AN AGENT-BASED APPROACH FOR OPTIMIZATION OF BAR ARRANGEMENTS WITHIN NIGHTLIFE VENUES

Bas Chatel, Alireza Faridamin, Mike la Grouw, Sara Neven and Viktor van der Valk

***Abstract*— Nightlife venues want to optimize the balance between income and customer-satisfaction. This satisfaction partially depends on perceived crowd-density and on waiting times at the bar. Agent-based models have shown to be of use in various crowd dynamic studies. Therefore this paper introduces an agent-based approach to model crowd in nightlife venues. This model is then used to analyze waiting time and crowd dynamics for different placements of points of interests like bars within a virtual nightlife venue. Optimal bar-placement is shown to be in the middle of the venue, where the combination of crowd and distance to the bar is minimal.**

I. INTRODUCTION

In nightlife venues long queues at bars and a crowded stage area are often considered unpleasant, while the income of the venue increases with the number of visitors. Because of this, nightlife venues struggle with deciding a maximum number of people and arranging the places of stages and bars to optimize safety and a pleasurable experience for visitors. A helpful tool in making these decisions could be modeling crowds within venues using computer models. An agent based approach in which agents represent visitors of a concert-like venue in which they move to stages and bars is an example of these models. This could be a way to research the effect of arrangements of bars and stages on overcrowding of the stage and waiting times for drinks. Such an agent-based model could be used to help designing venues and optimizing the experience of visitors of busy bars and concerts, while still guarantying sales volume for the venues.

An agent-based model is a computational model in which agents in an environment interact with each other and/or the environment. Agent based models are algorithmic models where the actions of agents and state changes of agents and environment follow a specific set of rules. Agents typically sense their surrounding environment and other agents, decide which action to take based on the sensed data and carry out the action. The environment in an agent based model is a multi-dimensional space in which the agents move and interact. Agent-based models are widely used to examine complex interactive systems such as segregation [1] and crowd dynamics at dangerous and busy places such as the Jamaraat bridge during the Hajj in Mecca [2].

As crowd-dynamics emerge from the behavior of individuals, human decision-making processes should be taken into account [3]. Humans in bars prefer a certain personal space around them, and waiting in line usually has a negative impact on the individual's happiness. In our design, the approach is not only to find realistic crowd-dynamics, but make the decision-making process similar to that of a human being. Because the model is of a single night, the behavior is kept simple. This simplistic view of a club-night should be enough to show real-life crowd-dynamics.

The model described in this paper is an elaboration of the classic Sugarscape-model [4]. In this model a grid with sugar-containing cells (that can grow to a certain maximum value) create a gradient on which agents can move. Agents in this model need to harvest and consume sugar to stay alive. In each time step, they move to the most sugar-rich position available within their vision range. In simulations of the Sugarscape model, agents typically tend to cluster in sugar-rich parts of the environment (sugar mountains). In the nightclub-model, the agents want to move as close as possible to the stage and the bars.

To examine the influence of bar placement on average waiting time and crowd forming at a concert-like venue, an agent-based model was constructed. In the model an artificial Sugarscape-like gradient was created by assigning score-functions to the bar and stage areas. These score functions were conducted by closeness to the point of interest and crowd density (since in real life it is hard to take a path through a crowded area and people tend to avoid overly crowded areas). Four bars in pairs of two standing opposite of each other at the far ends of the x-axis were placed on different y-coordinate positions. The average waiting time for agents was measured, as was the maximum amount of agents in one grid cell. A sensitivity analysis was conducted to see if the waiting time and crowd-forming was sensitive to bar placing, serving speed at bars and other input parameters.

It is expected that the placings of bars and serving speed will influence the waiting time and crowd-forming within the model. Furthermore it is expected that placing bars closely to the stage will increase overcrowding (since the stage- and bar-crowds will overlap). Placing the bars further away from the stage will result in less overcrowding. Placement of the bars close to the stage or very far from the stage is expected to increase the waiting time. This is expected because in the former situation the overcrowding can result in long waits and in the latter case it takes longer to commute to the bars.

The code from the model was built on the Sugarscape-model [5] Certain aspects from this code were used as a basis for our nightlife-model. The model description follows the ODD (Overview, Design concepts, Details) protocol [6] [7].

II. MODEL OVERVIEW

A. Purpose

The purpose of the model is to find the optimal arrangement for a pre-defined concert venue. This optimal arrangement is the one where the time spent on getting beer is low, so people spend most of their time listening to the music at the stage and little overcrowding is present. The stage will be kept at the same placement in the venues during simulations. Because of this the optimal arrangement is best defined as placement of the bars.

B. Entities, state variables and scales

The model comprises the following entities:

- *Agents*: The agents are individual humans, with a respective beer-need. Each agent occupies a maximum of one grid-cell, but they are able to share this cell with other agents. Each agent can have two different modes: a stage-mode and a bar-mode. All agents share the same preference for crowd-density, walking-distance and waiting-time. Each agent drinks beer with a standardized consumption rate.
- *Spatial units*: The grid cells represent a concert-area, containing a stage and bars. Each grid-cell represents a space of 0.75 meter by 0.75 meter, and the model concert-venue comprises 25.5 meter by 13.5 meter (34 by 18 grid cells). Some grid cells represent bar or stage areas
- *Bars*: There are four bars divided in two pairs, bar 1 and 3 and bar 2 and 4. These pairs of bars are located opposite to each other, they share their y-coordinate, but their x-coordinate is respectively the smallest or largest x-value on the grid. The y-coordinates of the bar pairs are the input parameter for bar position.
- *Stage*: The stage is located in the middle of the upper narrow side of the venue (y = 33).
- *Time*: One time step represents 1 second and simulations were run for 1/2 hour (1800 time steps).

C. Process overview and scheduling

Time is modeled as discrete steps. In every time step, each individual can move towards their desired goal. The individual's moving-cycle is represented by Figure 1. The order in which the agents act is random. For each agent, the act starts with an update of their thirst. After this, they select either stage-mode or bar-mode and move accordingly, see figure 1.

III. DESIGN CONCEPTS

A. Basic principles

In our model basic Sugarscape-agents are used that are traveling over a grid and try to find a grid cell with minimal score. The score-functions are constructed by evaluating the grid cells within a vision of 2 around the agent, based on crowd in that grid cell, the distance to a point of interest (POI) and crowd at the PO. The agents create an artificial Sugarscape on which the agents travel, finding the best place to go following the gradient. When the agents are in stage-mode their point of interest is the stage. When they are in bar-mode, they evaluate all the bars and select one. This bar has the best score, dependent on crowd and distance. If two bars have the same score, an agent chooses randomly between these.
This model tries to find the optimal arrangement of a concert-venue, based on the crowd-dynamics and customer waiting-times. The behavior of the agents was kept simple. Agents want to go to the stage or to a bar. As simulations are run for 1/2 hour, other behavior is not taken into account.

B. Emergence

From the agent's individual behavior, the crowd-dynamics emerge. The number of agents per cell are expected to change with the agent's preference for crowd-density.

C. Adaptation

In the initial stage-mode the agents will try to move as closely as possible to the stage within their environment. The agents change their mode, deciding to go to a certain point of interest (bar) when they are thirsty. Agents are helped at the bar according to a serving speed parameter and the busyness of the respective grid cell. Agents adapt their mode from stage- to bar-mode due to a thirst counter, which is initialized randomly and goes up according to a heterogeneous normally distributed beer-consumption parameter. Once agents are helped at a bar, the thirst-counter will go back to 0, and they will return to stage-mode.

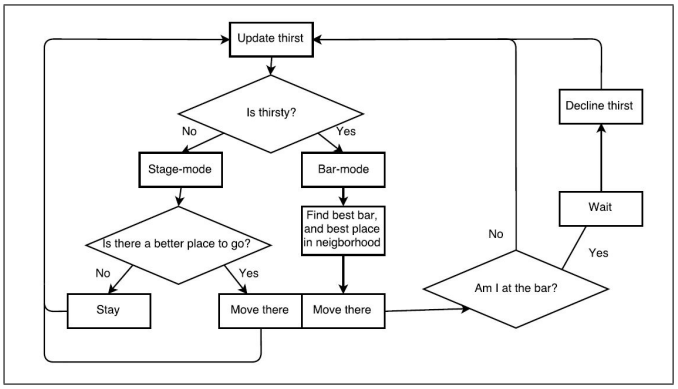

Fig. 1. Agent-behavior according to the Nightlife model

D. Objectives

The objectives of the agents include getting as close as possible to the stage and getting to the bar when they get thirsty. To get as close to the stage as possible is an ongoing objective when the agents are in stage-mode and the objective of getting a drink is succeeded when the agents get to a bar and are helped. When agents are thirsty they decide to go to one of the multiple bars. Simultaneously with trying to achieve their objectives, agents try to avoid overcrowding.

E. Prediction

Agents predict the waiting time at the multiple bars when they get thirsty. Decisions of agents on which bar to travel to is made only by following the artificial Sugarscape bar-gradient. The gradient is influenced by the busyness of the different bars. This can result in the prediction that a bar closer to an agent has a longer waiting time than a bar further away. When deciding which bar to go to, agents predict that the less-crowded areas are better to go to than the crowded ones.

F. Sensing

The agents are able to sense if they are thirsty, when deciding to go to a bar. Agents are also able to sense a artificial grid which represents closeness to stage or bar to decide in which direction they want to move to obtain their objectives. Other agents present in the grid cells within the vision range of an agent are sensed, and their presence is considered when making a decision to move. Agents are able to "see" how busy the multiple bars are.

G. Interaction

When agents are freely walking, they move at a speed of two grid cells per time step. This represents 1.5 m/s [8]. If there are more than six other agents in the same grid cell or in the Moore neighborhood of an agent, the agent moves at half that speed.

H. Stochasticity

Agents start of with a random thirst counter between 0 and 1 and an individual beer consumption rate, which is normally distributed over agents with an average of 0.0008333 and standard deviation of 0.0001. The thirst counter increases with the value of the beer consumption rate plus a random value between 0 and 2, in every time step.

I. Observation

From the model, the average waiting time and maximum number of agents in one grid cell are measured.

- *Waiting time*: For each agent, the waiting time increases with one per time step spent in beer-mode. When the agent buys a beer, the waiting time goes back to zero. When an agent is in stage-mode, the waiting time is 0 as well. Per time step, the mean of all agent's waiting time is calculated (including the agents with waiting time of 0). The average waiting time is the mean over all these means.
- *Maximum crowd*: The maximum crowd is measured over all the night. Per grid cell, the highest number of agents per cell is registered, and a plot is made. This plot is a representation of the night, so the maximum values are not necessary from the same time step.

IV. DETAILS

A. Initialization

For each simulation, the venue was initially occupied with 750 agents. These agents enter from a entrance at the bottom, from where they disperse into the area. Each agent has a thirst-counter, which is a random value between 0 and 1. Each agent has an individual beer-consumption rate. This is rate is taken from a normal distribution with an average of 0.0008333 and a standard deviation of 0.0001. The average value is taken from the assumption that people drink an average of 3 beers per hour (0.0008333 per second).
The agents have the same values for disliking crowds, walking-distance and waiting. These values are 1.9, 1.0 and 0.5 respectively, and are found by facial validation of the model in Mesa [5].

B. Input data

The grid size and number of agents was based on the size and maximum capacity of the Melkweg venue in Amsterdam. The bars are placed on the left and right boundaries of the grid. Their exact locations on the y-axis vary between simulations. The bars have a serving-speed of 0.1 per second, which is 360 beers per hour.

C. Submodels

- *Initiation, mode-choice and counter update*: After initiation all agents enter the venue from one place and are created with an random beer-need counter and an individual consumption rate. The first thing agents do in a time step is updating the thirst-counter. This counter goes up by the beer consumption parameter times a random number between 0 and 2. Secondly the agents decide if they are thirsty or not. This is simply decided by checking if the beer-need counter is higher than one. If this is the case agents switch from stage- to bar-mode, else they stay in stage mode.
- *Check neighborhood and move*: After these decisions the agents start checking their surroundings for a place to move, which brings them closer to their objective. In both modes the agents give a score to cells within their reach and move to the one with the lowest score.
  Scores in stage mode are given by the formula $\beta_d *$

$distance + \beta_c * crowd$. In this formula β_d is the walking-distance parameter. Distance is the Euclidean distance between the agent and the stage. β_c is the crowd-disliking parameter and the crowd is the amount of agents already present in the cell. In bar-mode the score is calculated by the formula $\beta_d * distance + ad\beta_c * crowd + \beta_w * bar - crowd$.
$ad\beta_c$ is the adjusted crowd- disliking parameter calculated as $\beta_c - 0.05*$ time steps in bar-mode, with a minimum of 1.3. This is done to give the agents an incentive to get out of the crowd, to get to a bar. This can be seen as the impatience or increased willingness to move toward more crowded areas when trying to reach a bar. β_w is the waiting-time parameter and bar-crowd is the amount of agents within a space of 4 by 4 grid cells around the bar. The score function is calculated concerning all four bars for every position within agent's reach. The lowest score of all four bar-formulas is than assigned to that location. After the scores are assigned to the positions within the agent's reach, the agent moves to the position with the lowest score.

- *Waiting and being served*: When in stage mode the agent keeps trying to go to a better place in front of the stage following the score function. If agents are in bar mode however they check if they are next to a bar every time step. If they are at a bar, a helped counter is introduced which goes up by the serving speed parameter divided by the number of agents at the bar. If the helped counter is one or higher, the agent is being helped and goes back to stage-mode. The beer need counter is set to 0 again. The first six time steps after being helped the agent is forced to move and does not mind moving to a crowded grid cell, so they can move out of the queue more easily.

V. SIMULATION EXPERIMENTS

An experiment was conducted to examine the effect of position of the bars on average waiting time and crowd forming. Simulations of the model were run with different position of the four bars. The four bars were divided into two pairs. The pair of bars were placed opposite of each other on the edges of the venue (same y-coordinates and x-coordinates of 0 or 17). Simulations were run for every possible combination of y-coordinates of the pairs of bars (1 to 31). An other set of simulations, in which just one pair of bars with double serving speed, was also run for all the possible y-coordinates of the bars. In these runs the average waiting time and the maximum amount of agents in one grid cell over the entire simulation were measured. A sensitivity analysis was conducted to examine the average waiting time and the maximum agents per grid cells sensitivity on differences in the models input parameters. Sensitivity analysis was conducted on a subset of the parameters, using Saltelli Sampling in SAlib [9]. Total and first order Sobol indices were calculated for the following parameters:

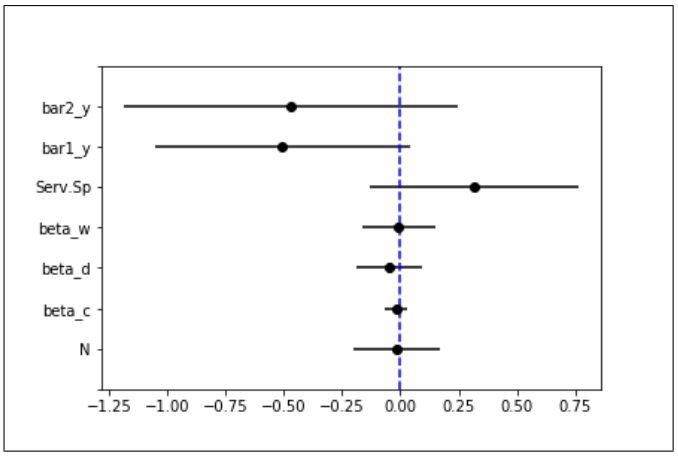

Fig. 2. Sobol sensitivity analysis of the average waiting: First-order sensitivity index for the influence on average waiting time per input parameter.

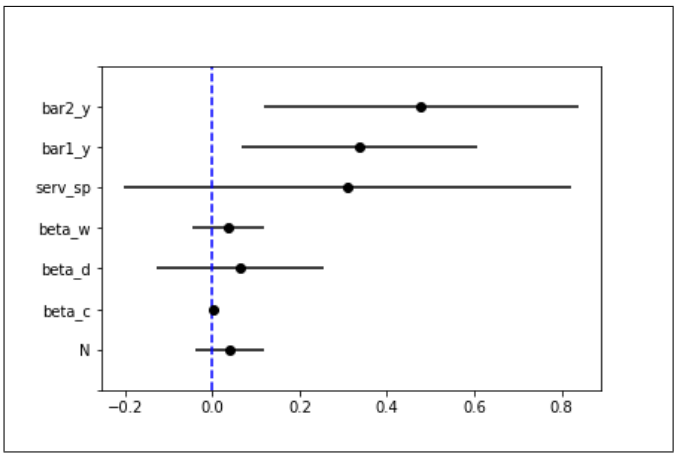

Fig. 3. Sobol sensitivity analysis of the average waiting: Total-order sensitivity index for the influence on average waiting time per input parameter.

Population size, (N), β-crowd, β-distance, β-waiting time, serving speed and the y-coordinates of the 2 bar-pairs.

VI. RESULTS

Figure 2 and figure 3 show respectively the first and total order sensitivity index for the waiting time of all parameters. The placement of the bars and the serving speed have profound effects on the average waiting times, even tough the confidence intervals are still quiet large, especially for the serving speed parameter. Other input parameters seem to have little to no effect on the average waiting time.

In figure 4 and figure 5 the first and total order sensitivity index for the maximum amount of agents per grid cell per parameter is shown. It can be concluded that the β distance parameter has the most influence on the maximum amount of agents per grid cell. All parameters show a possible influence on the amount of agents per grid cell, but the confidence intervals of the analysis are to large to determine

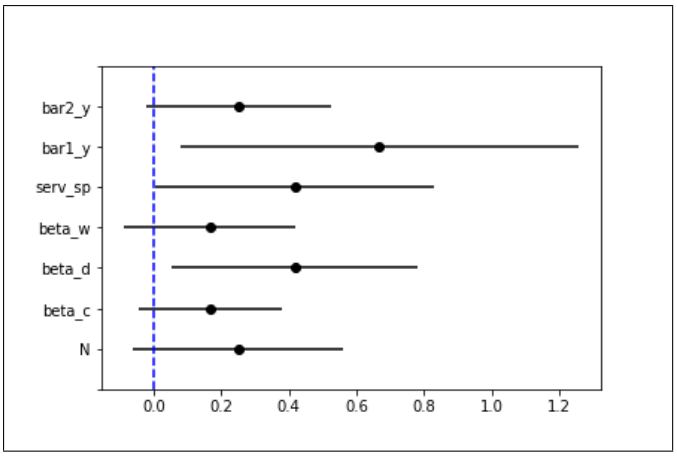

Fig. 4. Sobol sensitivity analysis of the maximum agents per grid cell: First-order sensitivity index for the influence on the maximum number of agents per grid cell per input parameter.

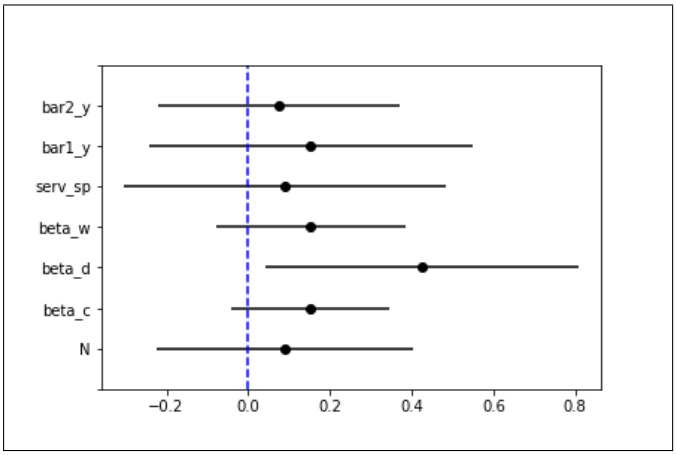

Fig. 5. Sobol sensitivity analysis of the maximum agents per grid cell: Total-order sensitivity index for the influence on the maximum number of agents per grid cell per input parameter.

this influence. Taken into account that the bar1-y and bar2-y parameters should have similar sensitivity indices because of the similarity of the parameters, the first and thus the total order sensitivity index of the bar-y parameters can be assumed positive since the first-order sensitivity index of bar1-y is positive within the whole confidence interval, as can be seen in figure 4

In Figure 6 the average waiting time over a 30-minute simulation is shown with different combinations of bar placements. A clear dark belt can be observed within the figure, showing a low average waiting time when one pair of bars is placed around the middle of the y-axis and the other pair is placed between the middle y-coordinates and the far end of the venue. If both sets of bars are placed close to the stage the waiting time is the highest (as seen in the lower right corner).

Figure 7 shows the maximum number of agents per grid-cell as a function of bar-placement. When both bars are

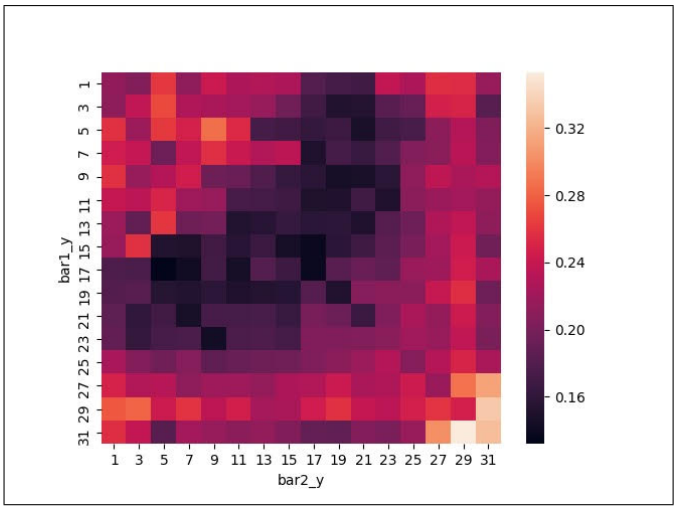

Fig. 6. Average waiting time for all possible bar-placements.

located far from the stage, the maximum number of agents decreases to 4 people per grid cell. Overcrowding happens when the all bars are close to the stage (seen in the lower right corner).

Figure 8 shows the average waiting time when one of the bar-pairs was excluded from the simulation. The y-location of the remaining bar-pair, with double serving speed, is shown on the y-axis. The graph shows the shortest waiting time when the bars are at a y-coordinate of 13. This increases when the bars are moved further away from the stage (to lower y-coordinates), or when the bars are moved closed towards the stage.

In Figure 9 the maximum number of agents in one grid cell is shown for only one pair of bars. The maximum number of agents is at it's lowest (a value of 4 agents per cell). This starts to increase when the y-coordinate equals 19. When the y-coordinate is at it's maximum, the maximum number of agents per cell is 6.

The exact values for the waiting times differ between the results from one and two pairs of bars (so Figure 8 differs from Figure 6) because the calculation for Figure 6 includes agents with a waiting time of 0. These agents are excluded from the calculations in Figure 8.

VII. DISCUSSION

Simulations of an agent-based model of concert-like venue were run to examine the effect of placements of multiple bars on the average waiting time and crowd forming. A Sobol sensitivity analysis was conducted to examine which input parameters had profound effects on the output parameters of the model (inclusive interaction effects). The sensitivity analysis showed that the parameters with the greatest effect on average waiting time were the serving speed of the bars and the placement of the bar pairs on the y-axis. These results are in line with the expectations.

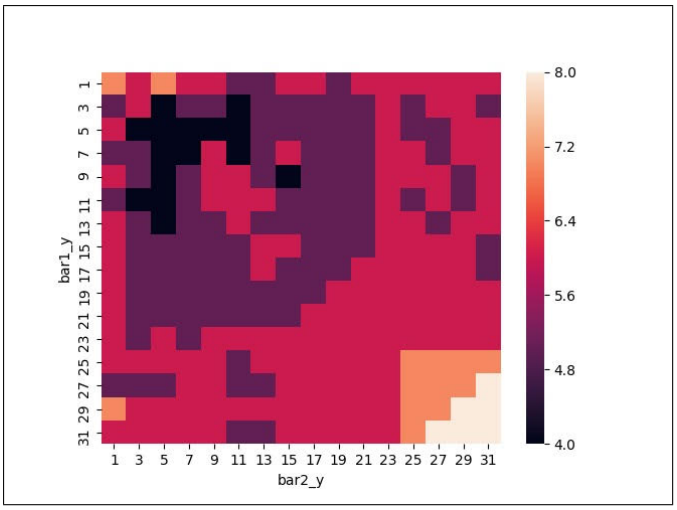

Fig. 7. Maximum number of agents per grid cell for all possible bar-placements

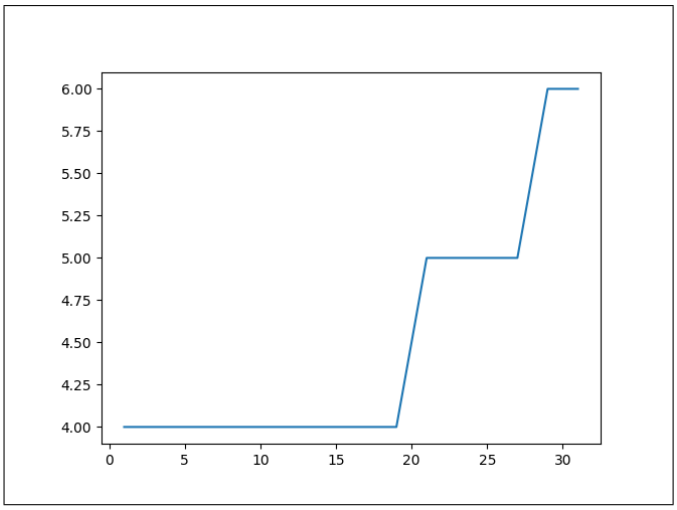

Fig. 9. Maximum number of agents in one grid cell as a function of the y-coordinate of a pair of bars.

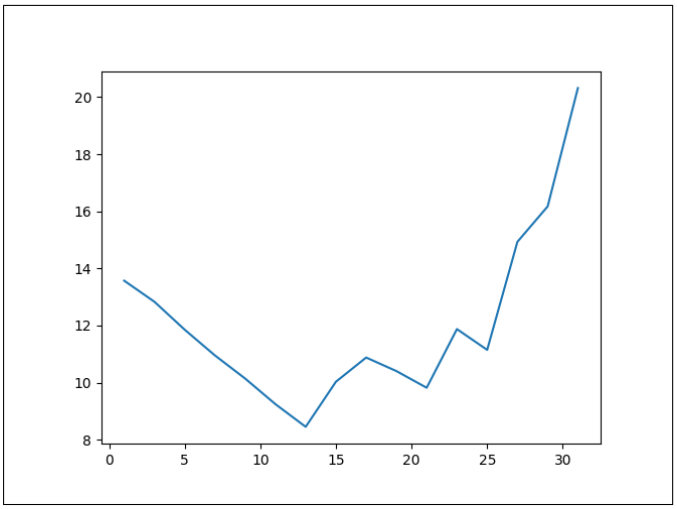

Fig. 8. Average waiting time as a function of the y-coordinate of a pair of bars.

Logically the serving speed has a great influence on the average waiting time, since agents are served less quickly if the serving speed decreases. This means that agents are on average, in bar mode for more time steps and the average waiting time increases. The placings of the bars can be of influence on the average waiting time because it influences the distance agents have to walk to the bar, which time is also incorporated in waiting time. Furthermore the placings of the bar can have an effect on crowd forming around the bar, as explained in more detail later, which can increase the waiting time.

The Sobol analysis showed that changes in the β parameters had the most effect on the maximum number of agents per grid cell within a simulation, especially the distance parameter. Serving speed and bar placements seemed to have some effects too. The strong effect of the β_d parameter with interaction effects on the maximum amount of agents per grid cell can be explained by the effect on the bar-choice of agents in bar mode. If agents strongly prefer to walk to a bar nearby, many agents will choose the same bar when switching to bar mode close to the stage. Especially when the crowd avoidance and waiting time dislike parameters are low and bars are close to the stage, crowd forming can easily occur. Every parameter can thus have influence on crowd forming, but especially the dislike of large walking distances has a great influence when the interactions with other parameters are included.

In the simulation where the effect of the y-coordinates of the bar pairs on the average waiting time was examined, it was shown that the average waiting time tended to be low when one pair of bars was placed around the middle of the venue and one pair was placed between the middle and the far-end site of the venue (away from the stage). These results are not very surprising since it can be unfavorable when the bars are placed close to the stage because of possible overcrowding. In the scenario of two bars far from and two bars close to the stage, agents can decide to go out of the stage area to the closest bars when they are thirsty. If the closest bars are too busy, the bars further away will be visited. This leads to short walking and low waiting times when the bars close by are quiet and somewhat higher walking times but short waiting times when these bars are busy and choices are made in favor of the bars further away.

In the same simulations the effects of bar placements on the maximum number of agents in one grid cell was examined. It was found that the maximum number of agents was very high when both bar pairs were placed very close to the stage and low when the bars were placed very far from the stage. If the bars were placed far from the stage, all the way in the corners of the venue, the maximum number of agents per grid cell increased slightly. This can be explained by the fact that the space around the bars is smaller when

its exactly in the corner, compared to a straight wall. When the bars are close to the stage, both crowds overlap, leading to higher numbers of people per grid cell.

Although the model described is relatively simple, it is sufficient for getting different results on average waiting time and crowd forming. It is however far from perfect for explaining these events in real life. In the model described the predictive capacity of agents is limited and following a gradient of scores is the main motivation to move. Because of this agents are unable to plan their path and would for example get stuck if walls would be present between them and their point of interest. Successful attempts were made to incorporate an Dijkstra algorithm [10] in a model of this sort. This model consists of agents moving over a multi layered grid in different (bar, stage) modes. Because of incoherence in input and output parameters between this model and the Dijkstra model and thus inability to asses the latter on our hypotheses, this model is not incorporated in this paper. However, for future research and reference this model is described in detail in an attached appendix.

Furthermore, the discrete spatial dimensions of the described model prevent realistic interactions between agents. In this model agents move one grid cell per time step when the crowd around them is dense and jump two cells when there is no or little crowd. Because of this agents can jump over each other and it is in general no realistic way to model crowd dynamics. For future research it is proposed to incorporate social force within a model of this sort to create a continuous space and more realistic movements of agents.

Optimization of bar placement within a nightlife venue is normally done based on experience and heuristics. Further elaboration on the agent-based approach as described in this paper, might result in an low-cost and effective way to optimize bar arrangement and thus increase the overall nightlife experience.

REFERENCES

[1] Schelling, T. C. (1971). Dynamic models of segregation. *Journal of Mathematical Sociology*, 1(2), pp. 143186

[2] Mulyana, W. W., & Gunawan, T. S. (2010). Hajj crowd simulation based on intelligent agent. *In Computer and Communication Engineering (ICCCE)*, 2010 International Conference on (pp. 1-4). IEEE.

[3] Luo, L., Zhou, S., Cai, W., Law, M. Y. H., Tian, F., Wang, Y., Xiao, X., Chen, D. (2008). Agent-based human behavior modeling for crowd simulation. *Computer Animation and Virtual Worlds*, 19, pp. 271-281

[4] Epstein, J. M. & Axtell, R. (1996). *Growing artificial societies: social science from the bottom up*. Brookings Institution Press. pp. 224.

[5] David Masad. URL: $https://github.com/projectmesa/mesa$ (accessed: 10.01.2018).

[6] Grimm, V. Berger, U. Bastiansen, F. Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S. K., Huse, G., Huth, A., Jepsen, J. U., Jorgensen, C., Mooij, W. M., Muller, B., Pe'er, G., Piou, C, Railsback, S. F., Robbins, A. M., Robbins, M. M., Rossmanith, E., Ruger, N., Strand, E., Souissi, S., Stillman, R. A., Vabo, R., Visser, U., DeAngelis, D. L. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*. 198, pp. 115-126.

[7] Grimm, V., Berger, U., DeAngelis, D. L., Polhill, J. G., Giske, J., Railsback, S. F. (2010). The ODD protocol: A review and first update. *Ecological Modelling*. 221, pp. 2760-2768.

[8] Willis, A., Gjersoe, N., Havard, C., Kerridge, J., & Kukla, R. (2004). Human movement behaviour in urban spaces: Implications for the design and modelling of effective pedestrian environments. *Environment and Planning B: Planning and Design*, 31(6), pp. 805-828.

[9] Saltelli, A., Annoni, P., Azzini, I, Campolongo, F., Ratto, M., Tarantola, S. (2010). Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communication*, 181, pp. 259-270.

[10] Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs" (PDF). *Numerische Mathematik*. 1: 269271.

UNIVERSITEIT VAN AMSTERDAM

1. Appendix: The implementation of the Dijkstra algorithm, walls and the layering of agents.

Date:
February 7, 2018

1. Appendix: The implementation of the Dijkstra algorithm, walls and the layering of agents.

February 7, 2018

Contents

1. Appendix: The implementation of Astar, walls and the layering of agents.

In this work, we use reactive agents, meaning that in every time step, they will react based on their current perception from the environment ($Ag : Environment \rightarrow Action$). The agents decide on their next actions based on their current state.

1 Environment and Agents

The environment space is defined as a grid, with a fixed width and height, where each cell can be referenced by its corresponding coordinates.

There are two different groups of agents within the environment:

1. Field Agents: In every cell of the grid, there is a field agent, which can be either of type POI (point of interest), type block, or be neither of these types (a regular visitable cell).

2. Commuter Agents: They are the commuters in our model, which can move to one of their neighboring cells in each step. Commuters are able to see around themselves within a defined Moore-rage. They can perceive the density around them, and decide on making their next move to a less crowded area with a defined intensity. Commuter agents occur in different modes, where modes corresponds on different goals of agents (e.g. party mode at the main stage or beer mode at the bar).

2 Agent Decision Tree

Each agent is initialized with a certain structure for decision making (visualized in figure 1). Upon creation each agents starts with a chance distribution of picking a particular POI to set as its goal. The "bar" POI's having a significantly lower chance of initial selection compared with the "stage" POI. Once this goal selection has been made the agents enter a loop in which every step the evaluate which mode they are in (e.g. which POI has been selected). After this is established they select the best course of action based on a mix of the Dijkstra algorithm (section 3) and a density evaluation (section 4) . The chosen action will then result in either the moving of the agent to another gridcell, or to stay within the same cell. An evaluation is then necessary, to determine whether the agent has reached its destination POI. If not, go back to evaluating to what POI the agent was going. However if the agent reached its destination its next action depends on the POI that the agent is at. If at the stage, decide next POI each timestep. If at bar, wait to be served and then move back to the stage POI.

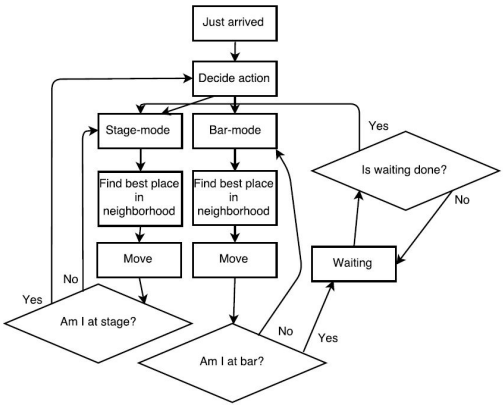

Figure 1: Agent Decision Tree. The diamond shaped nodes being an evaluational node and the square shaped nodes being actions.

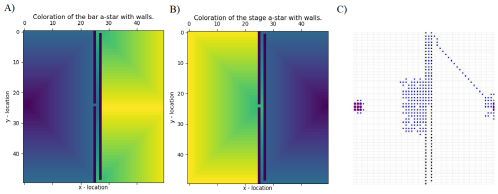

Figure 2: Coloration of Dijkstra with walls (A,B) and figure C is the grid as can be seen from the UI.

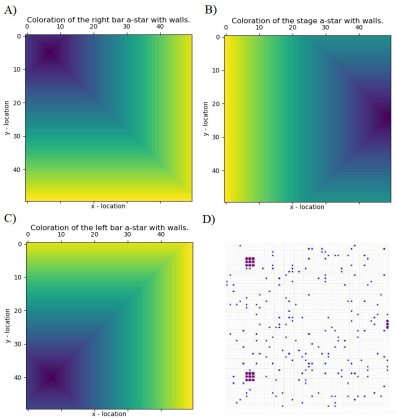

Figure 3: Coloration of Dijkstra without walls (A,B,C) and figure D is the grid as can be seen from the UI.

3 Dijkstra Algorithm

Given an N by N grid, a set of blocks with specified coordinates in the grid as impassable objects (walls), and a point of interest that agents are aiming to reach (bar, stage), Dijkstra algorithm precomputes the shortest path between each coordinate of the grid (x_i, y_i) to that point of interest coordinate (x_{poi}, y_{poi}). The algorithm considers the grid as a graph consisting of nodes and edges, where nodes are grid cells, and the edges are equally weighted and define the possible movement of an agent. This movement is in the eight neighboring cells based on Moore neighborhood. The algorithm does a breath first exploration with the POI coordinate as its root, and assigns a value for each cell as the number of steps required between the explored cell and POI cell. This value is calculated as $f(x, y)$, the minimum number of passable cells between (x_{poi}, y_{poi}) and the current cell (x_i, y_i).

The effect of the algorithm is shown in figure 3 where the darker colors represent lower Dijkstra values. Meaning that agents will be drawn towards their goal through this gradient. In this figure three points of interest are used representing two bars (fig 3A, 3C) and a stage (fig 3B). In figure 3D the map is shown including agents as can be seen from the UI.

In the analysis of this model, two scenario's will be compared. One without walls (fig 3) and one with a funneling structure comprised of walls as can be seed in figure 2). Here only two points of interest are used, and instead of random initialization all agents are spawned from a single point of interest being the bar (fig 2A). This is done to witness the dynamics of crowds in tight spatial situations.

4 Vision and Crowd density

One important hurdle in simulating crowd dynamics is the presence of other agents as dynamic obstacles. In this respect agents are given a vision with Moore range r and are able to observe the crowd as far as within their Moore-range. They perceive other agents density around their neighborhood upon their decision for their next move, The crowd density function is calculated as

$d(x, y) = p * C(x, y)$, where c

$$\frac{\text{count of agents with Moore range r}}{(2 * r + 1)^2 - 1}$$

and the value of the denominator is the total number of agents observable through the agent's vision.

Since $0 < c < 1$, and the maximum difference of route cost f among the next movement candidates, Moore neighbors, is at most 2, the choice of going to the next neighbor will depend on the observed density, the value of p, and the cost f associated with the candidate cell.

For example, if we choose path p as $2 * ((2 * r + 1)^2 - 1)$, the added value of cost in two extreme situations will be:

1. Only one agent observed within Moore Range: $(2 * ((2 * r + 1)^2 - 1)) * 1/((2 * r + 1)^2 - 1) \approx 2$
2. The agent is completely surrounded with other agents within its vision $(2*((2*r+1)^2-1)-1)*((2*r+1)^2-1)/((2*r+1)^2-1) \approx (2*((2*r+1)^2-1))$

and considering the difference of route cost f among the next movement candidates, which is at most value of 2, the agent would move to a cell that has the minimum value of $f(x_c, y_c) + d(x_c, y_c)$ among the neighbors.

Therefore, the increase on the value of p increases the intensity of agents to avoid the crowd observable with their vision.

5 Layering

Our initial aim was to allow a cell in the environment to be occupied by only one agent at each time step. This constraint was altered by placing agents that have the same intention (to move toward or be at a certain POI) in the same layer. So agents can only occupy cells that have none of the agents within the same layer, but they can move to cells in where other agents of other layers reside. This simplification, lets agents to move from a POI that includes a crowd to another POI, and avoid getting trapped within the crowd. As agents would be hindered by agents that follow the same intention.

6 Crowd Utility

Once an agent fulfills the goal of getting served at a serviceable POI, being bars in our case study, the value of total utility of the agent is increased by:

$$\frac{\textit{shortest time + serving time at the POI}}{\textit{total waiting time}}$$

where total waiting is defined as the time of deciding to go to a serviceable POI until the time the agent is served and changes its goal to move to another POI, which is the stage in our case study.

In case of no crowd the total utility will be equal to the number of visits to the serviceable POIs, and we can normalize this value as

$$\textit{Utility} = \frac{\textit{total utility}}{\textit{number of serviceable POI visit}}$$

which is always between 0 and 1, where 1 means that agent did not face any crowd in its commuting (i.e. no delays have taken place).

7 Experiments

In this case study several experiments were conducted to test the sensitivity of some parameters, namely the number of agents present in the system, Vision and Density. The effect of density was tested for both a small range and large range of values. And finally an experiment was conducted to test the extent of stochasticity. Each experiment was conducted twice, once without any walls and once with walls in a funneling formation as is shown in figures 3 and 2. Also each data point shown as of now is a complete run of 1800 time steps. Standard parameters are N-agents = 200, density coefficient and vision = 0. Due to time constraints these single parameter sensitivity analysis and their variations could not be done for each parameter.

7.1 Experiment 1 -Variation on the number of Agents

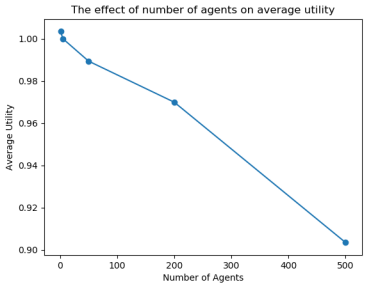

Figure 4: Figure showing a sensitivity analysis of the number of agents vs. utility. Walls were used in this graph.

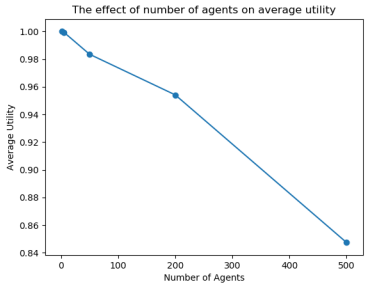

Figure 5: Figure of the model without walls showing a sensitivity analysis of the number of agents vs. utility

This experiment was conducted to test the effect that agents have on each other with respect to the efficiency of navigation (measured by utility as described in section 6).

The expectation is that as the population grows, each agent would experience more hinder by the others, both in the sense of the physical occupation of a grid cell as well as the predisposition of wanting to avoid area's that are more dense. Besides this, the expectation also arises that the effect of funneling by the walls will increase as more agents try to move through this narrow entrance.

As can be seen in figures 4 and 5 the utility decreases (10% with walls) and with (16% without walls) as the population increases.

This decrease in utility falls within the range of expectation as a higher dense population creates a greater delay with respect to navigating from A to B. However, the difference between the usage of walls or not does not lie within our expectation, as the utility is actually higher with the usage of walls. This phenomenon could arise from the way that density is calculated. Density is calculated by counting the population within a certain area. But walls aren't being taken into account, and other agents tend to avoid the outer sides of the walls, as the Dijkstra values are lower there (fig 2, and 3). In this sense agents are guided towards a POI through a less dense area, even though they might be surrounded by agents front and back.

7.2 Experiment 2 -Increase on the value of Vision

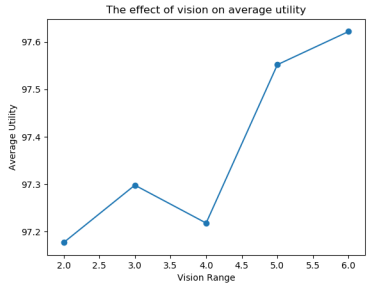

Figure 6: Stepwise increasing of vision with respect to the utility. Walls were used in this model.

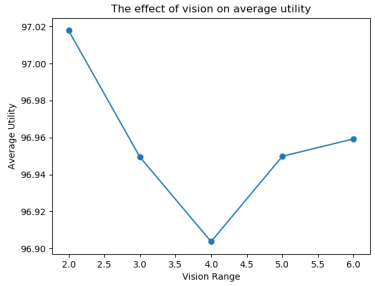

Figure 7: Sensitivity analysis of vision with respect to the utility.

This experiment was aimed to shed light on the matter of the effect of variations within the vision parameter. The vision was increased from 1 to 5, and each time the mean utility was calculated. The expectation was that if the vision of an agent increases the utility would go down because agents could calculate density easier and have a bigger radius to avoid other agents, thus resulting in a bigger "detour" when walking towards a POI.

As is plotted in figures 6 and 7 the differences in utility are 0.4% (with walls) and just 0.12% (without walls).

This difference in utility can be described as being of a insignificant manner. This can be an indication that the ranges chosen are not sufficiently large, or the effect of vision with the chosen density coefficient of 2, was simply ineffective. The parameter vision can as such be described (within these ranges) as insensitive.

7.3 Experiment 3 -Increase on the value of density coefficient

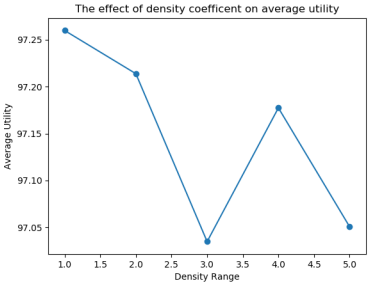

Figure 8: Sensitivity analysis of the density coefficient with respect to the utility.

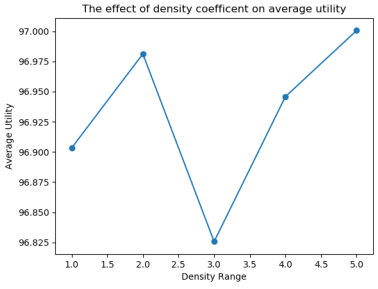

Figure 9: Sensitivity analysis of the density coefficient with respect to the utility.

This experiment was conducted to test the influence of the density coefficient on utility similarly to the previous experiment. The expectation here follows the same logic as with the vision parameter. A higher density coefficient would

result in a greater avoidance of other agents, in turn, resulting in an inefficient navigational mobility.

The effect of changing the density coefficient results in a decreased utility of 0.2% (with walls) and 0.175% (without walls).

These small differences seem incoherent and more based on stochasticity than an actual sensitivity to the parameter. This brings us to the following two experiments.

7.4 Experiment 4 -Increase on the value of density coefficient for a large range

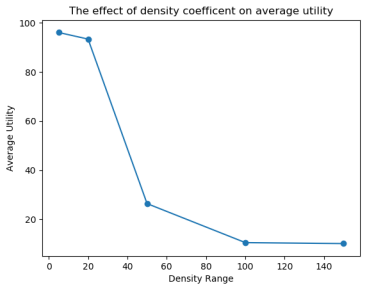

Figure 10: Large range value density sensitivity analysis done with respect to the utility with usage of walls.

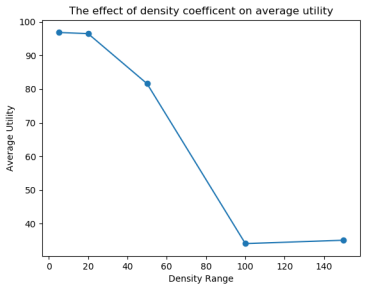

Figure 11: Large range value density sensitivity analysis done with respect to the utility. No walls were used here.

Since both vision and density on their own seem to have little effect on the utility, we took things to a more extreme situation to measure whether the density coefficient can have an effect at all. This is why a density coefficient of 5, 20, 50, 100 and 150 is used. In figure 10 and 11 we can see that the change of density starts to get noticeable after around the value of 50. So a strong repelling effect occurs here with respect to the other agents. The differences are all in all a drop in 80% in utility. Also the utility seems to drop faster in the model with the walls. This concurs with our previous expectations, but the values had to be larger to be noticeable.

7.5 Experiment 5 -Stochasticity over multiple runs

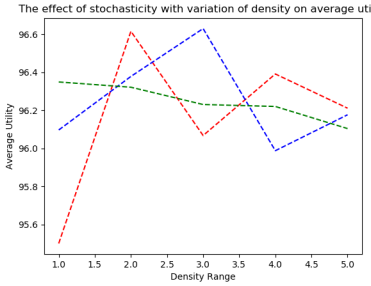

Figure 12: Showing the effect of stochasticity in a model with walls. Three complete runs as is done with the density coeffient. Same parameters were used in all three runs.

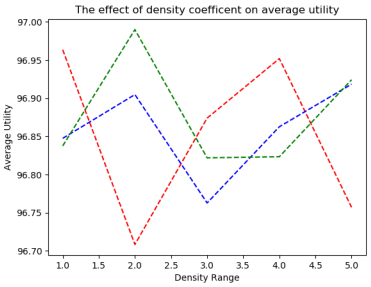

Figure 13: Showing the effect of stochasticity in a model without walls. Three complete runs as is done with the density coeffient. Same parameters were used in all three runs.

In this experiment we tested the effect of stochasticity over a set of three runs. All parameters were kept equal over the runs. The expectation is that because of the stochastic nature of the model itself each run would differ from each other. This could be an explanation for the non-significant differences between the parameter changes.

As we can see in figures 12 and 13 all runs in fact do differ in both models with and without walls. And they differ amongst themselves. This stochasticity might explain some of the dynamics seen before.