**Conceptual Model**

The model is based upon the cellular automata model shown in the paper [1]. It’s different in that it doesn’t implement the panic mechanism where there is a percent chance that a person won’t take an action.

**Formal Specification**

CD = <X, Y, I, S, θ, N, d, δint, δext, τ, λ, D>

X = Ø

Y = Ø

I = Ø

S = {float exitProximity, int type, int nextDestination, int directionIncoming, double sigma}

Empty: type=0

Person: type=1

Obstacle: type=2

nextDestination = {1, 2, 3,

4, 5,

6, 7, 8}

directionIncoming = {1, 2, 3,

4, 5,

6, 7, 8}

Sigma = output delay

N = {(-1,-1), (0, -1), (1, -1), (-1, 0), (1, 0), (-1, 1), (0,1), (1,1)}

d = 100 ms

τ = N -> S

The local computation rules in cadmium v2 are difficult to express as simple CD++ rules so I have included them as pseudo code.

**Rule 1**

If (s.type < 2) and (s.exitProximity == 0) then

For (neighbor in neighbors) :

If (neighbor is diagonal to cell) then

Append neighbor.exitProximity + 1.5 to list

Else if (neighbor is vertical or horizontal to cell) then

Append neighbor.exitProximity + 1 to list

s.exitProximity = minimum of list

endif

**Rule 2**

if (s.type == 1) and (s.nextDestination == 0) then

for (neighbor in neighbors) :

if (neighbor.type == 0) then

if (neighbor.exitProximity != 0) then

Append neighbor.exitProximity and nextDestination direction to list

Else then

Clear the list

Exit the loop

Find smallest exitProximity in list

If (smallest exitProximity < s.exitProximity) then

Find neighbors that have the smallest exitProximity

If more than 1 neighbor has smallest exitProximity then

Randomly choose between them

s.nextDestination = direction to chosen neighbor

s.sigma = 1

else

s.nextDestination = 0

s.sigma = 1

endif

**Rule 3**

If (s.type == 1) and (s.nextDestination != 0) then

For (neighbor in neighbors) :

If s.nextDestination points to neighbor’s relative position then

If (s.nextDestination + neighbor.directionIncoming == 9) then

s.type = 0

s.sigma = 1

s.nextDestination = 0

else if (neighbor.directionIncoming == 0) then

s.type = 1

s.sigma = 1

else if (s.nextDestination + neighbor.directionIncoming != 9) then

s.type = 1

s.sigma = 1

s.nextDestination = 0

endif

**Rule 4**

If (s.type == 0) and (s.directionIncoming == 0) then

For (neighbor in neighbors) :

If (neighbor.type == 1) and (neighbor.exitProximity > s.exitProximity) then

If (neighbor.nextDestination != 0) then

If (neighbor.nextDestination points to this empty cell) then

Append direction to neighbor to list

Else if (neighbor.nextDestination == 0) then

Clear list

Exit the loop

If more than 1 neighbor.nextDestination points to this empty cell then

Randomly select a neighbor from the list

s.directionIncoming = direction to chosen neighbor

s.sigma = 0

endif

**Rule 5**

If (s.type == 0) and (s.directionIncoming != 0) then

s.directionIncoming = 0

s.type = 1

s.sigma = 1

endif

**Rule 6**

If (s.exitProximity == 1) and (s.type == 1) then

s.type = 0

s.sigma = 1

endif

**Testing**

To test the model, I first ran it with a very simple scenario and verified the behaviour of the model. In this scenario it is a 5 by 5 grid with the border cells as obstacles to represent the walls of a room. Only the cell at (4, 2) is open and assigned to the be the door with an exit proximity of 1. Two people are placed in the room with one at cell (2, 2) and the other at cell (3, 2). At time 0 the exit proximity values of each cell are set as they increase the further they are from the exit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 500 | 500 | 500 | 500 | 500 |
| 500 | 4.5 | 4 | 4.5 | 500 |
| 500 | 3.5 | 3 | 3.5 | 500 |
| 500 | 2.5 | 2 | 2.5 | 500 |
| 500 | 500 | 1 | 500 | 500 |

Figure 1: Exit Proximity values of each cell for scenario 1

At time 0 the people select what direction they are going to travel in. This is represented by numbers from 1 to 8 as can be seen in figure 2.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 |  | 5 |
| 6 | 7 | 8 |

Figure 2: Directions around a cell represented by integers

Figure 3 shows where the people are with a “P” and a number to indicate the direction they have chosen. The walls with exit proximity values of 500 are represented by black cells. The exit is represented by a green cell. It can be seen that the person in cell (2,2) has chosen to go to cell (3,3) as the person in cell (3,2) is blocking the most direct route to the exit. The person in cell (3,2) however is right next to the exit so they are heading down towards the exit at cell (4,2).

A black and white grid with green and white squares

Description automatically generated

Figure 3: Positions of people at time 0 for scenario 1

Figure 4 shows the people’s new positions at time 1.

A white and green squares with black text

Description automatically generated

Figure 4: Positions of people at time 1 for scenario 1

At time 2 the person who is at the exit doesn’t need to select a direction as they have made it out of the room, but the person at cell (3,3) does need to select a direction. As the person is still in the exit and blocking it, they select cell (3,2). The person at the exit simply leaves the room. This can all be seen in figure 5.

A black and white grid with green and white squares

Description automatically generated

Figure 5: Positions of people at time 2 for scenario 1

In figure 6 you can see at time 3 the person moves to cell (3, 2)

A white and green squares

Description automatically generated

Figure 6: Positions of people at time 3 for scenario 1

In figure 7 you can see at time 4 the person chooses to move to the exit at cell (4,2).

A white and green squares

Description automatically generated

Figure 7: Positions of people at time 4 for scenario 1

In figure 8 you can see at time 5 the person moves to the exit.

A white and green squares

Description automatically generated

Figure 8: Positions of people at time 5 for scenario 1

And lastly, in figure 9 the room is empty as the final person has left the room and simulation ends.

A black and white square with green squares

Description automatically generated

Figure 9: Positions of people at time 6 for scenario 1

From running this scenario a few times I can see how people correctly select the best cell to move to, and how they randomly choose between their destinations if they are an equal distance to the exit.

To view the behaviour of two people who wish to enter the same cell I have run a slightly different scenario where the only difference is that the 2 people are placed in cells (2,1) and (2,3). This ensures that they will both want to reach the same cell at (3,2) to get to the exit faster. For the sake of brevity, I will show the grids with just their associated time steps.

A white and green rectangle with black text

Description automatically generated

A white and green squares

Description automatically generated

A white and green squares

Description automatically generated

A white and green squares

Description automatically generated

A black and white square with green squares

Description automatically generated

A screen shot of a computer

Description automatically generated

A white and green squares

Description automatically generated

A white and green squares

Description automatically generated

A black and white square with green squares

Description automatically generated

Figure 10: Positions of people from time step 0 to 8 for scenario 2

In figure 10 from the start the two people wanted to enter the same cell as it was closest to the exit. However, the person in cell (2,3) won the random selection and blocked the person in cell (2,1) from entering cell (3,2). This concludes testing as all the behaviours are functioning correctly.

**Simulations**

Now to run a more interesting simulation I will keep the 5 by 5 grid as it’s easy to interpret and display, but I will add 7 people to the room in positions (3,1), (3,3), (2,2), (2,1), (1,1), (1,2), and (1,3). The results of the simulation at each time step starting from 0 can be seen in figure 11.

A white and green rectangular box with black text

Description automatically generated

Time Step: 0

A white and green squares with black letters

Description automatically generated

Time Step: 1

A white and green squares with black text

Description automatically generated

Time Step: 2

A white and green squares with black letters

Description automatically generated

Time Step: 3

A white and green squares with black text

Description automatically generated

Time Step: 4

A white and green squares with black letters

Description automatically generated

Time Step: 5

A white and green squares with black text

Description automatically generated

Time Step: 6

A white grid with black and green squares

Description automatically generated

Time Step: 7

A white and green squares with black text

Description automatically generated

Time Step: 8

A white and green squares with black letters

Description automatically generated

Time Step: 9

A white and green squares with black text

Description automatically generated

Time Step: 10

A white and green squares with black letters

Description automatically generated

Time Step: 11

A white and green squares

Description automatically generated with medium confidence

Time Step: 12

A white and green squares with black text

Description automatically generated

Time Step: 13

A white and green squares with black text

Description automatically generated

Time Step: 14

A white and green squares with black text

Description automatically generated

Time Step: 15

A white and green squares

Description automatically generated

Time Step: 16

A white and green squares with black text

Description automatically generated

Time Step: 17

A white and green squares

Description automatically generated

Time Step: 18

A white and green squares

Description automatically generated

Time Step: 19

A black and white rectangle with green squares

Description automatically generated

Time Step: 20

Figure 11: Positions of people at each time step for scenario 3

The larger number of people interacting in this scenario has shown some interesting behavior. In time steps 18, 16, 14, 10, 6, and 2 the people closest to the exit already recognize that no one is blocking them, so they attempt to enter the exit at cell (4,2). However, in time steps 12, 8, and 4 they don’t seem to recognize this and try to enter the next closest cell at (3,2). This is likely because there isn’t an order to what cell will fire it’s local computation function next, so in the first case the exit cell fired first, but in the second case the exit cell fired last. In the future I would aim to fix this so that the behavior is consistent.

**References**

1. A. Varas, M.D. Cornejo, D. Mainemer, B. Toledo, J. Rogan, V. Muñoz, J.A. Valdivia, “*Cellular automaton model for evacuation process with obstacles,*” Physica A: Statistical Mechanics and its Applications, Volume 382, Issue 2, 2007, Pages 631-642, ISSN 0378-4371, https://doi.org/10.1016/j.physa.2007.04.006.