The Significance of Perfect Information within Social Simulations based on Cellular Automata

Joppe Bast

Abstract

This research covers the question if the assumption of perfect information within a cellular automaton based social simulation has a significant impact on the behaviour within the simulation. Perfect information means that all agents within the social simulation have access to all information in their neighbourhood. To test if perfect information has a significant impact, I first recreated the social network simulation of Hegselmann and Flache (1998). Their simulation assumes perfect information when agents move through the cellular automaton. Secondly, I removed perfect information from the simulation by implementing a basic function of uncertainty such that agents have less information about faraway cells. Afterwards I compared the two simulations to find that when the certainty drops fast enough there were significant differences in two cases.

Bachelor Thesis Artificial Intelligence
Author: Joppe Bast
Date: 19 November 2020
Supervisor: Gerard Vreeswijk
Second reader: Yupei Du
Faculty of Humanities
Utrecht University
7,5 ECTS

Contents

1 Introduction

The last few years, the influence of information on social behaviour has been made clear. Take for example the influence of fake news and bubble forming on social media. However, some social simulations using cellular automata still gloss over the importance of information. (For example, the following sources (Dabbaghian et al., 2010), (Dabbaghian et al., 2012), (Hegselmann Flache, 1998)) A lot of cellular automata based social simulations make the assumption that their agents have access to all information from their surroundings. This assumption is called perfect information.

It might not be surprising that a lot of cellular automata use perfect information, as most cellular automata work with local interactions. When programmers implement local systems they don't have to consider which information is accessible, as local information should be close by and accessible. But sometimes this 'local' neighbourhood gets very large. Because those programmers are used to implementing local interactions they might not consider that the agents should probably not have access to all information in a large neighbourhood. When implementing global interactions, it might be smarter to use a system for information sharing (Dudek–Dyduch & Wąs, 2016).

When a model includes perfect knowledge, papers rarely mention it as a deliberate choice. The absence of a system that reflects global information makes the simulation less realistic. This isn't necessarily a bad thing, all simulations have to make abstractions of reality. Be it because the abstraction improves performance or because the more realistic system had negligible effects. But those trade-offs should be assessed as small additions can quickly lead to surprising results, especially in emergent systems like cellular automata.

This brings us to the key question of this paper:

> Has the assumption that agents have perfect information about their surroundings a significant impact on the resulting social simulations within cellular automata?

To answer this question, I've first recreated an existing model from Hegselmann and Flache (1998). After that, I removed the perfect knowledge by implementing a simple uncertainty step and compared the resulting simulation with the simulation without uncertainty. After statistical analysis, I found that there was a significant difference in two cases.

2 The concept

This research makes use of an existing cellular automaton model from Hegselmann and Flache (1998). Their model simulates how networks of mutual support between agents can form and develop. In this cellular automaton cells spread over a 21 by 21 world can either be empty or occupied by an agent. All

agents play a two-person support game with each of its 4 Von Neumann neighbours. Based on if a neighbour cooperates or not, they get a certain reward. The cumulative reward of all 4 neighbours determines whether the agent is satisfied with his current location. Every agent has a chance to be able to migrate to a new location within a 11 by 11 neighbourhood. If an agent is satisfied with the reward of its current location it will only use this chance when it sees a better rewarding location to move towards. A dissatisfied agent will move to a worse location for the possibility to get access to a better location next time. The agents are assumed to be rational. This combined with that all agents have perfect information means that they know which locations will have a better reward or not.

Cellular automata originally work with only local neighbourhoods. It's thus difficult to make more realistic global interaction without removing perfect information by for example adding a knowledge sharing system (Dudek–Dyduch & Wąs, 2006). The migration system from Hegselmann and Flache (1998) is trying to simulate a global system as the migration window covers almost a quarter of the total world space. But on a technical level, it still works as if it was a local system as the agents have perfect information. Removing this perfect information should thus make the simulation more realistic.

Removing perfect information can be done in a lot of different ways. For example, by adding memory for each agent and a learning system so they learn the classes of other agents. However, systems like these add many variables that can significantly influence the model in more ways than just removing perfect information. To keep at the root of the question, it was important to remove perfect information with the least amount of changes. This lead me to implement a basic system that simulates uncertainty. Uncertainty comes into play when an agent wants to migrate to a new cell. Agents can no longer predict the best location with perfect certainty. This means the agents no longer have perfect information about their surroundings. The further a cell is from an agent, the less certain the agent is about the reward from playing with that cell, the more that agent has to guess what that cell's reward will be. This makes the simulation more realistic, as when something is further away, it is harder to have information about it.

3 The old model

3.1 Introducing the model

This section will cover the details of my interpretation from Hegselmann and Flache model (1998). The section after this one will cover which parts of the Hegselmann and Flache's explanation of their model were not detailed enough or contradicting themselves, and why I chose for my interpretations.

As said before, the simulation takes place on a 21 by 21 grid world. each cell of this world can be empty or occupied by a single agent. Each agent in this world

plays a two-player support game with each of their Von Neumann neighbours. The support game works as follows: Each agent has an ascribed static risk class. This risk class entails the chance (p) by which the agent will become in need of help. The risk-classes are 1 through 9 which correspond with the chances 0.1 through 0.9. If an agent requires help, it can't help the other agent. However, if an agent does not require help, it can choose to help their neighbour or not. Both these factors can lead to different rewards. The different rewards are: Being drowned (D), moving on without helping (M), helping (H), and being saved (S). To get the right behaviour the values have to be chosen such that S $>$ D and M $>$ H.These conditions cause agent to want to be saved and not want to help. Which decisions lead to these rewards can be found in Figure 1.

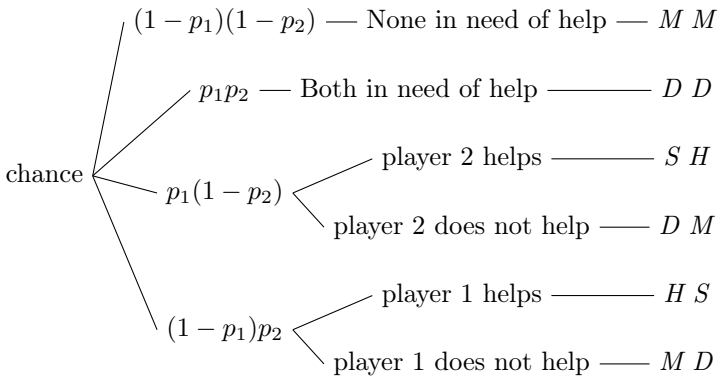

Figure 1: Two-player support game with:
D,M,S,H = reward for Drowned, Move On, Saved or Help
p_1 = risk class player 1
p_2 = risk class player 2
S $>$ D and M $>$ H
$0 < p_1 < 1$
$0 < p_2 < 1$
(Hegselmann & Flache, 1998)

This means that generally the payoff of one game for an agent i playing with agent j is defined as:

$$\text{Payoff}_i = p_i p_j D + p_i(1-p_j)S + (1-p_i)p_j H + (1-p_i)(1-p_j)M$$

If the agent tries to play a game with an empty cell, he can't help anyone and can't be helped himself thus the payoff then would be:

$$\text{Payoff}_i = p_i D + (1-p_i)M$$

Because the reward for not helping, M, is greater than the reward for helping, H, the rational choice in a single game will always be to not help. However, because the same two agents can play multiple games together over multiple iterations, we have to approach it as a repeated game. That means that agents can use supergame strategies that can result in cooperation. The model uses two conditions for two agents to cooperate. The conditions have to be true both ways, such that there is mutual support.

The first condition is that the cooperation will on average result in a higher reward than no cooperation:

$$p_i(1-p_j)(S-D) > p_j(1-p_i)(M-H)$$

The second condition is that the chance for further iterations of the game with the same players has to be sufficiently high:

$$a_i \geq \frac{1}{1 - p_j(1 - p_i + p_i(1-p_j)\dfrac{S-D}{M-H}}$$

a_i Stands for the probability of stability for agent i. The probability of stability is the same for all agents because they are pessimistic and assume that all agents will move away when given the chance. Thus $a = (1-q)^2$ with q being the chance that an agent gets the option to migrate. Because of this condition a risk-class may want to cooperate with fewer risk-classes when the migration chance increases.

Agents can be satisfied or dissatisfied with their current location. An agent is satisfied if its surplus, the reward of the current position minus the worst possible position (all neighbouring cells are empty), is at least half of the difference between the best and worst reward. The agents are rational and realistic and can thus figure out what their best payoff is. Each agent knows which classes will cooperate with them and which won't, based on the aforementioned conditions. The best possible reward is the reward from 4 neighbours of the best class with which mutual support is possible.

When an agent gets the opportunity to migrate, it will search for the empty cell with the highest expected reward within an 11 by 11 neighbourhood. If that

results in multiple candidate locations with the same value the agent chooses randomly between those. The agents have perfect information about their surroundings and are assumed to be perfectly rational, so this expected reward matches the reward the agents will get. Satisfied agents will only move if the new expected reward is better than their current location. A dissatisfied agent will always move. Sometimes agents will never be supported because no risk-classes will support them. Agents for which no support relations are possible should technically be satisfied because the best and worst-case scenario are the same. Thus they will always have a surplus equal to 50% of the difference which is 0. The model makes an exception to the rule so agents that have no possible support relations are always dissatisfied.

Hegselmann and Flache's model showed different behaviours for different migration probabilities. Based on the migration probabilities agents would be able to support different amounts of risk-classes. With an migration probability of 0.05 every risk-class works with other agents of the same class and at least one other class. With the middle classes working together with more different classes than the worst and best classes. A migration probability of 0.10 causes every class to support less other classes and thus the best and worst classes will only support others of the same class. An even higher migration probability of 0.15 causes classes to establish support relations even less likely. The worst and best classes will no longer work with others of the same class and are thus doomed to never find any support relation. The second best and second worst classes now only work with others of the same class.

The initial conditions:

Cell structure:
Interaction window: Von Neumann neighbourhood.
Migration window: 11 by 11 with the original position in the centre.
World: 21 by 21 torus.

Cell amounts:
35 agents of each of the 9 risk classes (0.1 to 0.9) represented by different colours.
126 empty cells.

Payoff values:
Saved = 5
Drowned = 1
Move On = 7
Help = 6

Minimum level to be satisfied: 50%

Probabilities for getting migration options:
q = 0.05 thus a = 0.903 (first experiment)
q = 0.10 thus a = 0.810 (second experiment)
q = 0.15 thus a = 0.723 (third experiment)

(Hegselmann & Flache, 1998)

3.2 Interpretation

The model described above is my interpretation of Hegselmann and Flache's model. There were some choices to be made, as there were inconsistencies and ambiguity in their description. While their mistakes are bad to make, it wasn't to difficult to find what they probably meant by testing different implementations and reading between the lines. This section will cover which choices I had to make and elaborate on why made them.

First off, a minor mistake: The authors claim there are $315(9 \times 35)$ agents and 136 empty cells on a 21 by 21 world. This is impossible as $21 \times 21 = 441$ and $315 + 136 = 451$. The amount empty cells instead has to be $441 - 315 = 126$ Secondly, the original article was not clear on how satisfied agents chose their new location. They stated that dissatisfied agents choose the location within the migration window with the highest reward. And they state that satisfied only move when the new location has a higher reward than their current location. While they never exactly stated it, they probably meant that satisfied agents also move to the best possible location. Provided that the new location is better than the current one.

The only proper decision I had to make was how to decide when an agent is satisfied. Hegselmann and Flache first state: "An individual will be satisfied with any social position which offers a certain fraction of the difference between the payoff in the best and the worst social positions"(Hegselmann & Flache, 1998, section 4.9). But further down they write: "they are satisfied: trivially, they get at least 50% of what they could get if all their neighbours were members of the best risk class willing to engage in mutual support with them"(Hegselmann & Flache, 1998, section 4.23). Here they don't mention the difference between best and worst but just a fraction, 50%, of the best. From these statements I had three options:

1. An agent is satisfied when its current reward is at least 50% of its best possible reward.
2. An agent is satisfied when its current reward is at least 50% of the difference between its best and worst possible reward.
3. An agent is satisfied when the difference between its current reward and the worst possible reward is at least 50% of the difference between its best and worst possible reward.

I went with option three as this interpretation ensures that the model has the same symmetry of class behaviour that the model of Hegselmann and Flache shows. So The middle classes work together with the largest number of other classes, and both the better and worse classes show a similar decrease in the amount of classes to support. It also causes the changes in migration chance to have the same effect on the best and worst classes. Most of all it ensures that it isn't too easy for agents to be satisfied. Because both option 1 and 2 make it so that agents are rarely dissatisfied, which causes the simulation to quickly get

into stable configurations. As Hegselmann and Flache's model stays unstable, I had to go for option 3.

3.3 Similarity

The data from my model was extracted by running the model 100 times for 2500 iterations for each migration probability used in the article of Hegselmann and Flache ($q = 0.05$, $q = 0.10$, $q = 0.15$). It is important to test all three values because the simulation shows different behaviour for each of them. We have to test if the change in migration probabilities for the recreation leads to the same behavioural changes. There were two reasons to let the simulation run for 2500 iterations. First, it made it easy to compare my graphs with those of Hegselmann and Flache, as theirs are also cover 2500 iterations. Second, the data I looked at was the average network dividend. This value is the average surplus, the current reward minus the worst possible reward, per class. After about 2000 iterations all risk classes will reach a semi-stable state. While the average network dividend of most classes won't ever be stable the value of these classes will no longer have a general upwards trend and will fluctuate around a single value. This average value over multiple runs will stabilise after those 2000 iterations .

To test how close my implementation was to theirs, we would have to run statistical tests. But the only data available from their implementation are 4 world states 2 bar-plots, and 3 graphs from individual runs. The graphs were not very high resolution and in the start there was a lot of overlap between classes. This made it not feasible to extract data reliably from the graphs. Without data, it was impossible to run a real statistical test. To at least make a visual comparison, I overlaid the averages with standard deviations of my runs over their graph in Figure 2. The Graph shows the average of the network dividend for each class over 2500 iterations. The average network dividend is the average difference between the current score and the worst score for all agents from one class. From Figure 2, we can see that their graph mostly falls in the standard deviation. So their graph is a possible result from my implementation. This doesn't prove that the models are the same because their data is just from a single run, however it serves as a good indicator.

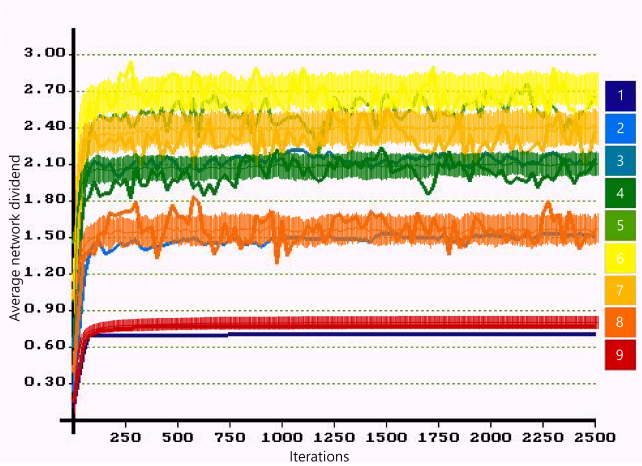

Fig. 2a: Comparison classes 9,8,7,6 and 4

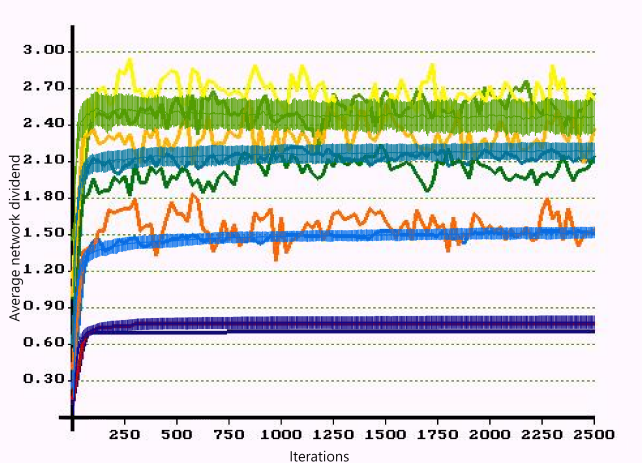

Fig. 2b: Comparison classes 5,3,2 and 1
The mean of the average surplus with standard deviation.
My implementation laid over the graph (q = 0.10) from
Hegselmann and Flache (1998)

It might be even more important to compare the behaviour of the two models. The visual comparison of general behaviour based on the migration probability can be found in Figure 3. This shows that the same variable changes result in the same behaviour patterns. Specifically, with a higher chance on migration($q = 0.10$) the highest and lowest classes will support only others of the same class. And with an even higher chance($q = 0.15$), there will be no cooperation possible at all for these classes, and the second worst and best classes will only cooperate with themselves.

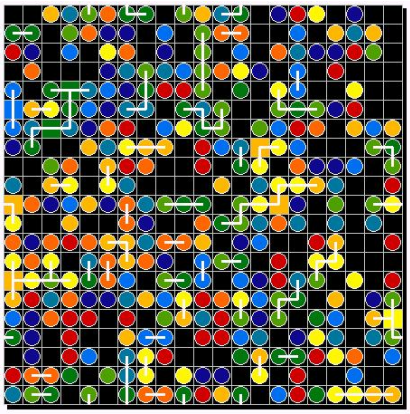

Fig. 3a: Primordial soup
(Hegselmann & Flache, 1998)

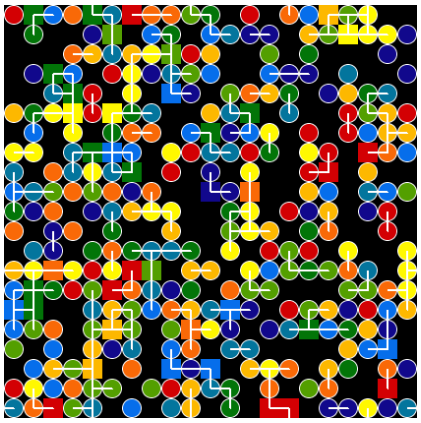

Fig. 3b: Primordial soup
(my implementation)

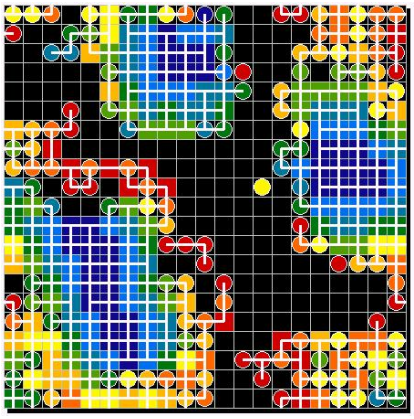

Fig. 3c: chance for migration 5%
(Hegselmann & Flache, 1998)

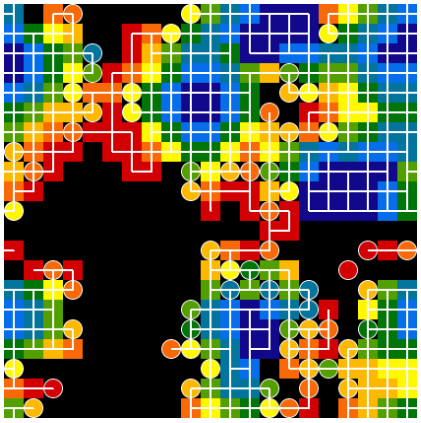

Fig. 3d: chance for migration 5%
(my implementation)

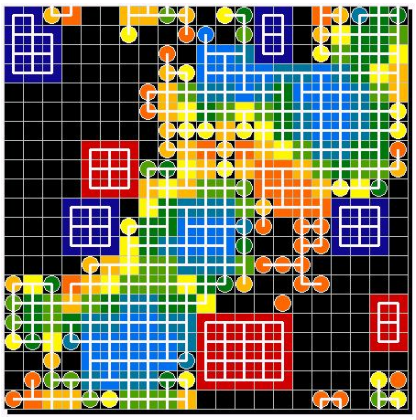

Fig. 3e: chance for migration 10%
(Hegselmann & Flache, 1998)

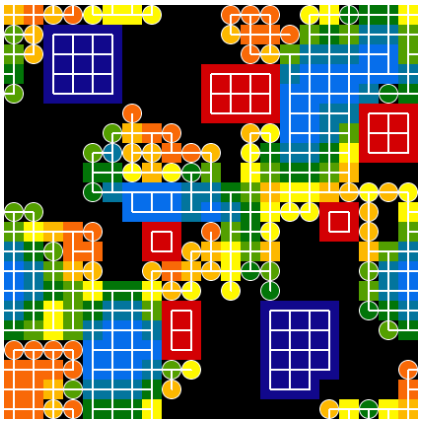

Fig. 3f: chance for migration 10%
(my implementation)

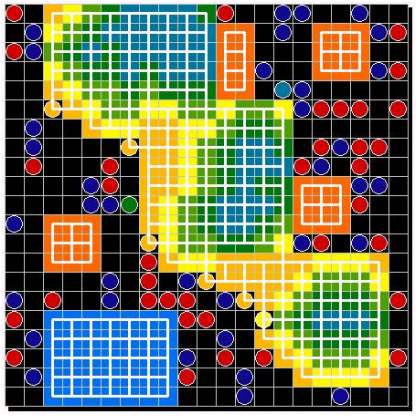

Fig. 3g: chance for migration 15%
(Hegselmann & Flache, 1998)

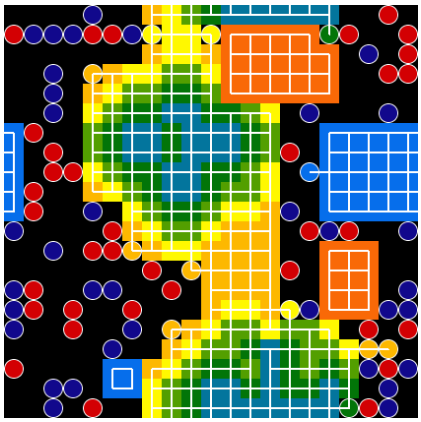

Fig. 3h: chance for migration 15%
(my implementation)

Multiple world states comparing Hegselmann and Flache (1998) model (left) with mine (right). Figures 3a and 3b are random starting states. 3c to 3h are world states after 2000 iterations. The colours correspond to the different risk-classes as can been seen in the legend of Figure 2. A white line between two cells mean that they have an active support relation.
When a cell is filled it means the agent is satisfied, when it's round the agent is dissatisfied.

4 The new model

4.1 Uncertainty

In this section, I'll explain in more detail how I implemented the uncertainty. When agents are uncertain they have to guess. A pure guess would be the average reward of playing with an arbitrary cell. Be it empty or not. That said, an agent will seldomly have to guess completely. Every agent has a percentage for every cell within a 13 by 13 neighbourhood to represent how certain it is about their knowledge of that cell. This certainty is based on the distance according to a Gaussian distribution. This Gaussian distribution relies on the variable σ which stands for the standard distribution. σ determines how fast the certainty drops as the distance increases. How certain an agent is about another cell can then be calculated with only σ and the distance between the cell and the agent.

Because the simulation takes place on a grid with discrete cells, there are a limited amount of distances possible. These possible distances are known to us. They are the distances to all cells that could be neighbours of possible migration options. This means we can calculate a certainty window based on sigma in advance. This would be a 13 x 13 matrix with the position of the agent at the centre. It would be 13 x 13 because the migration window is 11 x 11 and we need to be able to calculate the neighbours of those edge positions. The certainty of the original position and its direct neighbours will always be set to 1. As the agent knows his old position would be empty if he moved, and he just played with his old neighbours. An example of the certainty window can be found in Appendix B.

If we think of the certainty window as a matrix C with the values $c_{dx,dy}$ the expected reward from playing with a cell that lies dx cells away on the x-direction and dy cells away on the y-direction would then be:

$$E(reward) = c_{dx,dy}(p_i p_j D + p_i(1-p_j)S + (1-p_i)p_j H + (1-p_i)M) + (1-c_{dx,dy})A_i$$

Where A_i stands for the average reward for playing with a cell for agent i. A two-dimensional Gaussian distribution is defined as:

$$f(x,y) = e^{-1\cdot(\frac{(x-x_0)^2}{2\sigma_X^2} + \frac{(y-y_0)^2}{2\sigma_Y^2})}$$

Because we want the uncertainty to drop the same rate in both the x and y directions we take $\sigma = \sigma_X = \sigma_Y$. And because we know the possible distances, we can use those instead of calculating them in the function. Thus, C would be calculated as follows:

$$c_{dx,dy} = e^{-1\cdot\frac{dx^2+dy^2}{2\sigma^2}} \text{ if } c_{dx,dy} \neq c_{0,0}, c_{1,0}, c_{0,1}, c_{-1,0}, c_{0,-1}$$

$$c_{0,0}, c_{1,0}, c_{0,1}, c_{-1,0}, c_{0,-1} = 1 \text{ (the original spot and neighbours)}$$

$$c_{dx,dy} \in C \text{ for } -6 \leq dx, dy \leq 6$$

4.2 Methods

To test for significance, the new model will be run with 3 different values of σ (1,10,100) for each of the three migration chances q (0.05,0.10,0.15) covered before. The 9 combinations of parameters will each be run 100 times with random starting world-states. Every run will be 2500 iterations long. These parameters are mostly the same as seen previously for the data of the old model. The only addition is the three values for sigma which in combination with the 3 values for the migration probability (q) leaves us with 9 combinations. I have chosen 1, 10 and 100 as the values for σ for three reasons. The first reason is that $\sigma = 100$ has a large enough σ to cause the certainty to drop very slowly. So slowly that the data from this should not be significantly different as the agent should be perfectly certain about all their information. If we find a significant difference here that would mean there is something wrong. $\sigma = 100$ is thus meant to be a control-group. Secondly, $\sigma = 10$ is a realistic value where there is a slow decent of certainty but agents are still 70% certain about even the farthest possible new neighbours. Lastly, $\sigma = 1$ causes the certainty to drop quickly. Probably unrealistically fast as the agent is almost 0% certain when looking far away but it will be interesting to see if this leads to large differences. Running the new model 9 times with different parameters will give 9 data-sets which each consist of 900 values for each of the 2500 timestamps. 900 values because there are 100 runs with values for all 9 risk-classes. The same will be done for the old model with the three migration probabilities (0.05,0.10,0.15). To make it easier to compare the data-sets I will take the average and standard deviation from the 100 runs. This will give us a total of 12 data-sets with 9 values and 9 standard deviations for each of the 2500 timestamps.

The goal then is to compare the 9 data-sets of the new model with the corresponding data-set from the old model with the same migration probability. This can be done in multiple ways. I first wanted to approached it as a time-series. However, to test significance we usually perform a form of t-tests. Normal t-tests take the mean of each data-set to compare them. This does not work well in this case as it would average the data regardless of the timestamp, that means that the values from timestamp 1 would be a substantial outlier. To combat this I thought about using a paired t-test with using values of the same timestamp as pairs. But a paired t-test may not be used to compare a time-series as each value depends on the value of the earlier timestamp and the pairs should be independent of each other. Thus I searched for other means to compare time series. But none were applicable as they focus on comparing the trend or seasonality of the lines (Hamilton, 2020). Seasonality isn't applicable as there is no repeated time fluctuation. The trend didn't seem to be the fundamental difference. Most classes show a similar trend, quickly rising and then vary around a steady value.

In the end, I settled on not using the entire time series. A lot of data from the time-series, like the quickly rising start, was not significant. The interesting part is the value the classes vary around. The value which the classes, while never

stable, converge towards. This value will give an idea of how all the classes will behave after the quick increase in network dividend in the start. While most classes never reach a stable state, their average value will converge fast. We can see this clearly in the graph with the average surplus (Figure 2, Figure 6 and Figure 7). To find this value, I will take the average and standard deviation of the last 500 cycles for each class. Only the last 500 iterations will be used because by then all classes have stopped steadily increasing on average. While most classes stop growing earlier, any class that only works with others of the same class may need that time to settle. This is because those classes can find a stable configuration by clustering in perfect rectangles. It takes some more time for those classes to cluster in a way that those rectangles can be formed. The average network dividend a class converges towards will be called the asymptotic value from here on.

To compare the asymptotic value from the new model with the old model a paired t-test will be used. The t-test pairs will be defined such that the test compares the values of the same classes of the different models with each other. Again a normal t-test wouldn't make sense here as it would assume the values from all classes came from the same mean. When the paired t-test gives a p-value smaller than 0.05 it will be accepted as significant. If a significant difference is found we'll be making a visual comparison with the graphs of the average network dividend with standard deviation.

4.3 Results

This section will cover the resulting values of the methods. Compared to the old model, the asymptotic values of the new model with a σ of 100 doesn't show any significant p-values for all three migration probabilities (0.05, 0.10, 0.15). All p-values were above 0.9. This is as expected and means that the algorithm probably was correctly implemented. However, this is no conclusive proof that they are actually the same, because a t-test has the null-hypothesis that the models are the same. However, as we can see from Figure 4 the values are so close that it is likely.

With a smaller $\sigma(10)$, the first significant difference comes forth. Only in the experiment with $q = 0.05$ a significant difference (p-value is0.02371) between the asymptotic values was found. This is probably because this scenario has the most classes that are influenced by uncertainty. The experiments with $q = 0.10$ and $q = 0.15$ both have classes of which agents only work with others from the same class. These classes quickly form close to stable structures. That means that there is no incentive to move far away, thus the uncertainty has less effect. The uncertainty has, of course, no impact on the classes that never work with anyone. As they will always have a network dividend of 0.

An even smaller $\sigma(1)$ gives the same result as before. Again, a statistically significant difference (p-value is 0.009152) is only visible in the experiment with $q = 0.05$. The other experiments also had a bigger change but still not significant

($p = 0.1419$ for $q = 0.10$ and $p = 0.2844$ for $q = 0.15$) It is interesting to see on which classes the changes have a bigger impact. As we can see from Figure 5 there is a bigger impact on the agents with a higher risk class. This is the case because most classes move around more. That gives the higher risk classes the chance to wiggle in between other cells rather than being stuck outside the onion-like layers that otherwise develop.This also contributes to why the models at $q = 0.10$ and $q = 0.15$ don't seem to differ. Because with a higher chance to migrate less classes contribute to the onion-like layers. Thus this difference will be less prominent. I confirmed that agents move around more by comparing Figure 6 and Figure 7. Here we see that the standard deviation of the average network dividend is greater in the model with uncertainty. This is a direct consequence of the agents moving around more.

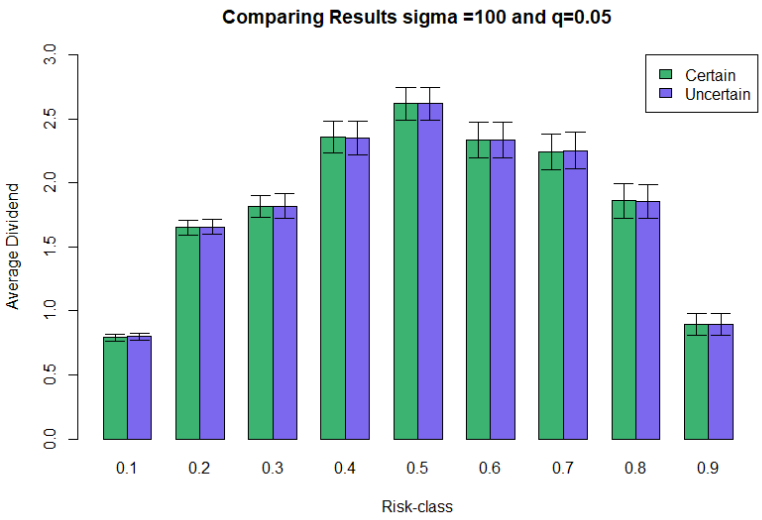

Figure 4: The network dividend the different risk-classes on average converge toward with $\sigma = 100$ and $q = 0.05$ paired t-test gave a p-value of 0.9968. The error-bars represent the standard deviation within the 500 values used to find the value it converges towards.

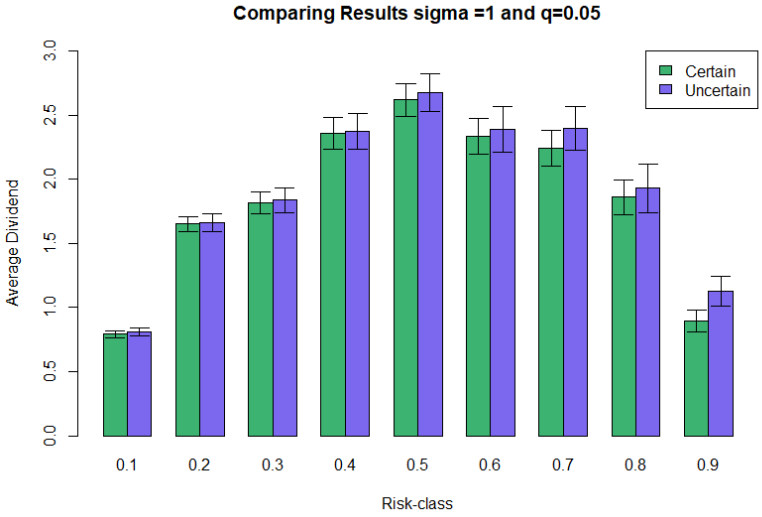

Figure 5: The network dividend the different risk-classes on average converge toward with $\sigma = 1$ and $q = 0.05$ paired t-test gave a p-value of 0.009152. The error-bars represent the standard deviation within the 500 values used to find the value it converges towards.

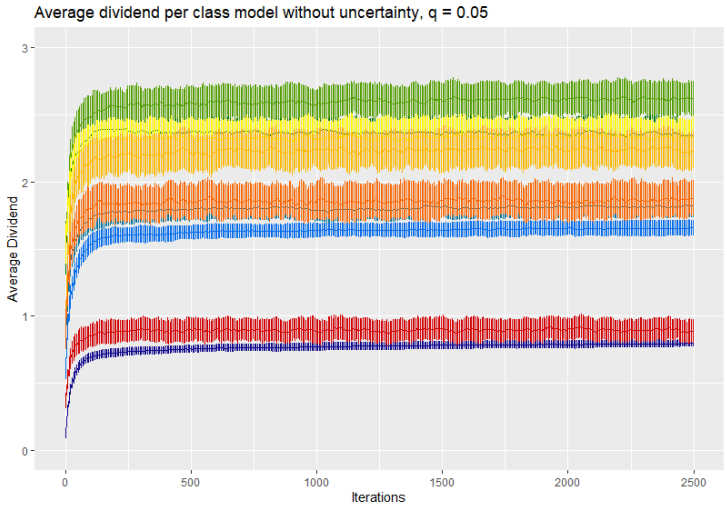

Figure 6: Results from my recreation of the Hegselmann and Flache model.The average surplus per class with $q = 0.05$. The error bars represent the standard deviation within the 100 runs.

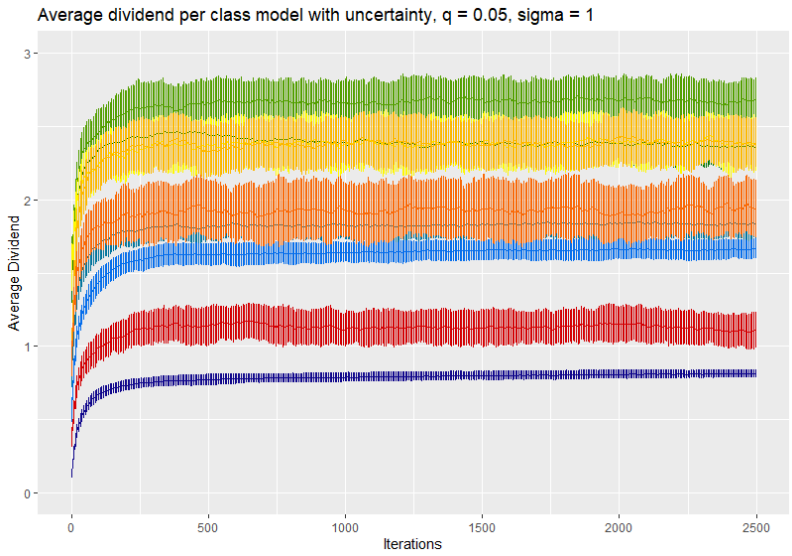

Figure 7: Results from my new model with uncertainty.The average surplus per class with $q = 0.05$ and $\sigma = 1$. The error bars represent the standard deviation within the 100 runs.

5 Discussion

First of I want to voice some reflecting criticism on my work. The the average network dividend graph of the new model show that the lines which represent averages of 100 runs are still noisy. I chose to do 100 runs for the recreation of the old model where the result ends up pretty stable, however I used the same parameters to test the new model with uncertainty. This was in hindsight a bad idea, as the resulting graph from the new model has far more fluctuation. Which makes sense as the agents move around more. It would have been better to use a larger sample size to get a better average. I'm not well versed in statistics, so when I used the t-tests I only tested the averages for each class. This doesn't take into account the number of measurements I took. On the other hand, a t-test with such a large sample size the result would quickly be significant. In the model with the introduced uncertainty, the agent first searches between the empty cells to find the best location. But during the evaluation of each location, he forgets if the neighbouring cells are empty or not. That is, of course, not very realistic.

Then for future research. Simulations are always simplifications of reality. This means that by making simulations you have to consider which parts you want to keep simple, and which you want to make more realistic. After concluding that the assumption of perfect information could have a significant impact. The next logical question is if it is worth adding the complexity in a simulation. In

this specific case, I would conclude it isn't. As almost all behavioural patterns, Hegselmann and Flache focus on are still the same. Still, I can only conclude this because I implemented it and saw the results. This rules nothing out either. Who knows what happens if we implement more complex systems. Like adding a knowledge acquisition and sharing systems. These could be interesting future research topics.

One might think Hegselmann and Flache did this in their own follow-up paper: Rationality vs. learning in the evolution of solidarity networks (1999). However, in this paper agents still have perfect information. They just have to learn how to handle this information rationally. They have to learn how profitable it is to work together with each class instead of not knowing of which class another agent is. Thus while they implement a knowledge system, it does not remove perfect information. There are already simulations which implement knowledge systems that do remove perfect information. Take for example the article from Dudek–Dyduch and Wąs (2006). In this article, they introduce a system for knowledge representation to a pedestrian dynamics simulation. They describe that by adding knowledge representation a whole new class of cellular automata opens which can make use of global relations instead of only local. Thus when implementing a rule that works over big global neighbourhoods, consider removing perfect information. There are also papers specifically about simulating knowledge spread within cellular automata. Su, Yang and Duan (2018) have published an article about knowledge dissemination inside knowledge-based organisations, in which they build the model using a cellular automaton. Further research can be done to see if their findings apply to other fields as well.

The crucial point I want to raise is that for future research, researchers should consider if the agents in their social simulations should have access to all information in their neighbourhood. But they should not be afraid to say that it isn't needed. As the added complexity might have little effect and harm efficiency and simplicity.

6 Conclusion

This experiment was done to see if the assumption of perfect information of agents within a cellular automaton had significant consequences for the resulting simulation. To test this, I introduced uncertainty into an existing social support network simulation from Hegselmann and Flache (1998). I did this by letting agent guess about future values when migrating.

In most cases, there was no significant difference to be found. However, we found that even with our very simple system, just adding uncertainty, we still had a two cases where it lead to a significant difference. Specifically, the cases with a small migration probability and a small enough σ. This means that agents having perfect information can make a significant difference in cellular automata based social simulations. For further research, it is thus important to keep this in mind, especially when implementing a feature over a large neighbourhood.

7 Literature

Dabbaghian, V., Jackson, P., Spicer, V., Wuschke, K. (2010). A cellular automata model on residential migration in response to neighborhood social dynamics. Mathematical and Computer Modelling, 52(9-10), 1752-1762.

Dabbaghian, V., Mago, V. K., Wu, T., Fritz, C., Alimadad, A. (2012). Social interactions of eating behaviour among high school students: a cellular automata approach. BMC medical research methodology, 12(1), 1-12.

Dudek–Dyduch, E., & Wąs, J. (2006, June). Knowledge representation of pedestrian dynamics in crowd: formalism of cellular automata. In International Conference on Artificial Intelligence and Soft Computing (pp. 1101-1110). Springer, Berlin, Heidelberg.

Flache, A., & Hegselmann, R. (1999). Rationality vs. learning in the evolution of solidarity networks: A theoretical comparison. Computational Mathematical Organization Theory, 5(2), 97-127.

Hamilton, J. D. (2020). Time series analysis. Princeton university press.

Hegselmann, R., & Flache, A. (1998). Understanding complex social dynamics: A plea for cellular automata based modelling. Journal of Artificial Societies and Social Simulation, 1(3).

Su, J., Yang, Y., & Duan, R. (2018). A CA-based heterogeneous model for knowledge dissemination inside knowledge-based organizations. Journal of Intelligent Fuzzy Systems, 34(4), 2087-2097.

8 Appendix

8.1 A

My implementation was programmed in NetLogo. NetLogo is a multi-agent programmable modelling environment. It has a built-in system of patches that's ideal for cellular automata. More information can be found on:
https://ccl.northwestern.edu/netlogo/
I used the included Behaviour Space tool to export data from multiple runs and used R for statistical analysis over that data.

My code isn't accessible to the public yet. To request access, you can reach me by emailing me at: joppe.bast@xs4all.nl.

8.2 B

0.70	0.74	0.77	0.80	0.82	0.83	0.84	0.83	0.82	0.80	0.77	0.74	0.70
0.74	0.78	0.81	0.84	0.87	0.88	0.88	0.88	0.87	0.84	0.81	0.78	0.74
0.77	0.81	0.85	0.88	0.90	0.92	0.92	0.92	0.90	0.88	0.85	0.81	0.77
0.80	0.84	0.88	0.91	0.94	0.95	0.96	0.95	0.94	0.91	0.88	0.84	0.80
0.82	0.87	0.90	0.94	0.96	0.98	0.98	0.98	0.96	0.94	0.90	0.87	0.82
0.83	0.88	0.92	0.95	0.98	0.99	1.00	0.99	0.98	0.95	0.92	0.88	0.83
0.84	0.88	0.92	0.96	0.98	1.00	1.00	1.00	0.98	0.96	0.92	0.88	0.84
0.83	0.88	0.92	0.95	0.98	0.99	1.00	0.99	0.98	0.95	0.92	0.88	0.83
0.82	0.87	0.9	0.94	0.96	0.98	0.98	0.98	0.96	0.94	0.90	0.87	0.82
0.80	0.84	0.88	0.91	0.94	0.95	0.96	0.95	0.94	0.91	0.88	0.84	0.80
0.77	0.81	0.85	0.88	0.90	0.92	0.92	0.92	0.90	0.88	0.85	0.81	0.77
0.74	0.78	0.81	0.84	0.87	0.88	0.88	0.88	0.87	0.84	0.81	0.78	0.74
0.70	0.74	0.77	0.80	0.82	0.83	0.84	0.83	0.82	0.80	0.77	0.74	0.70

Figure 8: Certainty window with $\sigma = 10$