Presentation

June 9, 2016

```
In [1]: from SchellingModel2Functions import *
        from Entropy_Estimator import *
        import time

        import requests
        from PIL import Image
        from StringIO import StringIO

        from matplotlib import pyplot as plt
        import matplotlib.image as mpimg

        from IPython.display import display
        from IPython.display import clear_output

        %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib


WARNING: pylab import has clobbered these variables: ['shuffle', 'randint', 'random
`%matplotlib` prevents importing * from pylab and numpy
```

1 Analysis of the Schelling Model by Joshua Parker

1.1 The Schelling Model:

1.1.1 Introduction

- Invented in the 1960's by economist Thomas Schelling to model segregation.
- n x n lattice with three states:
  - 'Red','Blue','Empty'
- Utility function:'Satisfied' if percentage of like-neighbors in Moore neighborhood is above given threshold.

1.1.2 Rules

- On each time step, unsatisfied cells get moved to an empty cell that will make them satisfied, in random order.
- If an unsatisfied cell has nowhere to go, it stays in the same place.
  - Unstable fixed point?
- Reaches equilibrium when no cell can increase its utility.
- Rules vary.

1.1.3 Example

- Percentage of empty cells = 2
- Percentage of blue cells = 49
- Percentage of red cells = 49
- Satisfaction threshold = 50

```
In [2]: example = np.load('example.npy')
```

```
In [9]: for arr in example:
            clear_output()
            grid = array_to_block(arr)
            grid.show()
            time.sleep(2)
```

```
<IPython.core.display.HTML object>
```

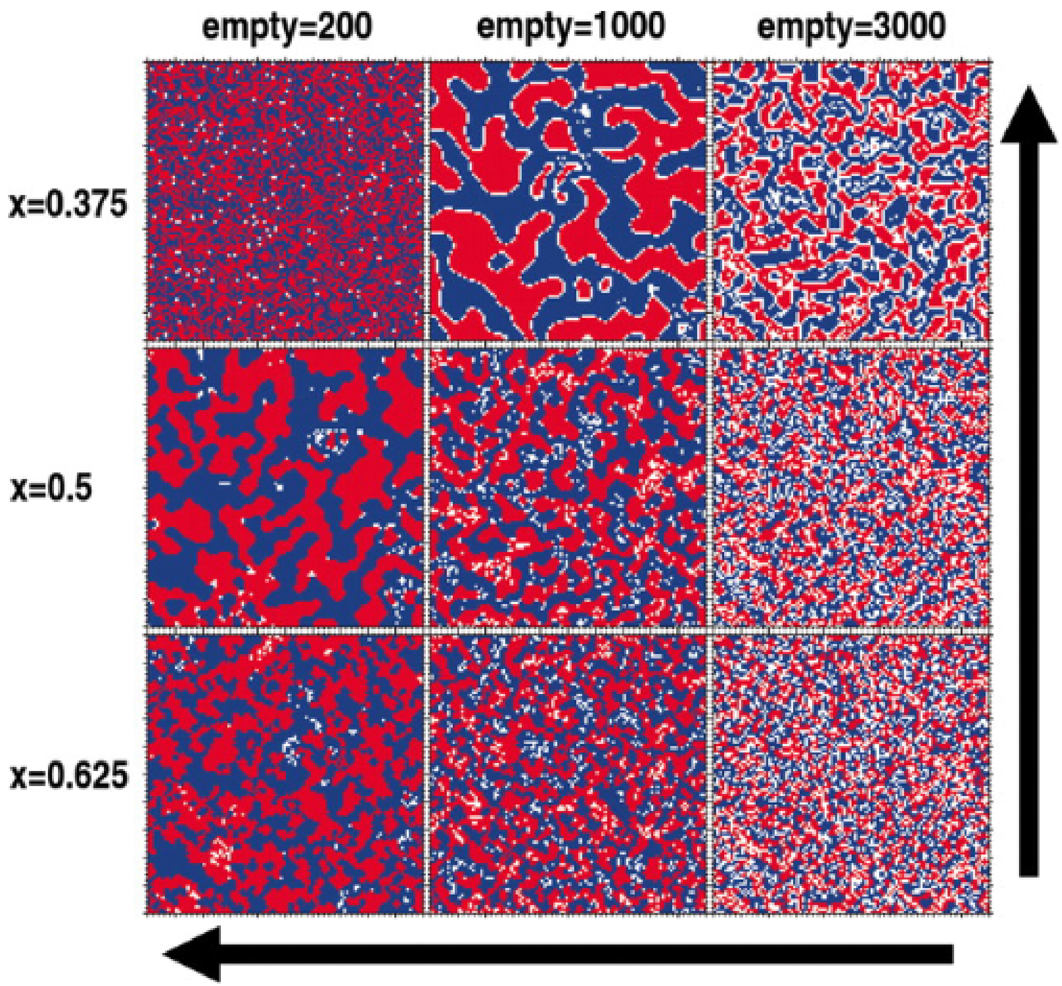
```
In [ ]:
```

1.2 Physical Analogue?

- Each cell gets treated as a physical particle.
- Constant pressure.
- Not a closed system.
- This model describes the formation of a solid.

```
In [4]: response = requests.get("http://www.pnas.org/content/103/51/19261/F3.large.
        pic = Image.open(StringIO(response.content))
```

```
In [5]: fig, ax = plt.subplots(1, 1,figsize=(9,9))
        ax.imshow(pic)
        ax.set_axis_off()
```



1.3 Estimating Entropy Density

1.3.1 Procedure

Entropy is found by estimating frequency of possible M-Block template and feeding the distribution into the following estimator:

$$\hat{H}_2 = \sum_{i=1}^{M} \frac{k_i}{N}\left(\psi(N) - \psi(k_i) + \log 2 + \sum_{j=1}^{k_i-1} \frac{(-1)^j}{j}\right)$$

$$B \leq \frac{M+1}{N}$$

1.3.2 Results

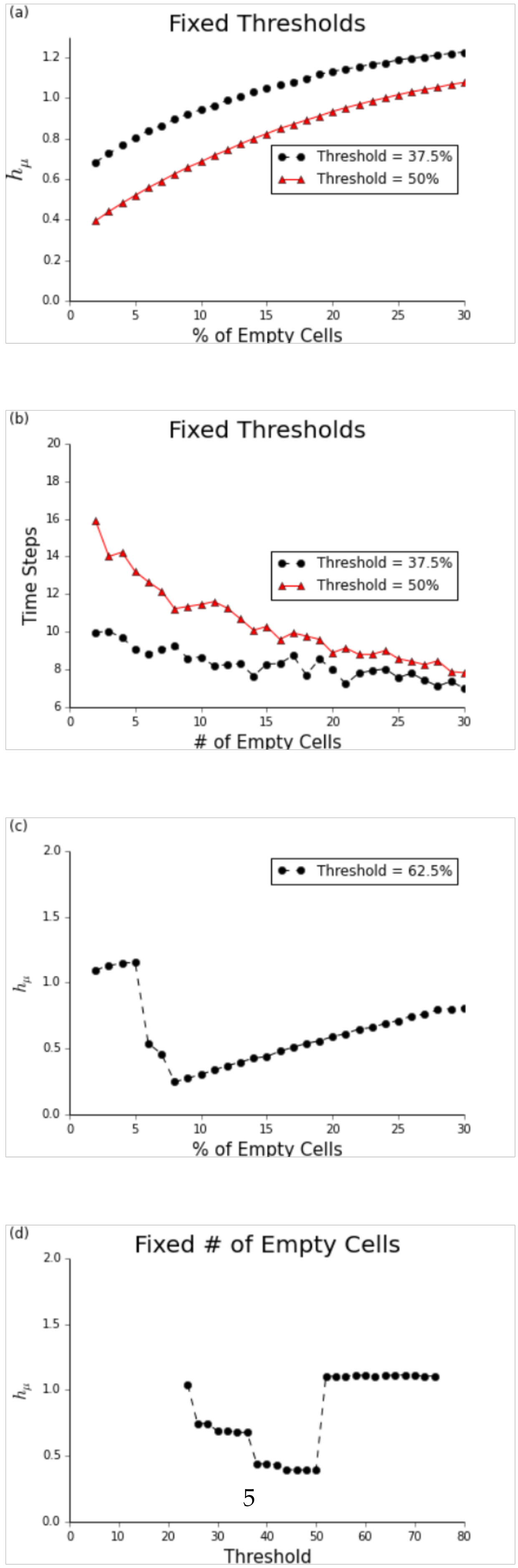
```
In [6]: img0 = mpimg.imread('Fixed_Thresholds.png')
        img1 = mpimg.imread('Fixed_Thresholds2.png')
```

```
img2 = mpimg.imread('Fixed_Thresholds3.png')
img3 = mpimg.imread('Fixed_Thresholds4.png')
```

In [7]:
```
#Make figure
fig, ax = plt.subplots(4, 1,figsize = (30,30))


ax[0].imshow(img0)
ax[1].imshow(img1)
ax[2].imshow(img2)
ax[3].imshow(img3)

ax[0].set_axis_off() # Hide "spines" on first axis, since it is a "picture'
ax[1].set_axis_off()
ax[2].set_axis_off()
ax[3].set_axis_off()
```


(a)
Fixed Thresholds
h_μ
Threshold = 37.5%
Threshold = 50%
% of Empty Cells
(b)
Fixed Thresholds
Time Steps
Threshold = 37.5%
Threshold = 50%
# of Empty Cells
(c)
h_μ
Threshold = 62.5%
% of Empty Cells
(d)
Fixed # of Empty Cells
h_μ
Threshold

1.4 Moving Forward…

- Find excess entropies
- Change boundary conditions
- Change rules
  - Change diffusion rate
  - Make into a liquid