination in an ad-hoc manner.

We assume the stochastic processes in this simulation to be homogenous

Method	Description
Move(NewNode)	Takes one argument: a *Node* object. When called, marks the node set to *Current* as unavailable, sets the *Current* attribute to NewNode, and marks the NewNode object as unavailable.

Table 4: Methods of the *Pedestrian* class.

stochastic processes, which can be defined as stationary stochastic processes where the random variables $X = X_1, X_2, ..., X_n$ are independent and identically distributed. While not always realistic in the context of simulation model-building, we do believe this assumption to be reasonable in the context of this simulation.

There are inherent simplifications in the choice of a cellular automata-based simulation approach. Specifically, it is assumed that each individual travels a constant distance per time step through the simulation. Nevertheless, we attempt to mitigate this simplification by drawing the speed for each individual from a probability distribution, which does introduce some variation, albeit only on an individual-to-individual basis.

4 Description of Simulation Software

4.1 Architecture

In generating our simulation, we take an object-oriented design approach. We use the Python programming language, which is an imperative, object-oriented, interpreted language that facilitates rapid development.

Distinct types of entities in our simulation are represented as *classes*. For interacting with these classes, we define a number of methods that function as their public interfaces.

4.2 Interfaces

4.2.1 Pedestrian Class Interfaces

Pedestrians class in Table 4.

Method	Description
GetAvailable()	Returns the current value of the *Available* attribute for the node.
SetAvailable(bool)	Takes one argument: a boolean value. Sets the *Available* attribute for the node to either true or false.

Table 5: Methods of the *Node* class.

Method	Description
OpenMe()	When called, sets the *Available* attribute to true for every node in its *Nodes* array. Sets the *Open* attribute for the intersection to true.
CloseMe()	When called, sets the *Available* attribute to false for every node in its *Nodes* array. Sets the *Open* attribute for the intersection to false.

Table 6: Methods of the *Intersection* class.

4.2.2 Node Class Interfaces

Node class in Table 5.

4.2.3 Intersection Class Interfaces

Within our simulation, we utilize a number of intersections that are parametrically closed and opened. Thus, we define two key methods for the *Intersection* class in Table 6.

References

[1] Ofer Biham, A Alan Middleton, and Dov Levine. Self-organization and a dynamical transition in traffic-flow models. *Physical Review A*, 46(10):R6124, 1992.

[2] Victor J Blue and Jeffrey L Adler. Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B: Methodological*, 35(3):293–312, 2001.

[3] Carsten Burstedde, Kai Klauck, Andreas Schadschneider, and Johannes Zittartz. Simulation of pedestrian dynamics using a two-dimensional cellular automaton. *Physica A: Statistical Mechanics and its Applications*, 295(3):507–525, 2001.

[4] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[5] Zhixiang Fang, Xinlu Zong, Qingquan Li, Qiuping Li, and Shengwu Xiong. Hierarchical multi-objective evacuation routing in stadium using ant colony optimization approach. *Journal of Transport Geography*, 19(3):443–451, 2011.

[6] Dirk Helbing. A fluid dynamic model for the movement of pedestrians. *arXiv preprint cond-mat/9805213*, 1998.

[7] Dirk Helbing, Lubos Buzna, Anders Johansson, and Torsten Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. *Transportation science*, 39(1):1–24, 2005.

[8] Dirk Helbing, Illés Farkas, and Tamas Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490, 2000.

[9] Armel Ulrich Kemloh Wagoum, Armin Seyfried, and Stefan Holl. Modeling the dynamic route choice of pedestrians to assess the criticality of building evacuation. *Advances in Complex Systems*, 15(07):1250029, 2012.

[10] Hubert Klüpfel, Michael Schreckenberg, and Tim Meyer-König. Models for crowd movement and egress simulation. In *Traffic and Granular Flow'03*, pages 357–372. Springer, 2005.

[11] Shigeyuki Okazaki and Satoshi Matsushita. A study of simulation model for pedestrian movement with evacuation and queuing. In *International Conference on Engineering for Crowd Safety*, pages 271–280, 1993.

[12] Fang Weifeng, Yang Lizhong, and Fan Weicheng. Simulation of bi-direction pedestrian movement using a cellular automata model. *Physica A: Statistical Mechanics and its Applications*, 321(3):633–640, 2003.

[13] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[14] Xiaoping Zheng, Tingkuan Zhong, and Mengting Liu. Modeling crowd evacuation of a building based on seven methodological approaches. *Building and Environment*, 44(3):437–445, 2009.