
Modeling the Spread of Fashion Trends in Today's Society

A GESS project by

Forster Tim, D-CHAB, forstert@student.ethz.ch

Gschwend Oliver, D-MATL, olivergs@student.ethz.ch

Meiser Linda, D-CHAB, meiserl@student.ethz.ch

supervised by

Dr. Sanders Lloyd, D-GESS, lloyd.sanders@gess.ethz.ch

Dr. Woolley Olivia, D-GESS, olivia.woolley@gess.ethz.ch

December 17, 2017

ABSTRACT In a world where the acquisition of data and its interpretation is controlling most of the consumer market, with dynamic pricing targeting each individual's decisions in order to maximize profit, it is of highest essence to possess a model simulating the spread of trends. In order to establish a model predicting the flow of fashion, a cellular automaton was created with which various parameters, expressed as probabilities, were tested. These included the number of seeds (defined as the initial number of trend-followers), the effect of the environment (including Moore and a random neighbourhood), as well as the influence of a product's price. Social behaviour simulated had shown to follow patterns observed in reality, such as increased spreading rates when exposing a target person to a random network, representing the internet and traveling. Additionally a correlation between the price and willingness to buy a product is being established, with a factor accounting for resistance from people against elevated product prices. The simulation demonstrates how the chosen market parameters can be adjusted with empirical data in order to optimize the profit and sales of a product in a social system. With this solution, we introduce the world to a pathway of enormous potential financial savings.

CONTENTS

1. Individual Contributions and Declaration of Originality	3
2. Introduction and Motivation	4
3. Model Description	5
3.1. Terminology	5
3.2. Effect of Seeds	5
3.3. Effect of Environment	5
3.4. Effect of Price	6
3.5. Effect on Revenue	6
4. Implementation	7
4.1. Effect of Seeds	7
4.2. Effect of Environment	8
4.3. Effect of Price	8
4.4. Effect on Revenue	9
5. Simulation Results and Discussion	10
5.1. Simulation Process	10
5.2. Effect of Seeds	11
5.3. Effect of Environment	13
5.4. Effect of Price	14
5.5. Effect on Revenue	15
6. Summary and Outlook	16
References	17
A. Appendix	18
A.1. Matlab Codes	18
A.1.1. Simulation Codes	18
A.1.2. Results Generation and Plotting Codes	21

1. INDIVIDUAL CONTRIBUTIONS AND DECLARATION OF ORIGINALITY

DECLARATION OF ORIGINALITY The work presented was conducted in fulfillment of the requirements for the GESS course *Modelling and Simulating Social Systems with MATLAB* at the Swiss Federal Institute of Technology Zurich. We are aware that the work may be screened electronically for plagiarism. With our signatures we confirm that we are the sole authors of the written work here enclosed and that we have compiled it in our own words. Parts excepted are corrections of form and content by the supervisors. Previously published material of other people are indicated by a proper reference in the end of this project report. Additionally, we confirm that we have documented all methods, processes, and results truthfully, without manipulating data.


INDIVIDUAL CONTRIBUTIONS This work represents the product of the whole team's intellectual environment. Ideas were generated to various degrees by all members throughout the entire procedure of producing this simulation, report, and presentation. All members accept full responsibility of the entire manuscript. Main individual contributions can be summarized as follows, although cannot be quantified:

- **Tim Forster** contributed to the concept and design generation, simulation and modeling, report writing and to the presentation preparation
- **Oliver Gschwend** contributed to the concept and design generation and to the simulation and modeling
- **Linda Meiser** contributed to the concept and design generation and to the report writing and presentation preparation

Forster Tim
December 17, 2017



Gschwend Oliver
December 17, 2017



Meiser Linda
December 17, 2017



2. INTRODUCTION AND MOTIVATION

Generating data in the form of computer-mediated simulations offers a tremendously useful tool for pre-determining social network behaviours, especially in the areas of opinion or information spreading. This can be largely applied for deterministic consumer studies when the topology of the network and its effect on the information cycle are understood in order to develop mathematical algorithms accordingly. [1]

Opinion modeling offers a great basis for demonstrating the flow of information between individuals and larger groups of people. Relevant in this field of study is the so-called "0-1-2" effect, conveying that contact with one infected person (meaning possessing the opinion), will lead to a successful transmission with a lower probability than contact with two infected people. Two other main models are also relevant in describing the flow of information, the first one being the contagion approach, which is applied when modeling the spread of disease, and the second one being the "complex contagion" model. Described by the "complex contagion" approach, is the necessity of reinforcement of a certain piece of information as well as stimulation by multiple sources in order to result in transmission from one person to another. Here, the importance of strength of ties is brought into perspective. An actor will be more likely to accept a certain piece of information if reassured by trusted surroundings than when distant actors possess such. [2]

Intriguing in today's world is the evolution of fashion and thus it is of high value being able to predict the behaviour and response of certain trends to various stimuli. These may include price variations set by the producer, the total number of actors in the network as well as the initial number of infected actors (defined as the fraction of the population carrying a certain product as soon as it enters the given market). However, resistances such as a barrier to buying a certain product, given that the price is above a defined limit also exist in society and affect the spreading of fashion respectively. It would be advantageous to coordinate and direct the introduction of a consumer good by estimating how to achieve the largest market share.

We here present a model to predict materialistic fashion trends in a network representing society to answer the following research question: *In order to optimize revenues of selling a certain product in a social system, what marketing parameters have to be implemented and adjusted to achieve this goal?* By regarding the effect of several factors such as the number of influencers around a person, the number of infected actors present over time, the spreading mechanisms and various effects of price and resistances thereof, transmission of trends can be qualitatively simulated. As fashion has become a deterministic factor of our overall lifestyle and willingness to spend money, great insights can be provided for predicting the distribution of a product in a market when fashion tendencies can be modeled mathematically. We thus show what marketing parameters have to be implemented and adjusted in order to optimize the profit of selling a certain product in a social system.

3. MODEL DESCRIPTION

The model used is based on a cellular automaton, where objects are arranged in cells on a grid. Each cell represents an individual, who either follows the given trend or not (state 0 or 1). Each individual can be directly influenced by the surrounding individuals, represented by neighbouring cells. There is the possibility of including the von Neumann neighbourhood of four neighbours, or the extended surrounding of the Moore neighbourhood, where eight neighbours are included. When modeling a random neighbourhood, eight cells that are not necessarily positioned in close proximity are being considered. For modeling trends, it is important to quantify the fraction of neighbours who are infected with the trend, in order to predict the probability of infection. This probability will increase with an increasing number of infected neighbours. [3]

3.1. TERMINOLOGY

In order to describe the evolution of trends according to the parameters tested, a fundamental terminology needs to be established. When mentioning a person or group being "infected", it is implied that this person or group has accepted the given trend and has bought the respective product. This is equal to being an owner. The target person is represented by a cell in the automaton, which is observed while testing a certain effect, in order to see if that person accepts the trend or not. If someone is referred to as an influencer, the person already follows the trend. A neighbour is a person in the corresponding neighbourhood of the target person (within von Neumann, Moore, or the random environment, respectively). If referred to a pixel, one cell of the matrix is meant, thus one person.

3.2. EFFECT OF SEEDS

The value of the cells of the cellular automaton evolve by iteration in discrete time steps. One of the initial configurations set, is the number of seeds (a seed being a person who is initially infected with the trend). Depending on the initial number of seeds, the evolution of the fraction of infected people will differ. Thus, over a series of simulations and iterations, we aim to find the pattern of spreading of fashion, taking into consideration probabilities describing the effect of time as well as the effect of infected users.

3.3. EFFECT OF ENVIRONMENT

The environment of each cell in the cellular automaton determines whether or not the person represented by each cell accepts a trend or not, with a given threshold probability. When considering the Moore environment, 8 neighbors are potentially influencing the target person. However, this environment can be expanded in order to determine a random cell in the automaton to be the influencing

neighbour. This situation was investigated, as in everyday life, the internet and extensive traveling brings different groups of people together. This enhances the transfer of trends through various populations that per se are not in contact with each other besides virtually (through the internet) or by traveling.

3.4. EFFECT OF PRICE

Each cell in the automaton was given a random probability value serving as a threshold value, representing the willingness to pay for a given product. Depending on the price for a commodity, this threshold value was too high for the target person to buy the product and thus accept the trend, or it was low enough for the target person to accept the trend and buy the product. This gives extremely valuable insight into everyday life, especially as dynamic pricing of various industries spreads through society. Producers have to find exactly the right price for a product to target the people prone to buying it at the determined price directly. With this simulation, a platform for integrated consumer-pricing as well as price-resistance evaluation is established.

3.5. EFFECT ON REVENUE

Closely related to the price of the product are sales thereof. As our research question stated the intriguing challenge of adjusting and implementing various marketing parameters in order to optimize profit of selling commodities in a social system, it is imperative for us to, in the last section, focus on the potential of product revenue by simulating a correlation between the price, the fraction of owners and sales. With this simulation, we have aimed at giving a comparative standard for determining prices to achieve maximum revenue in order to fully answer the research question posed initially. The model for this simulation was taken equivalently to the model of the effect of the price, with the extension of adding the parameter of revenues.

4. IMPLEMENTATION

For testing various trends by means of a cellular automaton, we chose a continuous sheet with 70×70 cells, each cell representing a person in society. Iterations across all cells will be performed during multiple simulations, using probability models and threshold probabilities of each cell in order to monitor the trend evolution through a network.

4.1. EFFECT OF SEEDS

To investigate the influence of the number of seeds present on trend spreading through a given population, we chose to change the number of seeds present initially. We then observe the fraction of trend-infected people. Per seed, 10 simulations, each containing 1,000,000 iterations were averaged. Parameters were chosen to qualitatively show the response to various sources of stimuli. For empirical verification in the future, data sets could be fitted to obtain parameters tailored to each individual product. The total probability was then obtained from using the following relationship

$$p = p_1 \cdot p_2 \quad (1)$$

where p_1 and p_2 correspond to the following individual probabilities, where t is the dimensionless evolution of time, taking values from 0 to 1, and f is the fraction of neighbours who already are trend-infected, considering the Moore environment.

$$p_1 = a_1 \cdot (t + a_2) \cdot \exp(-a_3 \cdot (t + a_2)) \quad (2)$$

$$p_2 = 0.5 \cdot f \quad (3)$$

Here, p_1 is the probability that an individual accepts a certain trend over time. We consider that the longer the product is in the market, the lower the probability that other people will buy it. Additionally, the more people infected with the trend in the surroundings of a person, the higher the probability for the person to be infected themselves. With these two probabilities counter-actively influencing the total number of infected people, a maximum of the fraction of people carrying the trend will be established. The knowledge of how the number of seeds influences consumer behaviour towards the product represents a very important marketing parameter, which can then be adjusted with empirical data to optimize product revenues.

Having an expression for p_1 , the parameters $a = [a_1, a_2, a_3]$ were set to arbitrary default values $a = [20, 0.02, 15]$ to reach physically meaningful values for p_1 . By performing a sensitivity analysis, one can find the most sensitive parameter for this model. The normalized local sensitivity coefficient is calculated by using,

$$\bar{S}_{p_{1,max},i} = \frac{\partial p_{1,max}}{\partial a_i} = \frac{\log(\partial p_{1,max} / p_{1,max})}{\log(\partial a_i / a_i)} \approx \frac{\log(p_{1,max}^*) - \log(p_{1,max})}{\log(a_i^*) - \log(a_i)} \quad (4)$$

where $p_{1,max}^*$ and a_i^* are the maximum probability ($p_{1,max}$) for a parameter $a_i^* = a_i(1 + \delta)$, having $\delta = 0.05$ to include a small disturbance.

4.2. EFFECT OF ENVIRONMENT

To investigate the influence of the environment on trend spreading through a given population, it is vital to consider the modes of contact between people. A target person having very strong ties with only a few friends (cells in the Moore) may have strong influencers, but only very few influencers. [2] Expanding to the Moore network adds a few weaker ties to the strong ties and considers more distant acquaintances. However, as the internet and traveling are a major source of information spreading and thus the evolution of fashion and trends thereof, a random network of cells was considered (consisting of very weak ties) for gaining a more realistic spectrum of pathways for trend transmission.

Here, the Moore and random environment were investigated, using 100 simulations with 1,000,000 iterations each. The number of seeds was chosen to be 30. For analysis, the probabilities as shown in equations 2 and 3 were used with a total probability represented by equation 1. When implementing the influence of the environment on spreading of trends, an utterly indicative parameter is obtained for being adjusted in order to eventually maximize profits.

4.3. EFFECT OF PRICE

Price regulation of products and the effect thereof with respect to consumer behaviour is a very valuable parameter to be aware of when trying to sell a certain commodity. To investigate the effect of a product's price on the spreading of a trend, a probability representing the price as well as the resistance to elevated prices was developed, and can be found as equation 5, where P is the price and $\tau = 1$.

$$p_3 = \exp\left(\frac{-P}{\tau}\right) \quad (5)$$

Here, τ represents the resistance from people towards high pricing. If someone is rich, the affinity to buying highly-priced products is much higher, than if someone doesn't have any money to spend. Thus, a $\tau = 1$ represents zero resistance, meaning people considered for this simulation have a lot of money. The total probability of accepting a trend, including the effect of price, and seeds was then defined as shown in equation 6.

$$p = p_1 \cdot p_2 \cdot p_3 \quad (6)$$

As the simulation was modeled with a cellular automaton in which each cell was given a random threshold probability, which was being used to judge whether or not a price was too high for an individual or not, every iteration offered a novel chance of each target cell to accept the trend. When the probability of the price function was higher than the innate (random) probability given to each

cell, the trend was accepted. If not, the trend was not accepted. Thus, the spreading of the trend could be simulated for given prices. A total of 20 simulations with 1,000,000 iterations was averaged with an initial number of seeds of 90 and a Moore environment.

4.4. EFFECT ON REVENUE

To investigate the effect of price on potential revenues that can be generated by selling a product, this last investigation was made to fully answer the initial research question. The following equation for estimating revenues was used, where R stands for revenues, P , is the price, and x the fraction of the total population, F .

$$R/F = P \cdot x \quad (7)$$

From the simulation, the final fraction of owners after 1,000,000 iterations was investigated over a dimensionless price range from 0.1 to 1. The product of price and fraction of owners after each simulation was then plotted as revenue and can be seen to show a maximum. From this maximum, thus, the optimal price can be determined for selling a certain product. With this optimal price, another very important marketing parameter has been determined. Using the graphics presented, it can be adjusted using the models for any set of empirical data, such that personalization of the model to producers can be guaranteed.

5. SIMULATION RESULTS AND DISCUSSION

5.1. SIMULATION PROCESS

The simulation process is shown in figure 1. What can be seen is the cellular automaton created, and its evolution throughout the iterative process. The yellow cells represent the infected people, and thus initially represent the number of seeds present. During each iteration, the neighbourhood of a target cell is tested whether or not the probability of accepting a trend is higher than the cell's threshold level or not. In this figure, two scenarios are being compared. The first, shown by sub-figures 1a, 1b, 1c, and 1d represents the spreading of a trend in a Moore environment. In sub-figures 1e, 1f, 1g, 1h, trend-spreading is shown at the same iteration steps as the upper Moore equivalents. A clear difference can be observed, which will be explained in more detail in section 5.3.

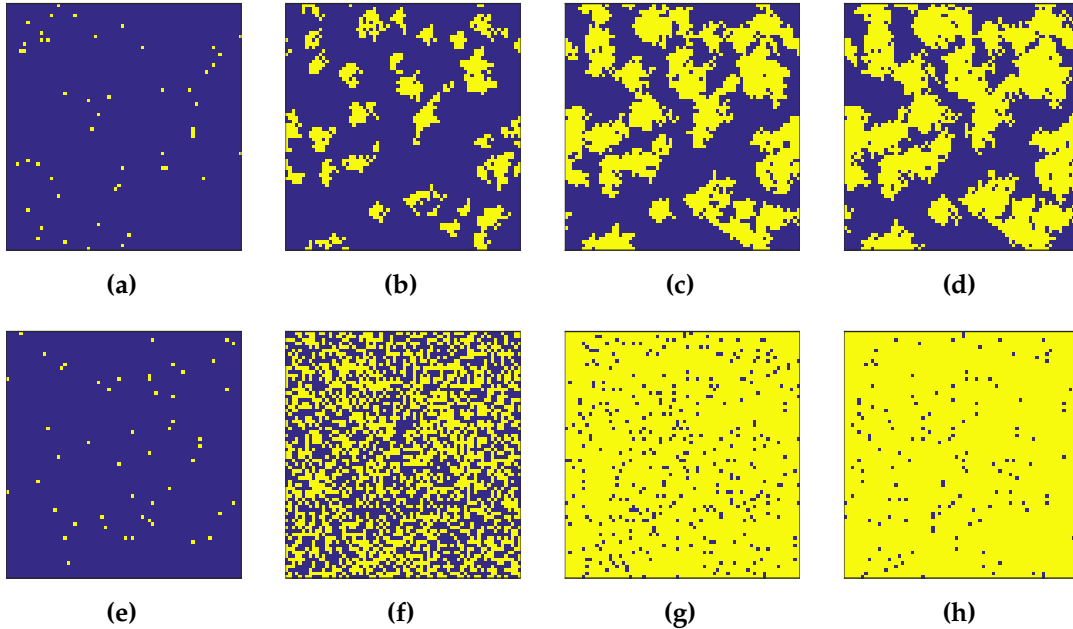


Fig. 1: Cellular automaton with cells representing the infected state when taking a value of 1 (represented by yellow pixels), and non-infected states when taking a value of 0 (represented by blue pixels). Spreading of the trend in a Moore environment can be observed in figures 1a, 1b, 1c, and 1d, whereas spreading of the trend in a random environment is represented in figures 1e, 1f, 1g, 1h. Figures in each column are taken at the same iterative step. Homogeneity of the two different environments as well as the rate of transmission can be observed.

The cellular automaton was chosen to be continuous, which is not represented by the above figures. However, the pattern of trend evolution can well be observed. It is interesting to note the speed of spreading, as well as the homogeneity of spreading in the two cases, which will be established further in the corresponding section.

5.2. EFFECT OF SEEDS

In a first step, equation (4) was applied to p_1 to check how sensitive the chosen parameters are. In case of high sensitivities, the model for p_1 should therefore be rearranged, since in this work the parameters cannot be cross validated with experimental data. Considering the time evolution of the sensitivity, one can observe a significantly decreasing sensitivity coefficient $\bar{S}_{p_1, \max, 3}$ for parameter a_3 , which is shown in figure 2. Nevertheless, since due to the exponential shape the probability is approaching zero after a very short time t , eliminating the influence of the sensitive parameter, the expression for p_1 is used for further investigations. If experimental data would be used, the focus should be laid on the parameter a_3 due to the highest sensitivity coefficient.

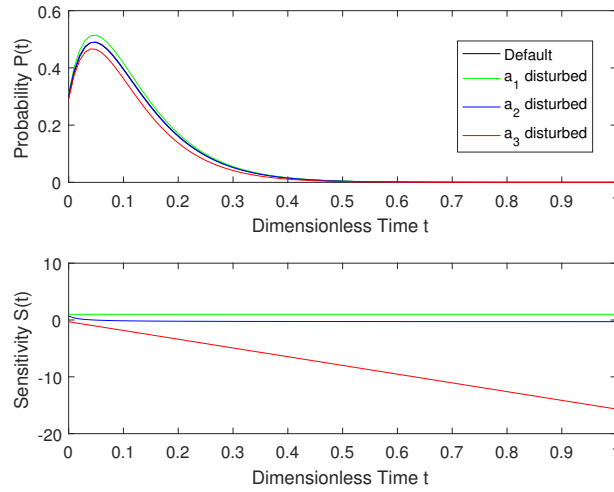


Fig. 2: The probability $p_1(t)$ (top) and normalized sensitivity coefficient $\bar{S}_{p_1, \max, i}(t)$ (bottom) are shown as a function of time t . The chosen parameters $a = [20, 0.02, 15]$ are used for further investigation since the parameter a_3 is sensitive just after a certain time $t \approx 0.25$, implying a small change in parameters are not effecting the model significantly.

When modeling the effect of seeds initially present in the system investigated, the following results were obtained. As the fraction of seeds initially is 0, statistically, no one will accept the trend. This goes to show that bringing a product into an already existing market is rather complicated indicating the necessity of improved marketing strategies. These could include a time period of free-product distribution, where the producer hands out a free sample to a group of people to show what has been developed. As soon as this barrier is broken, the evolution of infecting people follows a trend, which is shown in figure 3.

From the simulation, it can be seen that the more seeds present in a system initially, the faster the initial infection of people, and the larger the resulting fraction of owners the trend can reach. Comparing this trend with reality, close parallels can be observed. In everyday life, the more people

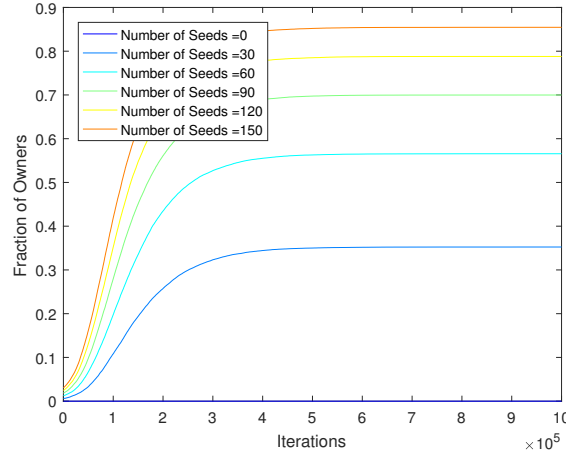


Fig. 3: Evolution of fraction of owners as a function of iterations for different numbers of seeds can be seen. When increasing the number of seeds, the initial rate of accepting the trend as well as the final fraction of owners increases.

carrying a certain trend, the more likely it is for the trend to spread. However, as soon as too many people carry the trend, there is a barrier to spreading, and the number of people owning the trendy object stagnates. Exactly this is shown by the simulation, where the fraction of owners shows to go to a limit after a certain number of iterations.

In figure 4, the fraction of owners as a function of seeds is plotted with the standard deviation of simulations shown as error bars. It can be seen that as the number of seeds is becoming increasingly

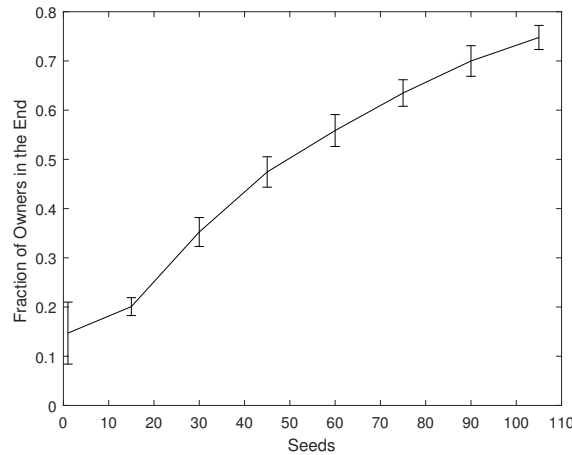


Fig. 4: Fraction of owners as a function of seeds plotted at for 10 simulations of 1,000,000 iterations each. It can be seen that as the number of seeds present in the system initially increases, the resultant fraction of owners of the product increases with a decreasing rate. Error bars show the standard deviation of 10 simulations per seed.

larger, the fraction of owners that is infected with the trend at the end of 1,000,000 iterations increases, reaching a plateau. This shows that the more seeds initially following a trend, smaller the change in fraction of owners affected after all iterations are completed.

5.3. EFFECT OF ENVIRONMENT

In this simulation, where the extension of the network was included, a more complex but realistic picture of trend evolution in society can be shown. Here, the inclusion of having access to the internet and being able to travel shows to largely increase the rate at which a trend is spread, as well as the total number of trend-carriers at the end of the simulation. This gives an estimate of the gigantic influence media and internet transmissions can have on the connectivity of people in today's society. According to this simulation, which of course does not take into account any legal limitations, getting a product online and spreading the trend throughout different groups of people has a huge effect on the rate of transmission.

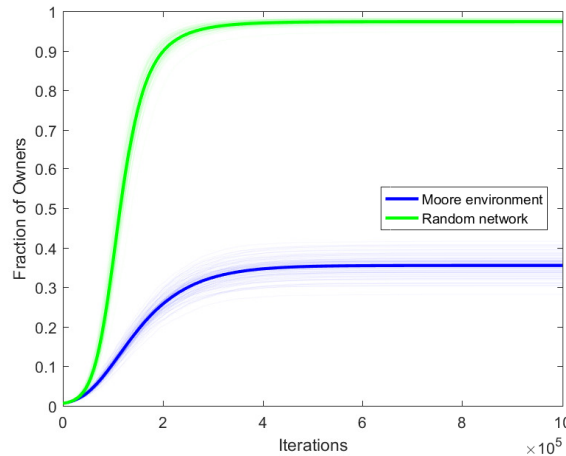


Fig. 5: Comparison of Moore and random environment behaviour on the fraction of owners when modeled as a function of iterations. Here, 100 simulations with 1,000,000 iterations each were used. The lightly plotted lines show results of all simulations, whereas the solid line shows the average result of all simulations. The trend observed shows that a random environment reaches a much larger fraction of people much quicker than a standard Moore environment.

Figure 5 contrasts the spread through the network of friends (represented by the Moore environment) with the random network. All runs are plotted by transparent curves. The major, bold line of the plot represents the average of all simulations. From this, a solution for optimizing the target group of expected customers for a product can be constructed by tailoring the marketing strategy to use the influence of media.

The direct comparison of the pattern of evolving trends can be found in figure 1, where transmis-

sion using the Moore environment (in the upper sub-figures) is plotted as well as transmission using a random environment (in the lower sub-figures) for the same iteration steps. It can be seen that when the network of contact is chosen randomly, the trend spreads much more homogeneously, and more people accept the trend in a shorter time. For these two scenarios, two different marketing strategies could be developed depending on the desired homogeneity-requirement of the trend.

In contrast to a very homogeneous spreading in the random environment is the observation that with a Moore environment, transmission occurs in patches. Around one seed, a convolution of infections occurs, thus spreading trends through the periphery of established infected groups for each iteration.

5.4. EFFECT OF PRICE

When simulating the effect of product-pricing on the fraction of owners, it could be found that as the dimensionless price increases, the resultant fraction of owners as well as the rate of accepting the trend over all iterations decreases. This trend can well be seen in figure 6, where the fraction of owners is plotted along all iterations for different product prices. The plot nicely shows the plateau, which is reached, indicating that a maximum of owners does exist. This means that some people will never buy the product, and will thus never follow the trend established with such product.

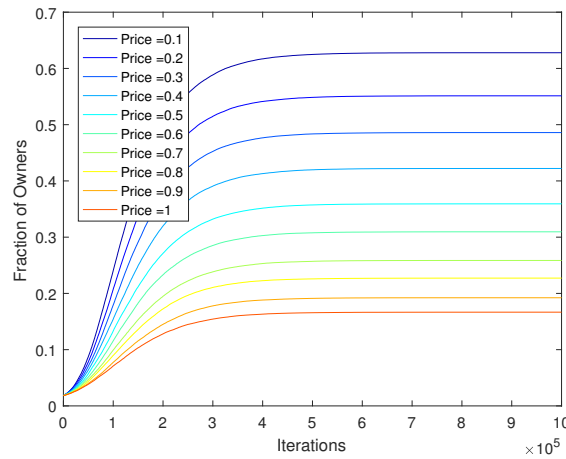


Fig. 6: Fraction of owners in dependence of iterations. For 1,000,000 iterations, a different dimensionless price was set. A clear trend can be seen, namely that when the price increases, the final fraction of owners decreases, as fewer people are willing to spend money for the expensive trend.

Reality is very well reflected by this simulation. When regarding the mobile phone industry, for example, no matter what trend is set by one brand, there will always be some people in society who are more prone to buying product of the other brand, as certain factors of the brand personalize the

mobile phone experience. These factors preventing consumers from buying the product show the effectiveness of estimating a resistance to prices.

5.5. EFFECT ON REVENUE

Figure 7 compares the price, fraction of owners and revenue in one plot in order to graphically evaluate the price at which maximum revenues are achieved. It can be seen that as the price increases linearly, the resultant fraction of owners decreases, as established in the previous section, in figure 6. When plotting the revenue, a maximum can be found and the price thereof can be determined.

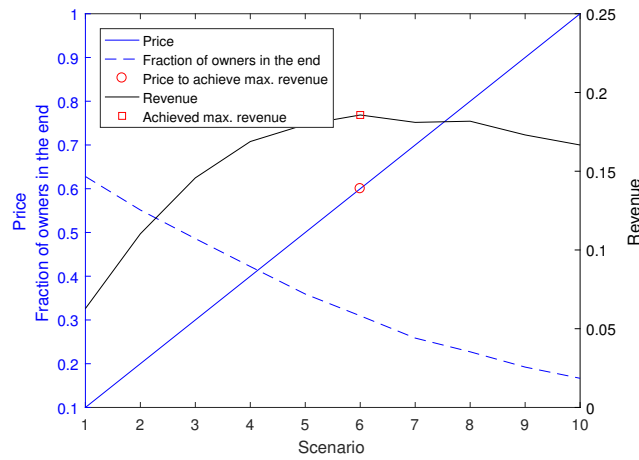


Fig. 7: Fraction of owners at the end of 1,000,000 iterations plotted together with the price and calculated revenues. A maximum revenue is found and the corresponding product price is marked in red. This given an indication on how to optimally price products in order to achieve maximum revenues.

From the information obtained through such correlations, a great benefit to society can be generated. For reality predictions, companies can fit their consumer data to generate the respective parameters in the probabilities as established for this simulation, and find the tailored data-fits. With these, it will be possible to predict consumer behaviour to a certain extent, from which it will be possible to estimate sales potential given an optimum price. This will have a great effect on the marketing strategies companies are investing in. Adjusting these according to our simulations' results will help society maximize its utility.

6. SUMMARY AND OUTLOOK

In a world where data generation and its excessive acquisition increasingly takes over all sectors of industry, a prediction model for companies to assess the transmission of trends can offer a very valuable tool. This potential for the market led to the investigation of different marketing parameters and a simulation of how to implement them in order to optimize the profit when selling a certain product in a social system.

This work presents a method for modeling the social behaviour of trend-spreading in a network of populations, with the focus on the number of seeds, and thus the size of the initial population being aware of the new product. It was found that the larger this groups is, the faster the trend will be accepted by other members of society. However, a plateau representing the maximum number of people who have accepted the trend was always reached. The environment around the targeted people was also tested. This was done with the Moore surroundings, representing close friends and more distant acquaintances. Increasing the number of these ties showed that the fraction of owners increased over time. However, a random network was also tested, in which the targeted person was put in contact with any random group of people, also representing the unknown and thus weaker ties. However, this showed much faster spreading of a trend, as contact between random social groups increases the exposure to yet unknown trends. The third marketing factor evaluated was pricing of a product. Being able to adjust the resistance towards high prices enables the producer to tailor the model to a very specific social group. The target group can thus be chosen according to income and the ability to spend money.

The parallels to reality are as follows: Increasing the number of seeds could be achieved by various advertisement and initial marketing strategies. Thus, handing out free samples of a novel product may elevate the number of seeds. As compared to the simulation, there will nearly always be a limit to the maximum number of people accepting a trend, as branding plays an important role on the market. Not everyone will buy a fashionable product from the same brand. Thus there will always be the remainder of people who sticks to other principles and does not buy the product. Additionally, the factor of the environment was tested, which in reality represents the huge influence of media, the internet, and traveling. As expected, it could be seen how powerful the internet can be in spreading information. The simulation resulted in a highly increased spreading rate when such tools were made available. Simulation of product pricing showed the behavioural trend of having a resistance towards buying such product as prices increased. This can be observed in reality, as increased prices reduces the purchasing power of each individual, when salaries remain constant. Thus, if a product is priced low, more people will be able to afford it.

For future work, empirical data fitting would be an interesting addition to this simulation. With parameters obtained from such information, more product-specific behaviour could be determined

which would offer an even greater insight into potential marketing methods for companies selling their products. Additionally, as the simulation does not take into account disparities between different social classes, a future project could be to take into account that different social classes act in different social systems. This could enable tailoring of product marketing campaigns to respective social classes targeted.

REFERENCES

- [1] I. Kanovsky, O. Yaari, *IEEE SocialCom* **2013**, 971–974.
- [2] D. Centola, M. Macy, **2007**, *113*, 702–734.
- [3] S Wolfram, *Statistical mechanics of*, Vol. 55, **1983**, pp. 1–16.

A. APPENDIX

A.1. MATLAB CODES

A.1.1. SIMULATION CODES

Listing 1: Main file to start the simulation.

```
1 function[M,p] = simulateOneStep(M,parameters)
2
3 %choose one site randomly
4
5 l = parameters.sideLength;
6 progressInSimulation = parameters.progressInSimulation;
7 i = randi(l);
8 j = randi(l);
9
10 %check if the value is already 1. if yes, exit the fundction
11 if(M(i,j)==1)
12     p = 1;
13 return;
14 end
15
16 %If the site contains a zero, compute if it changes its value to one (buy the product)
17
18 %Here calculate the probability p
19 p = ProbabilityMaker(parameters, M, i, j);
20
21 c = rand();
22 if(p>=c) % accept the action with probability p
23 M(i,j)=1;
24 end
```

Listing 2: Function to calculate the fraction of neighboring owners.

```
1 function[f] = getFractionOfNeighboringOwners(M,i,j,parameters)
2
3 %This function returns the fraction of people in the neighborhood, which already have the
  product. (Ebtry in Matrix == 1)
4 %In this case an infinite grid with periodic overlapp is used.
5
6 nnCondition = parameters.nnCondition;
7 l = parameters.sideLength;
8
9 %This function returns the corrected index for periodic boundary conditions
10 function[correctedIndex] = getInfIndex(index,l)
11 correctedIndex = index-1*floor((index-1)/l);
12 end
13
14 if(strcmp(nnCondition,'vonNeumann'))
```

```

15 f = (...
16 M(getInfIndex(i-1,1),getInfIndex(j,1))+...
17 M(getInfIndex(i,1),getInfIndex(j-1,1))+...
18 M(getInfIndex(i+1,1),getInfIndex(j,1))+...
19 M(getInfIndex(i,1),getInfIndex(j+1,1)))/4;
20 end
21
22 if(strcmp(nnCondition,'moore'))
23 f = (...
24 M(getInfIndex(i-1,1),getInfIndex(j,1))+...
25 M(getInfIndex(i+1,1),getInfIndex(j,1))+...
26 M(getInfIndex(i-1,1),getInfIndex(j-1,1))+...
27 M(getInfIndex(i+1,1),getInfIndex(j-1,1))+...
28 M(getInfIndex(i-1,1),getInfIndex(j-1,1))+...
29 M(getInfIndex(i+1,1),getInfIndex(j-1,1))+...
30 M(getInfIndex(i,1),getInfIndex(j+1,1))+...
31 M(getInfIndex(i,1),getInfIndex(j-1,1)))/8;
32 end
33
34 if(strcmp(nnCondition,'randomNetwork'))
35 f = (...
36 M(randi(1),randi(1))+...
37 M(randi(1),randi(1))+...
38 M(randi(1),randi(1))+...
39 M(randi(1),randi(1))+...
40 M(randi(1),randi(1))+...
41 M(randi(1),randi(1))+...
42 M(randi(1),randi(1))+...
43 M(randi(1),randi(1)))/8;
44 end
45 end

```

Listing 3: File to calculate the fraction of overall owners.

```

1 function[fractionOfOwners] = getFractionOfOwners(M,parameters)
2
3 %This function returns the fraction of product owners in total. The Matrix entries are assumed
   to be
4 %1=Product owner or 0 = not a product owner
5
6 N = (parameters.sideLength)^2;
7 fractionOfOwners = sum(sum(M))/N;
8 end

```

Listing 4: Function to initialize the total population matrix.

```

1 function[M] = initializeMatrix(parameters)
2
3 %This function initializes the grid on which the simulation run.
4 %First all sites contain a zero.
5

```

```

6 sideLength=parameters.sideLength;
7 M = zeros(sideLength,sideLength);
8 end

```

Listing 5: Function to calculate the probability if the person should buy the product or not.

```

1 function[probability] = ProbabilityMaker(parameters, M, i, j)
2
3 %     method = parameters.ProbabilityCalculationMethod;
4
5     f = getFractionOfNeighboringOwners(M,i,j,parameters);
6     t = parameters.progressInSimulation;
7
8 %calculate the probability wich decreases in time. The product gets less
9 %famous with longer time on the market
10 p1 = exp(-t*8);
11
12 %calculate the probability of the neighbor interaction. It is just a linear
13 %function starting at 0.2 for no neighbor and 1 if all neighbors are
14 %product owner.
15 if(f == 0)
16     p2 = 0;
17 else p2 = 0.2 + 0.8/8*f;
18 end
19
20 %Calculate the probability dependent on the price
21
22 % z          ...dimensionless price of the opinion/trend/product
23 % tauz       ...Resistance towards a high price (people with more money
24 %            have a higher tau, so they buy an expensive product much faster)
25
26 %     P = exp(-z ./ tauz);
27
28
29     p = [p1, p2];
30
31 %     if strcmp(method, 'MultiplicativeProbability')
32 %         probability = prod(p);
33 %     elseif strcmp(method, 'MinumumAdjustedMultiplicativeProbability')
34 %         Probability = prod(p) ./ min(p);
35 %     elseif strcmp(method, 'MaximumAdjustedMultiplicativeProbability')
36 %         Probability = prod(p) ./ max(p);
37 %     elseif strcmp(method, 'MultiplicativeProbabilityWithNeglection')
38 %         p(p < 1e-4) = [];
39 %         Probability = prod(p);
40 %     end
41 end

```

Listing 6: Function to set an initial number of owners (seeds).

```

1 function[M] = seedProductOwner(M,parameters);
2
3 %This function seeds the number of seeds randomly in the matrix
4 seeds = parameters.seeds;
5 l = parameters.sideLength;
6
7 %Check if the user has chosen too much seeds (more than matrix seeds)
8 if(seeds>l^2)
9 error('you have choosen to much seeds')
10 end
11
12
13 counter = 0;
14 while(counter ~= seeds)
15     i = randi(l);
16     j = randi(l);
17     if(M(i,j)~=1)
18         M(i,j)=1;
19         counter = counter +1;
20     end
21 end
22 end

```

A.1.2. RESULTS GENERATION AND PLOTTING CODES

Listing 7: Function to generate the results based on a moore network.

```

1 %This is the main script. Here we run our Experiments
2
3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets L^2 sites in the simulation
5 parameters.price = 0.1; %Price for the product
6 parameters.resistivity = 0.5; %Ressistivity to price. E.g r=1 means rich people r = 0; ultra
    poor people
7 parameters.seeds = 30; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'moore'; %One can choose between Von Neumann ="vonNeumann", Moore "
    moore" nearest neighbor interaction
9                                     %Or "randomNetwork" to choose 4 neighbors by random out of
                                     the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.
    progress = actual step/steps
11
12
13
14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');

```

```

18
19
20
21
22 for k = 1:100
23
24 waitbar(1/100*k)
25
26 Results(k).environment = 'moore';
27
28 M = initializeMatrix(parameters);
29 M = seedProductOwner(M,parameters);
30 fractionOfOwners = zeros(1,steps);
31 probability = zeros(1,steps);
32
33
34 for i = 1:steps
35     parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
36     fractionOfOwners(i) = getFractionOfOwners(M,parameters);
37     [M,p] = simulateOneStep(M,parameters);
38     probability(i)=p;
39
40 end
41
42
43 Results(k).fractionOfOwners = fractionOfOwners;
44 Results(k).probability = probability;
45 end
46
47 close(h)

```

Listing 8: Function to generate the results based on a random network.

```

1 %This is the main script. Here we run our Experiments
2
3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets  $L^2$  sites in the simulation
5 parameters.price = 0.1; %Price for the product
6 parameters.resistivity = 0.5; %Resistivity to price. E.g r=1 means rich people r = 0; ultra
    poor people
7 parameters.seeds = 30; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'randomNetwork'; %One can choose between Von Neumann ="vonNeumann",
    Moore "moore" nearest neighbor interaction
9                                     %Or "randomNetwork" to choose 4 neighbors by random out of
                                     the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.
    progress = actual step/steps
11
12
13

```

```

14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');
18
19
20
21
22 for k = 1:100
23
24 waitbar(1/100*k)
25
26 Results(k).environment = 'randomNetwork';
27
28 M = initializeMatrix(parameters);
29 M = seedProductOwner(M,parameters);
30 fractionOfOwners = zeros(1,steps);
31 probability = zeros(1,steps);
32
33
34 for i = 1:steps
35     parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
36     fractionOfOwners(i) = getFractionOfOwners(M,parameters);
37     [M,p] = simulateOneStep(M,parameters);
38     probability(i)=p;
39
40 end
41
42
43 Results(k).fractionOfOwners = fractionOfOwners;
44 Results(k).probability = probability;
45 end
46
47 close(h)

```

Listing 9: Function to plot the results based on a moore and random network.

```

1 %This is the main script. Here we run our Experiments
2
3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets L^2 sites in the simulation
5 parameters.price = 0.1; %Price for the product
6 parameters.resistivity = 0.5; %Resistivity to price. E.g r=1 means rich people r = 0; ultra
    poor people
7 parameters.seeds = 30; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'moore'; %One can choose between Von Neumann ="vonNeumann", Moore "
    moore" nearest neighbor interaction
9
    %Or "randomNetwork" to choose 4 neighbors by random out of
    the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.

```

```

    progress = actual step/steps
11
12
13
14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');
18
19
20
21
22 for k = 1:100
23
24     waitbar(1/100*k)
25
26     Results(k).environment = 'moore';
27
28     M = initializeMatrix(parameters);
29     M = seedProductOwner(M,parameters);
30     fractionOfOwners = zeros(1,steps);
31     probability = zeros(1,steps);
32
33
34     for i = 1:steps
35         parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
36         fractionOfOwners(i) = getFractionOfOwners(M,parameters);
37         [M,p] = simulateOneStep(M,parameters);
38         probability(i)=p;
39
40     end
41
42
43     Results(k).fractionOfOwners = fractionOfOwners;
44     Results(k).probability = probability;
45 end
46
47 close(h)

```

Listing 10: Function to generate graphs of the matrices.

```

1 %This is the main script. Here we run our Experiments
2
3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets  $L^2$  sites in the simulation
5 parameters.price = 0; %Price for the product
6 parameters.resistivity = 0.5; %Resistivity to price. E.g r=1 means rich people r = 0; ultra
    poor people
7 parameters.seeds = 50; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'randomNetwork'; %One can choose between Von Neumann ="vonNeumann",

```



```

    Moore "moore" nearest neighbor interaction
9                                     %Or "randomNetwork" to choose 4 neighbors by random out of
                                     the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.
    progress = actual step/steps
11
12
13
14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');
18 i = 0;
19
20
21
22
23
24 M = initializeMatrix(parameters);
25 M = seedProductOwner(M,parameters);
26 %fractionOfOwners = zeros(1,steps);
27 %probability = zeros(1,steps);
28 figure
29 imagesc(M)
30 xlabel([strcat('- matrix after-', num2str(i),'-steps');'- random network'])
31
32 for i = 1:steps
33     waitbar(1/steps*i)
34     parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
35     %fractionOfOwners(i) = getFractionOfOwners(M,parameters);
36     [M,p] = simulateOneStep(M,parameters);
37     %probability(i)=p;
38     if mod(i,100000)==0
39         figure
40         imagesc(M)
41         xlabel([strcat('- matrix after-', num2str(i) ,'-steps');'- random network'])
42     end
43
44
45 end
46
47 imagesc(M)
48 xlabel([strcat('matrix after',' ',num2str(steps),' ', ' steps');'moore environment';'started
    with 5 seeds'])
49
50 close(h)

```

Listing 11: Function to generate the results for different amounts of seeds.

```

1 %This is the main script. Here we run our Experiments
2

```

```

3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets  $L^2$  sites in the simulation
5 parameters.price = 0.1; %Price for the product
6 parameters.resistivity = 0.5; %Resistivity to price. E.g r=1 means rich people r = 0; ultra
    poor people
7 parameters.seeds = 3; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'moore'; %One can choose between Von Neumann ="vonNeumann", Moore "
    moore" nearest neighbor interaction
9                                     %Or "randomNetwork" to choose 4 neighbors by random out of
                                     the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.
    progress = actual step/steps
11
12
13
14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');
18 counter = 0;
19
20
21
22 for d = 0:2:20
23     counter = counter +1
24     waitbar(1/11*counter)
25     Results(counter).numberOfSeeds = d;
26     parameters.seeds = d;
27     SumfractionOfOwners = zeros(1,steps);
28     Sumprobability = zeros(1,steps);
29
30     for averaging = 1:20;
31
32         M = initializeMatrix(parameters);
33         M = seedProductOwner(M,parameters);
34         fractionOfOwners = zeros(1,steps);
35         probability = zeros(1,steps);
36
37
38         for i = 1:steps
39             parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
40             fractionOfOwners(i) = getFractionOfOwners(M,parameters);
41             [M,p] = simulateOneStep(M,parameters);
42             probability(i)=p;
43
44         end
45
46     SumfractionOfOwners = SumfractionOfOwners + fractionOfOwners;
47     Sumprobability = Sumprobability + probability;

```

```

48 end
49
50 Results(counter).fractionOfOwners = SumfractionOfOwners/10;
51 Results(counter).probability = Sumprobability/10;
52 end
53
54 close(h)

```

Listing 12: Function to generate the graphs for a different amount of seeds.

```

1
2 %Average over the trials
3
4 [fileName,pathName] = uigetfile('*.mat','Select the data to plot')
5 load(fileName)
6
7
8
9 for i = 1:6
10 plot(Results(i).fractionOfOwners)
11 hold on
12 title('evolution of fraction of owners dependent on Seeds')
13 xlabel('iterations');
14 ylabel('fraction of owners')
15 end
16
17
18 str = {strcat('Number of Seeds = ', num2str(Results(1).numberOfSeeds))};
19 for i = 2:6
20 % at the end of first loop, z being loop output
21 str = [str , strcat('Number of Seeds = ', num2str(Results(i).numberOfSeeds))]; % after 2nd
    loop
22 % plot your data
23 legend(str{:},'Location','northwest');
24 end
25
26 limitValue = [];
27 seeds = 0:30:150;
28
29 for i = 1:6
30 limitValue = [limitValue,Results(i).fractionOfOwners(end)]
31 end
32 figure
33 plot(seeds,limitValue)
34 title('fraction of owners at the end dependent on Seeds')
35 xlabel('number of seeds');
36 ylabel('fraction of owners')

```

Listing 13: Function to generate the results dependent on different prices.

```

1 %This is the main script. Here we run our Experiments
2
3 %%List of parameters
4 parameters.sideLength = 70; %Length of matrix. One gets  $L^2$  sites in the simulation
5 parameters.price = 0.1; %Price for the product
6 parameters.resistivity = 0.5; %Resistivity to price. E.g  $r=1$  means rich people  $r = 0$ ; ultra
    poor people
7 parameters.seeds = 90; %amount of seeds in the matrix, which contain a one. Sites are picked
    randomly.
8 parameters.nnCondition = 'moore'; %One can choose between Von Neumann ="vonNeumann", Moore "
    moore" nearest neighbor interaction
9                                     %Or "randomNetwork" to choose 4 neighbors by random out of
                                     the lattice
10 parameters.progressInSimulation = 0; % The progress in simulation, a normalised time measure.
    progress = actual step/steps
11
12
13
14
15
16 steps = 1000000;
17 h = waitbar(0,'Simulating...');
18 counter = 0;
19
20 for price = 0.1:0.1:1
21     counter = counter +1;
22     waitbar(1/10*counter)
23
24     parameters.price = price;
25     Results(counter).price = price;
26     Results(counter).environment = 'moore';
27     Results(counter).numberOfSeeds = parameters.seeds;
28
29     SumFractionOfOwners = zeros(1,steps);
30     SumProbabilities = zeros(1,steps);
31
32 for k = 1:20
33
34
35 M = initializeMatrix(parameters);
36 M = seedProductOwner(M,parameters);
37 fractionOfOwners = zeros(1,steps);
38 probability = zeros(1,steps);
39
40
41 for i = 1:steps
42     parameters.progressInSimulation = i/steps; %Adjust this parameter to measure time evolution
43     fractionOfOwners(i) = getFractionOfOwners(M,parameters);
44     [M,p] = simulateOneStep(M,parameters);
45     probability(i)=p;

```

```

46
47 end
48
49     SumFractionOfOwners = SumFractionOfOwners + fractionOfOwners;
50     SumProbabilities = SumProbabilities + probability;
51
52
53 end
54 Results(counter).fractionOfOwners = SumFractionOfOwners/20;
55 Results(counter).probability = SumProbabilities/20;
56 end
57 close(h)

```

Listing 14: Function to generate the graphs dependent on different prices.

```

1
2 for i = 1:10
3 plot(Results(i).fractionOfOwners)
4 hold on
5 title('fractionOfOwners dependent on price')
6 xlabel('iterations');
7 ylabel('fraction of owners')
8 end
9
10
11 str = {strcat('Price = ', num2str(Results(1).price))};
12 for i = 2:10
13 % at the end of first loop, z being loop output
14 str = [str , strcat('Price = ', num2str(Results(i).price))]; % after 2nd loop
15 % plot your data
16 legend(str{:},'Location','northwest');
17 end
18
19 limitValue = zeros(1,10);
20 price = zeros(1,10);
21
22 l = 1:10;
23 for i = 1:10
24 limitValue(i) = Results(i).fractionOfOwners(end);
25 price(i) = Results(i).price;
26 end
27 sales = limitValue.*price;
28
29 fig = figure
30 left_color = [0 0 1];
31 right_color = [0 0 0];
32 set(fig,'defaultAxesColorOrder',[left_color; right_color]);
33
34 yyaxis left
35 plot(l,price,l,limitValue)
36 ylabel({'price' ; 'fraction of owners in the end'})

```

```

37 hold on
38 [d,c]=max(sales)
39 plot(l(c),price(c),'ro')
40
41 hold on
42 yyaxis right
43 plot(l,sales)
44 ylabel('Sales')
45 set(gca,'YLim',[0 0.25])
46 [d,c]=max(sales)
47 plot(l(c),sales(c),'rs')
48 legend('price','fraction of owners in the end','price to achieve max. sales','sales','achieved
max. sales','Location','northwest')

```

Listing 15: Function to calculate the probability of a person to update or not dependent on the price.

```

1 function[probability] = ProbabilityMaker(parameters, M, i, j)
2
3 %     method = parameters.ProbabilityCalculationMethod;
4
5     f = getFractionOfNeighboringOwners(M,i,j,parameters);
6     t = parameters.progressInSimulation;
7     z = parameters.price;
8 %calculate the probability wich decreases in time. The product gets less
9 %famous with longer time on the market
10
11 p1 = 20*(t+0.02)*exp(-15*(t+0.02));
12 %calculate the probability of the neighbor interaction. It is just a linear
13 %function starting at 0.2 for no neighbor and 1 if all neighbors are
14 %product owner.
15
16 p2 = 0.5*f;
17
18
19 %Calculate the probability dependent on the price
20
21 % z          ...dimensionless price of the opinion/trend/product
22 % tauz       ...Resistance towards a high price (people with more money
23 %            have a higher tau, so they buy an expensive product much faster)
24     tauz = 1;
25
26     p3 = exp(-z ./ tauz);
27
28
29     p = [p1,p2,p3];
30
31 %     if strcmp(method, 'MultiplicativeProbability')
32 %         probability = prod(p);
33 %     elseif strcmp(method, 'MinumumAdjustedMultiplicativeProbability')
34 %         Probability = prod(p) ./ min(p);
35 %     elseif strcmp(method, 'MaximumAdjustedMultiplicativeProbability')

```

```

36 %         Probability = prod(p) ./ max(p);
37 %     elseif strcmp(method, 'MultiplicativeProbabilityWithNeglection')
38 %         p(p < 1e-4) = [];
39 %         Probability = prod(p);
40 %     end
41 end

```

Listing 16: Function to plot the standard deviations of the simulations.

```

1
2 [fileName,pathName] = uigetfile('*.mat','Select the data to plot')
3 load(fileName)
4
5 for i = 1:20
6 c = plot(Results(i).fractionOfOwners,'b','color',[0,0,1,0.05]);
7 hold on
8 end
9 title(strcat('fraction of owners for different runs, number of seeds = ',num2str(Results(1).
    numberOfSeeds)))
10 xlabel('iterations');
11 ylabel('fraction of owners')
12
13 %average
14 SumProbabilities=0;
15 SumFractionOfOwners=0;
16 for i = 1:20
17 SumProbabilities = SumProbabilities + Results(i).probability;
18 SumFractionOfOwners = SumFractionOfOwners + Results(i).fractionOfOwners;
19 end
20 probability=SumProbabilities/20;
21 fractionOfOwners=SumFractionOfOwners/20;
22 d = plot(fractionOfOwners,'LineWidth',2,'color',[0,0,1])
23 legend(d,'average curve of fraction of owners')
24
25 %Gives a vector back with the values at the end of the simulation. With
26 %this values we have to calculate std deviation etc...
27 fractionsInTheEnd = zeros(1,10);
28 for i = 1:20
29 fractionsInTheEnd(i)=Results(i).fractionOfOwners(end);
30 end
31
32 s = std(fractionsInTheEnd)
33 m = mean(fractionsInTheEnd)
34
35 %All standard deviations and means
36 figure
37 seeds = [1,15,30,45,60,75,90,105]
38 m = [0.147,0.2008,0.3524,0.4744,0.5586,0.6348,0.6999,0.7477];
39 s = [0.063,0.0182,0.0294,0.0309,0.0324,0.027,0.0311,0.0245];
40 errorbar(seeds,m,s);
41 title('evolution of mean and standard deviation for different number of seeds')

```

```
42 xlabel('seeds');  
43 ylabel('fraction of owners in the end')
```
