TU WIEN Mathematical Modelling and Simulation

Emergency Building Evacuation

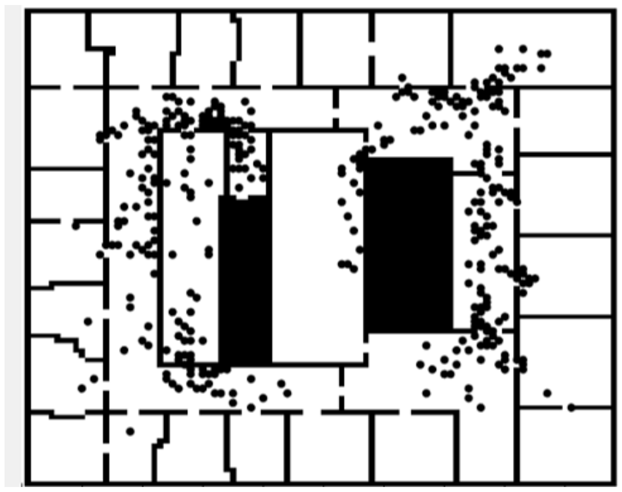

Figure 1: Snapshot of an evacuation simulation

Motivation

The design of residential buildings is a multifaceted process with many constraints. The safety of residents is one of them. For example, it must be ensured at all times that residents can quickly leave the building in the event of an emergency such as fire or smoke. In this context, a valid simulation model can assist in the building planning process. It allows for comparisons between different floor plan layouts or different structures for corridors and door areas in terms of evacuation time even before the building is constructed.

Model

There are a variety of different modelling approaches for simulating the evacuation of a building. We will apply an approach from Tanimoto et.al. 2010 [1] based on a cellular automaton / agent-based model. We explain it by defining model initialisation and dynamics.

0.0.1 Initialisation:

We consider a map of the floor plan as input to the model and assume that the map is available as an image with $M \times N$ pixels whereas each pixel is assumed to be $0.5 \times 0.5[m^2]$. We consider three states/colours of the pixels, that is, white (floor), black (obstacle/wall) and red (exit).

In the first step of the initialisation, a so called static-floor-field (SFF) is established. That means, each white and red pixel in the floor plan is assigned a value indicating its moving-distance towards the most nearby exit.

The SFF, henceforth regarded as matrix $S \in (\mathbb{R} \cup \{\infty\})^{M \times N}$ is constructed iteratively. First, for all i, j which refer to red pixels in the floor plan, we assign $S_{i,j} = 0$, because they are, by definition $0[m]$ away from the exit. For all other i, j we initialise $S_{i,j} = \infty$. Henceforth the following process is repeated until new further progress is made: Let $I := \{(i_1, j_1), \ldots, (i_n, j_n)\}$ be a set of index/coordinate pairs, so that (a) the corresponding pixel in the floor plan is white and (b) any of the eight neighboured indices (Moore Neighbourhood) has a value smaller than ∞. Then, for any $(i, j) \in I$, define

$$q_{i,j} := \min\left(\min(S_{i+1,j}, S_{i-1,j}, S_{i,j+1}, S_{i,j-1}) + \frac{1}{2}, \min(S_{i+1,j+1}, S_{i-1,j+1}, S_{i+1,j-1}, S_{i-1,j-1}) + \frac{\sqrt{2}}{2}\right).$$

That means, the distance of cell (i, j) to the exit is by x farther than the distance of the neighbour cell, whereas x is the euclidean distance between the two cell-midpoints. As soon as this computation

[1] *Study of bottleneck effect at an emergency evacuation exit using cellular automata model, mean field approximation analysis, and game theory*, https://doi.org/10.1016/j.physa.2010.08.032

is finished for all $(i,j) \in I$, update the SFF matrix via $S_{i,j} = q_{i,j}$. That means, the matrix has to be updated for all indices I simultaneously.

At the end of this procedure, S contains value $< \infty$ for all coordinates on the floor plan from which an exit is reachable. It is universally valid for each floor plan and only has to be computed once, independent of the actual evacuation simulation.

In the final step of the initialisation, we create $N \in \mathbb{N}$ agents and assign them a random coordinate (i,j) associated with a white floor pixel. Note that spots can only be occupied once.

Dynamics:

All agents are updated simultaneously and in equidistant time steps. Every time-step, every agent who has not yet reached the exit is addressed once:

- Let (i,j) refer to the current coordinate of the agent, compute the weight matrix
$$\forall k,l \in \{-1,0,1\}, k \vee l \neq 0 : P_{k,l} = e^{-kS_{i+k,j+l}},$$
whereas k is a positive model parameter.
- Draw a random new target coordinate $(i',j') := (i+k, j+l)$ weighted by the probability matrix $P_{k,l}$. Note that either k or l must be non-zero, so the agent always wants to move.

Since agents cannot occupy the same spot twice, a collision rule must be applied. For all new target coordinates (i',j'):

- Compute how many agents want to move to the target coordinate, say n.
- If $n = 1$, the corresponding agent moves to the destination.
- If $n > 1$, compute
$$\mu(n) = 1 - (1-\xi)^n - n\xi(1-\xi)^{n-1}$$
Hereby, ξ is a model parameter to capture potential collisions between the agents.
- With probability $\mu(n)$ the agents are assumed to block themselves and eventually none of the agents can move. Otherwise one of the n agents is randomly selected to move.

Any agent who moved to place with $S_{i',j'} = 0$ is considered to have left the building and is removed from the model.

ToDo's

Task 1

Draw several (fictional) maps and write a program to compute the static floor field (SFF). Use an interface file format to store precomputed SFFs. Validate your computation by visualisations of the SFF. Compute the gradients $S_{i,j} - S_{i-1,j}$ and $S_{i,j} - S_{i,j-1}$ and interpret them as vector field pointing towards the exits. Make plots displaying the vector field.

Task 2

Implement the actual agent-based model as defined above. Make an animated visualisation to verify that the model produces reasonable results.

Task 3

Think about how you could measure the "escape-robustness" of floor field quantitatively. Note that both, the initial configuration and the model dynamics, involve randomness. Thus, Monte-Carlo simulation must be performed.

Task 4

Vary the model parameters ξ and k and investigate their impact on the simulation results qualitatively and quantitatively. Vary the internal layout of one of the floor-plans and investigate what structures help or hinder fast escape routing.

Task 5

Make your model more realistic. Examples are: Use a floor plan of an existing building, use a more realistic initial agent-distribution, introduce heterogeneity among agents, introduce a more goal oriented behaviour etc.

Task 6

Document your findings in a written protocol.

Programming Language

Any programming language / simulator that supports ABM and animated outputs (e.g. Python, MATLAB, Java, Netlogo, AnyLogic, . . .).