Final Project Review

January 10, 2021

M Ahsan Al Mahir
ahsan.al.mahir@g.bracu.ac.bd
20216007

Atonu Roy Chowdhury
atonu.roy.chowdhury@g.bracu.ac.bd
20216008

Ekram Wasi Shatil
ekram.wasi.shatil@g.bracu.ac.bd
16301072

1 Abstract

Which system do you want to model?

The system we modeled here is a Monte Carlo Simulation based on Percolation theory. It deals with sales in Children's toy industry. This model tries to answer the question, "How much quality should a product have in order to get x% of the target customers?"

This system is modeled after the Social Percolation Model (Solomon et al. 2000).

What do you want to simulate using this model?

We wish to simulate the relation between the percolation threshold (product quality in this case) and the ratio of the number of customers who buy the product and total target customers.

What questions will this simulation answer about the system?

This simulation model will answer the following:

1. Why do we experience sudden hits and flops in toy industry?
2. How does the preference of parents and children effect a percolating system?
3. What is a better percolating condition for social models? The spanning cluster method or the density method?

Why is this simulation important?

This simulation is important to predict the system's behavior and take decisions accordingly. Using this model with information from previous sales, companies can predict the success rate of a given product.

2 Model Formulation

2.1 Write down the questions you asked to understand the system?

1. What are the differences between the percolating model from the paper 'Social Percolation Models' and our model?
2. How can we take both parents' and children's preferences into consideration?
3. How should we take the weights parents give to their childrens' preferences into account?
4. If we take a weighted average of the preferences, should we set the weight 0.5 or we set thw weights at random?
5. Is "existence of a spanning cluster implies hit" and "no spanning cluster implies flop" a really good metric to decide hit/flop?

2.2 Formulate the abstract model

What are the conditions of the experimental frame of this system?

We consider parents and children to be cells in a square lattice. At first, a random set of these cells are selected to be "agents". These agents have the knowledge of the product, and can decide to buy it.

Every cell has three numbers in $[0, 1]$ assigned to it as the parents' preferences, children's preferences and the weights on childrens' preferences. These numbers are assigned randomly. The initial set of agents is chosen randomly. The change in preference, dp for each children is the same.

Define the system state variables, units and events.

1. The $L \times L$ lattice is stored in a two dimensional array.
2. Parent preferences p_i for each of the child on the lattice.
3. Child preferences c_i for each of the child on the lattice.
4. weight w_i that indicates how much child i's preference is valued.
5. The initial product quality q
6. Number of agents
7. The values dp, dq that the dynamic model uses.

Events:

1. The agents are informed about the product.

2. They buy the product if the product quality is higher than the weighted average of parent preference and child preference.

3. If an agent buys the product, he will inform his neibouring lattices about the product.

4. Then the neighboring lattices become agents. A lattice that has already made a decision never becomes an agent again.

Explain the relationships between these system state variables and explain how the variables change when events occur.

1. If an agent buys the product, that child's expectation rises up. So his preference index increases by dp.

2. Otherwise, that child's expectation goes down. So his preference index decreases by dp.

3. If the ratio (number of agents that buy the product)/(total number of agents) is greater than or equal to the desired ratio, then we will say the product is a hit. Otherwise we wil say the product is flop.

4. If the product is a flop, the quality q needs to be increased. If the product is hit, the quality q needs to be decreased to ensure optimal profit. In each of the cases, q increases or decreases by dq.

Using the above answers define the Mathematical Model for this system.

1. Total number of agents is L^2. An agent's decision is denoted by $d = c \times w + p \times (1 - w)$, where c, p, w are child preference, parent preference, weight respectively.

2. If $q > d$, the agent will buy the product, that agent's neighbor's will be agents, and $c = c + dp$ for that agent.

3. If $q < d$, the agent doesn't buy the product, and $c = c - dp$ for that agent.

4. We shall continue this for all the agents. Then check if the product is hit or flop for that particular q.

5. If the product is hit, $q = q - dq$. If the product is flop $q = q + dq$. Then we shall continue the whole process. Here the 2D arrays denoting parents' preferences, children's preferences and weights - all are generated randomly.

What percentage of your model can you implement? (Which parts of your model can you implement) Justify your expectation.

The model uses very basic concepts of percolation theory. So we think the model is easily implementable.

What percentage of the simulation can be done using your expected model implementation? Justify.

The model implementation is expected to simulate the whole simulation, except validation. Because we do not have access to a reliable data set with which we can compare the simulation result.

The papers you read to formulate the model of your system should be referred within the document.

"Social Percolation Models", writers: Solomon, S., Weisbuch, G., Arcangelis, L., Jan, N., Stauffer, D (2000).

URL: https://www.sciencedirect.com/science/article/abs/pii/S0378437199005439

3 Implementation and Output Analysis

3.1 Implementation of the Model

Which simulation techniques (names of numerical methods, modern techniques) will you need to simulate the system?

We used Monte Carlo Simulation for producing our data. We used the ideas from Percolation theory for social models. We didn't require any numeric analysis methods.

Describe the organization of your program.

Our program uses a system class that holds all the system state variables and methods. We have the following methods:

```
class system: # main class
    variables:
        q        # initial product quality
        r        # we say system is percolating if r*L*L people purchased
        parents  # array for parents preferences
        children # array for children preferences
        weights  # array for weights
        agents   # stores the current agents
        dp, dq   # increment values

    gen_agents()     # generates random agents at the start of a phase
    neighbors()      # helper function to return the neighbors of a lattice cell
    decision()       # returns the decision for a cell
    is_percolating() # checks whether the system is percolating or not
    increment_time() # runs a full simulation of all the lattices
    increment_vals() # changes the values using dp and dp
```

We store the values p_i, c_i, w_i in arrays named `parents, children, weights`. When we increment time, that is when we evaluate the change for a product, we first create a random set of agents using `gen_agents()` method. We then create a `blacklist` array to keep track of all the people who made a decision about this product. We also create a `purchased` array to list those who bought the product.

After the time increment, we change the values of c_i and q following the criterions. Then we run the same simulation again.

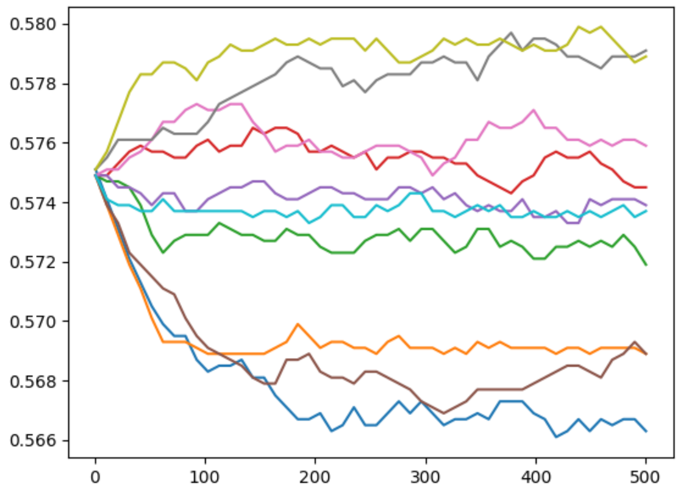
Update your codes

The code can be found at `https://github.com/AnglyPascal/cse474_final_project/`

3.2 Result Analysis

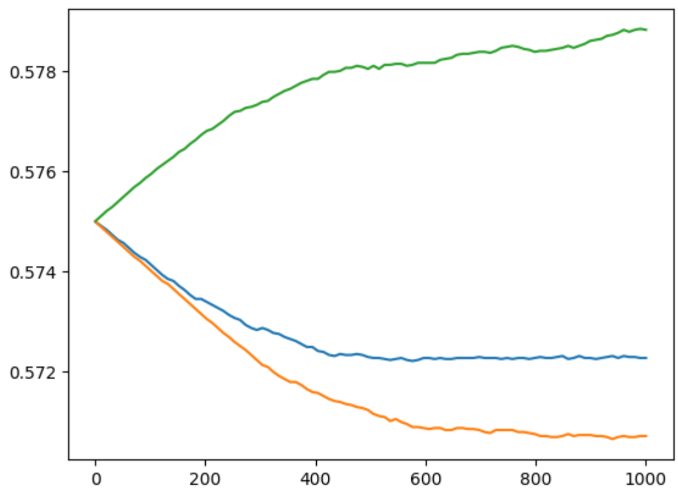
We ran multiple tests for different values of L, mostly from the set $\{50, 100, 150, 200\}$, with different values of $q \in \{.3, .4, .5, .55, .575, .5755, .6, , .7\}$, different values of dq, dp, with a time limit of $100-1000$ steps. Because of the limited resources, we couldn't conduct simulation with higher lattices or for smaller values of dq with longer period of times. But in all of our test cases, the value of q converged to a small interval of length .02.

As for the ratio of $r = .5$, the interval for q for $L = 100$ was $[.566, .580]$. This was true for starting values of $q \in \{.3, .5, .57, .7\}$. The graph below shows how q evoleved during the 500 time increment from a starting value of .575 on a lattice of size 100×100, $dq = 1e-4, dp = 1e-6$.



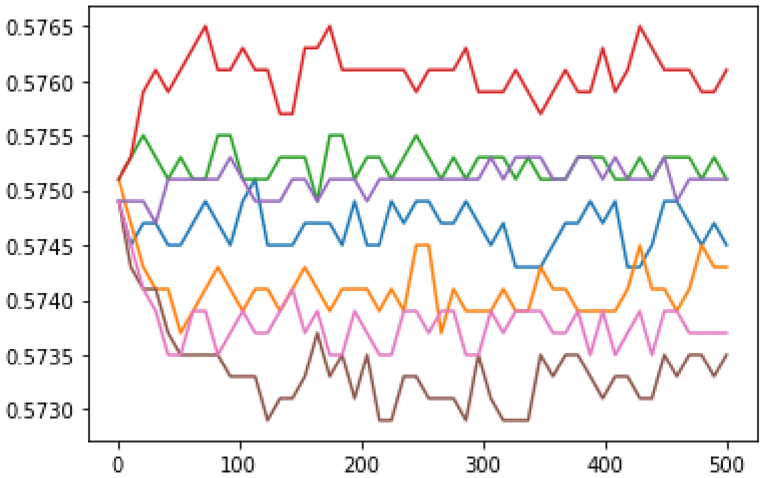
This q vs time graph shows the convergence of q as time increases in 10 lattices of size 100×100 with starting value of $q = .575$.

The following graph plots the same for 200×200 lattice with a total runtime of 1000, with starting value of $q = .575$, $dq = 1e-5, dp = 1e-7$ and ratio= .5.



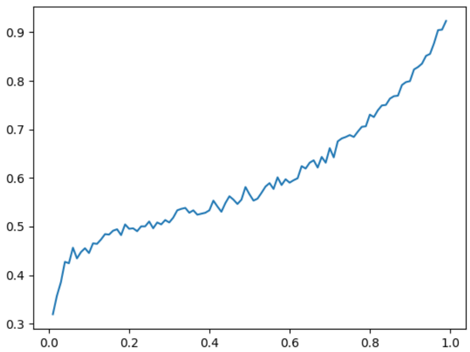
This q vs time graph shows the convergence for 3 random lattices of size 200×200 with values $r = .5, q = .57, dq = 1e-4$

In the following graph we plotted the convergence of 10 lattices of size 400x400. The running time was 500 with initial values $q = .575$. We find an even tighter bound for the final value of q, that is $[.5730, .5765]$

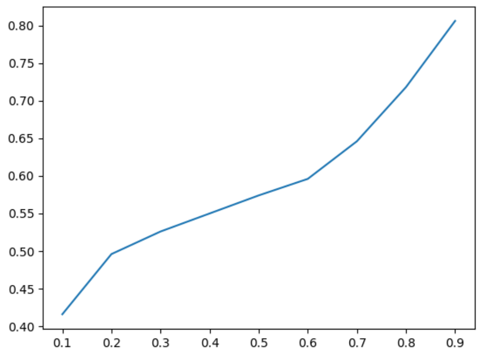


This q vs time graph shows the convergence for 10 random lattices of size 400×400 with values $r = .5, q = .575, dq = 1e-5$

In the following graph, we see how the value of q evolves with our definition of percolation based on density. As the value of r goes from 0 to q, the value of q increases from .4 to 1.



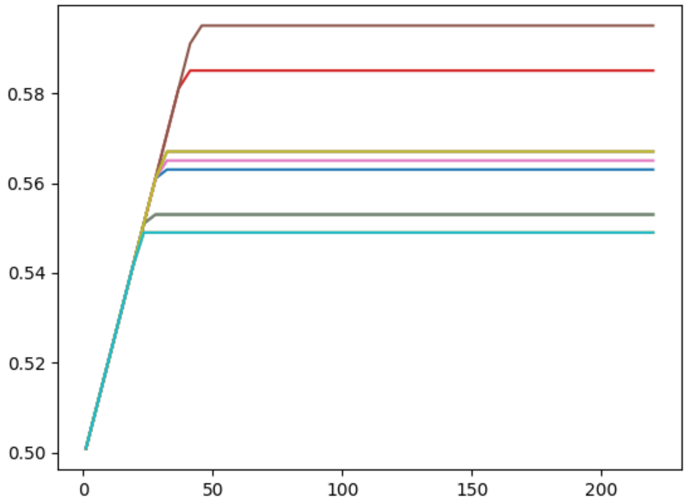
density ratio vs final q graph for lattice size $L = 100$, initial $q = .5$.



density ratio vs final q graph for lattice size $L = 200$, initial $q = .5$.

The surprising fact about this result is that even if r is closer to 0, the final value of q is never below .3, which is counter intuitive. But this happens probably because the system actually updates the values of `parents` which makes the value of q go up. Another reason might be that we weren't able to simulate the system for more than 200 times for each of the lattices because of our infrastructural limitations, so the value of q could've reached 0 if we had given it enough time.

If we used the spanning cluster method in the place of density method, we get something like the following plot:



q vs *time* plot for $L = 100, r = .5, dq = 1e-4$, initial $q = .575$ using spanning cluster check for perconaltion.

As seen in the figure above, using the spanning clusters produces really smooth results. But real life situations aren't like this.

Our explanation for why this happens is: since `parents` is changing based on the purchase they make, their value gets closer to the percolation threshold. However, the spaning clusters doesn't really change that much. There might be a value c such that if $q = c - \varepsilon$, then there is no spanning cluster, but if $q = c + \varepsilon$, then suddenly there is one. And so in the simulation, the value of q oscillates between these two values, while the values of `parents` strenghten this situation.

But by using density analysis, we can randomize this problem further. Because the set of purchased can be very random, and thats what makes whether q increases or decreases, the population has more control over the product value.

3.3 Validation and Verification

Out initial prediction for this model was that the values of q should converge over time, and it should be depended on the ratio value. If our predictions were correct, then this model could be used to predict whether a product would be successful of not based on people's reaction to previous similar products.

Verification:

Our model accurately reflects our conceptual model. During our model development, we found that if we used the spanning cluster method for determining whether the product quality should increase or not, our system behave wildly. Instead if we used the density method, we can predict the behavior of the system. It also reflects the real life scenario more accurately, since industries collect their data statistically.

Since our output also reflects the fact that for some products with quality less than our predicted threshold, then it can be classified as a flop, and that prepresents the real life phenomena.

Validation:

We can't yet validate our model, since we don't have the real life data. But with access to the data from market, our model can be easily validated.

3.4 Self Assessment

Our model could be further improved by taking the following factors into account:

1. Children's social environment, i.e. their schools and neighbors, and the interactinos between them and their friends.
2. Multiple sibling scenarios
3. Children's age factor