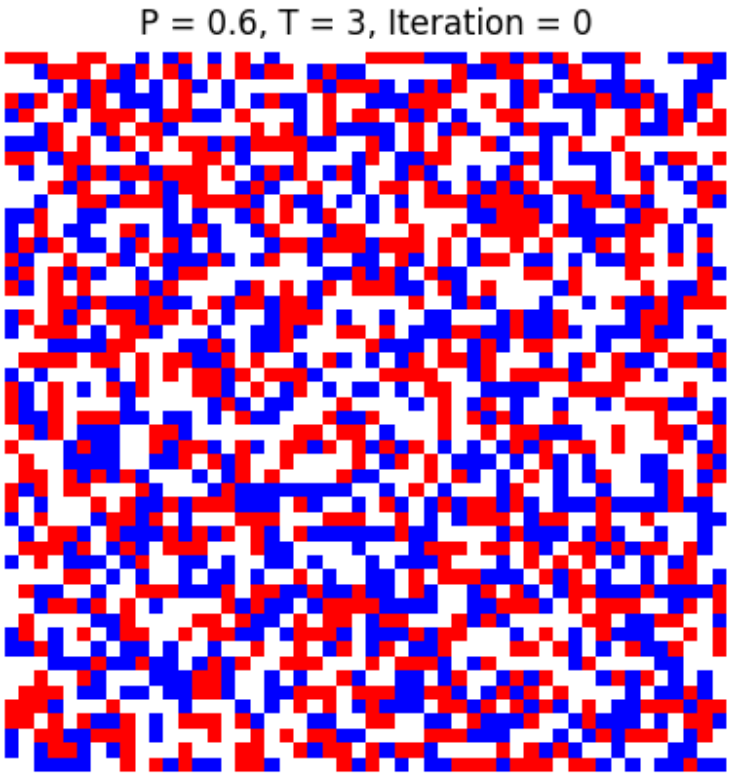
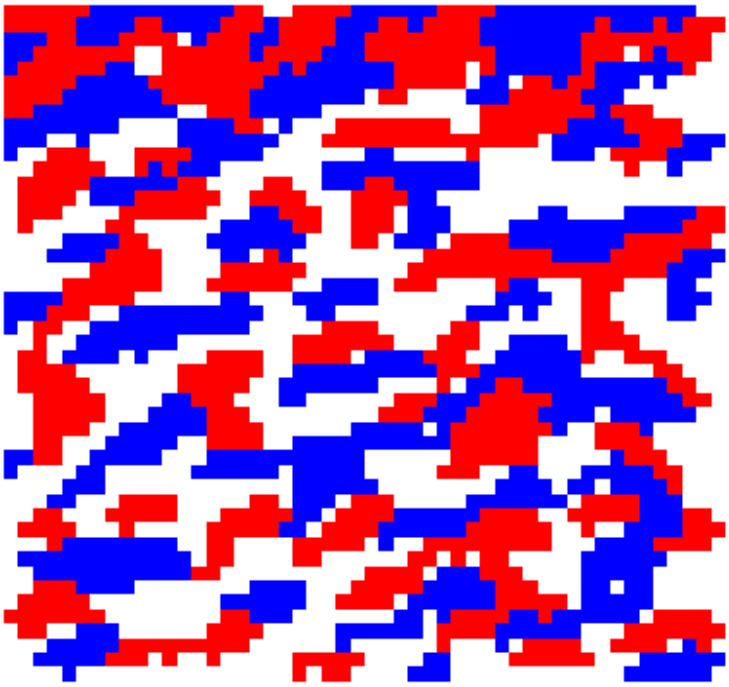
Schelling Model

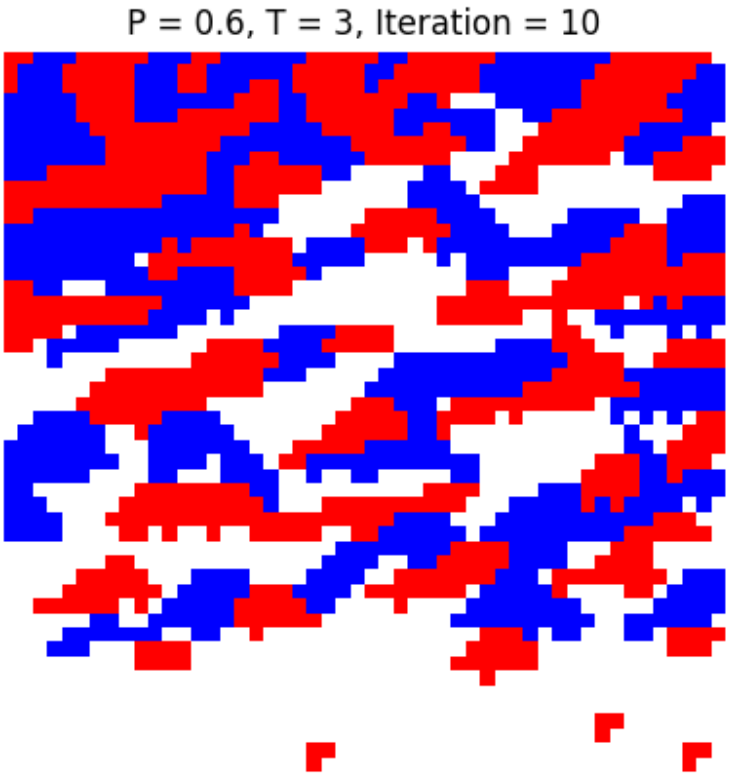
The program I created can run the Schnelling model in a 50 by 50 grid based environment. Using the fraction of the grid that's filled with agents, as well as the satisfaction threshold of each as parameters, users can control aspects of the simulation. Additionally, one can add optional parameters to the dual thresholds test file in order to simulate a fraction of the population having alternate satisfaction thresholds.

For P = 0.6, T = 3:

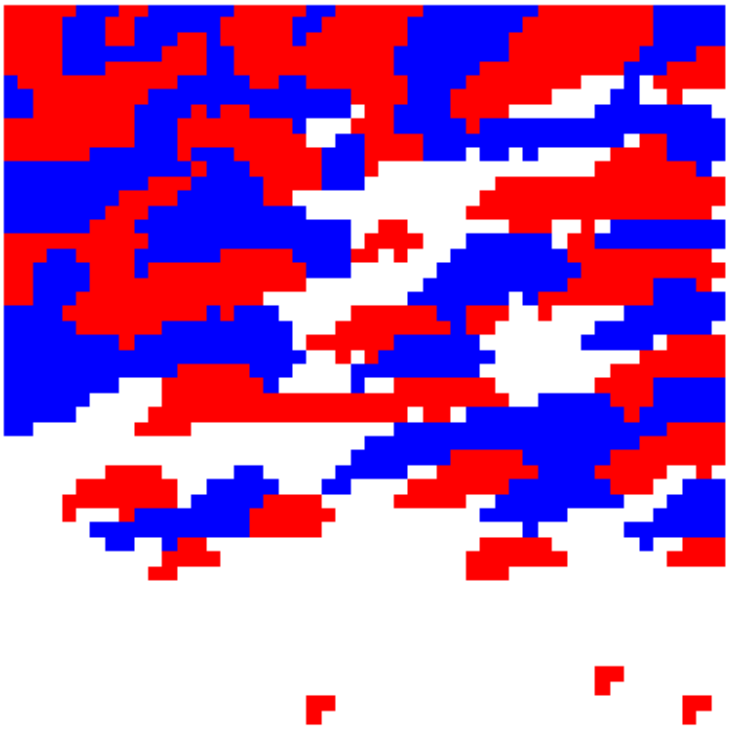

P = 0.6, T = 3, Iteration = 0

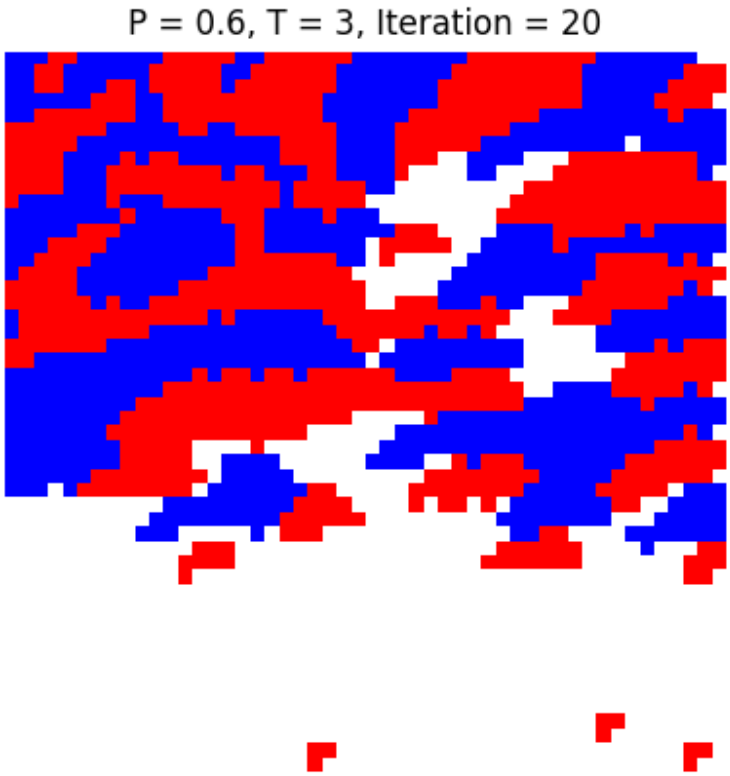
P = 0.6, T = 3, Iteration = 5




P = 0.6, T = 3, Iteration = 10

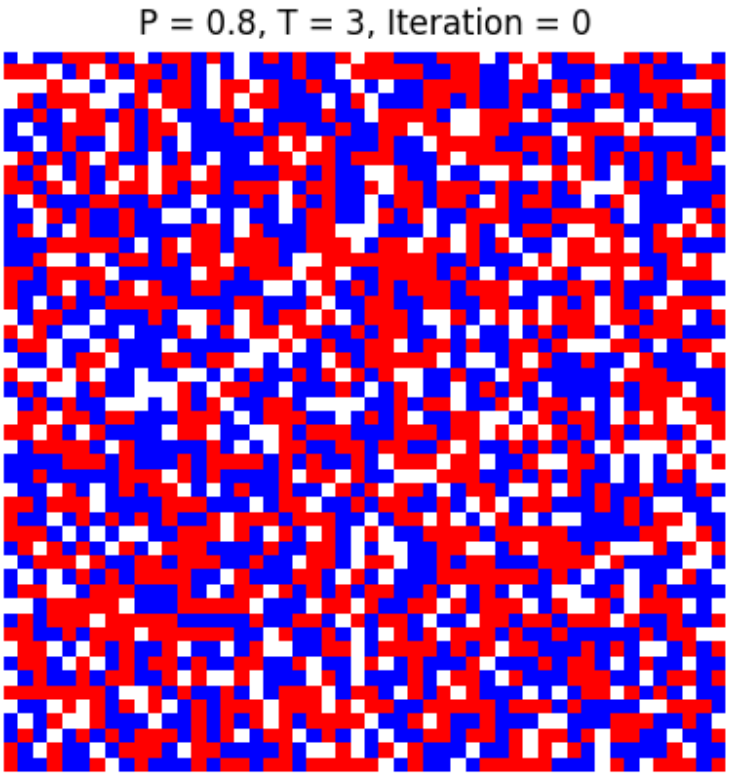
P = 0.6, T = 3, Iteration = 15

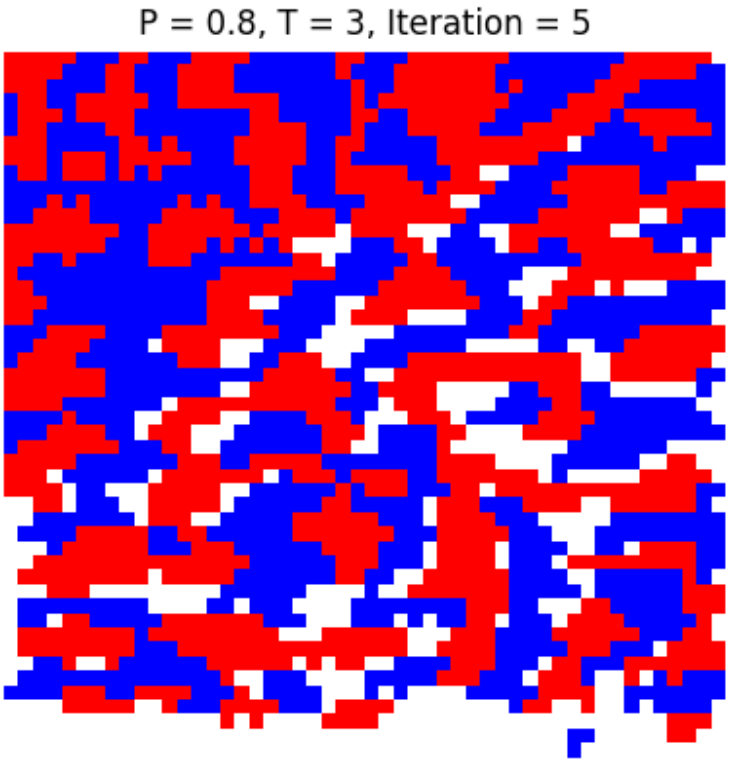


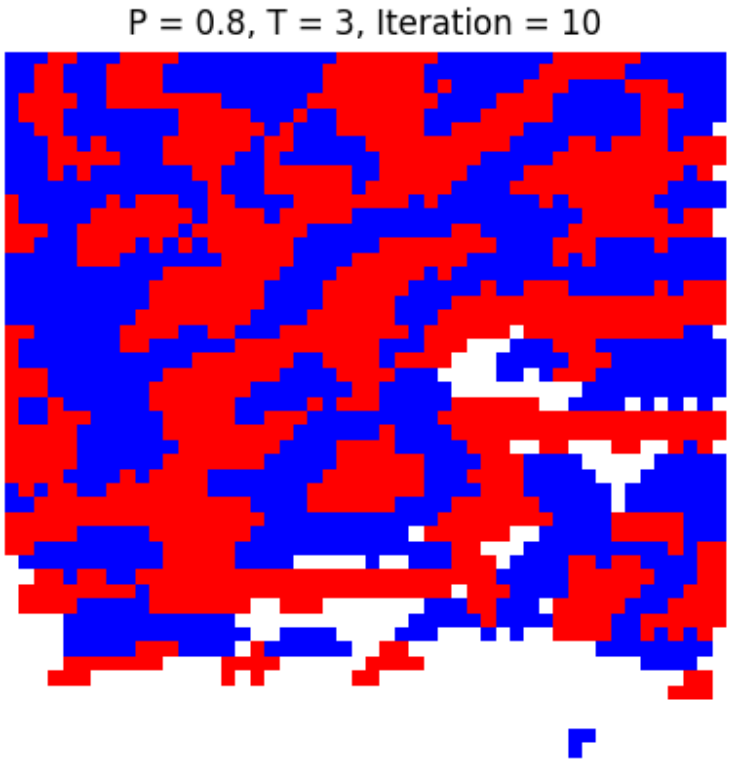


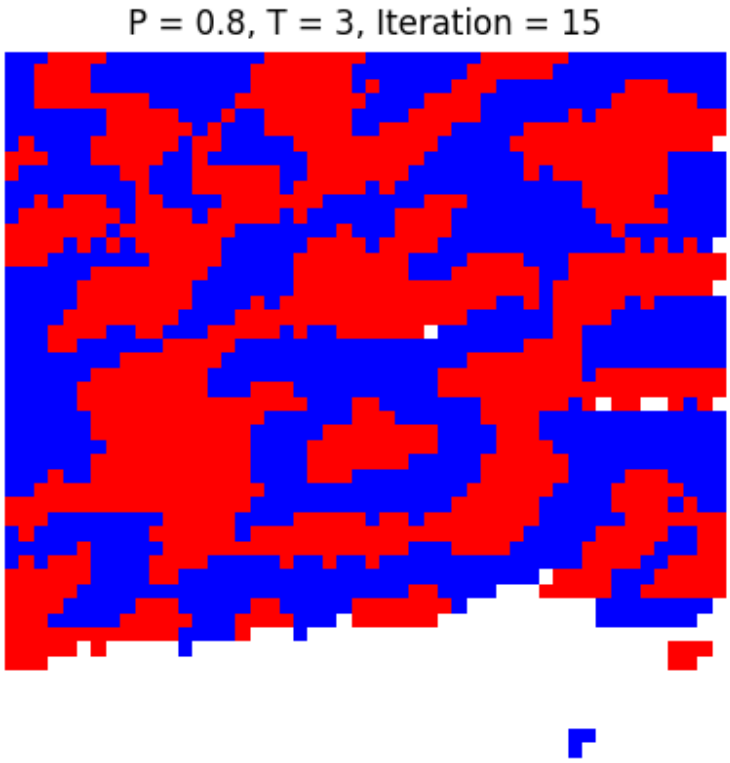
One interesting thing about this model that I quickly noticed, and applies to all other models as well, is that the squares congregate towards the top left a fair bit. This is likely because the model steps from the top left corner from left to right, and then top to bottom. Since the top left is where agents initially cluster at, the program as a whole has the tendency to cluster in the top left as that's where the first few groups are formed. Another interesting thing about this model is that it forms some "islands" that are stuck in the open area outside of the cluster, despite the simulation being mostly complete.

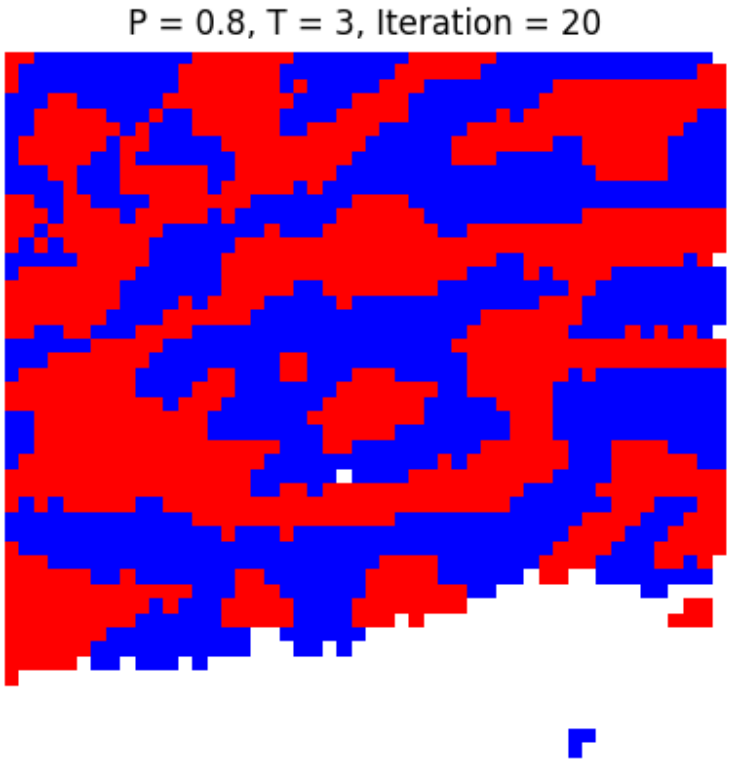
For P = 0.8, T = 3:


P = 0.8, T = 3, Iteration = 0


P = 0.8, T = 3, Iteration = 5

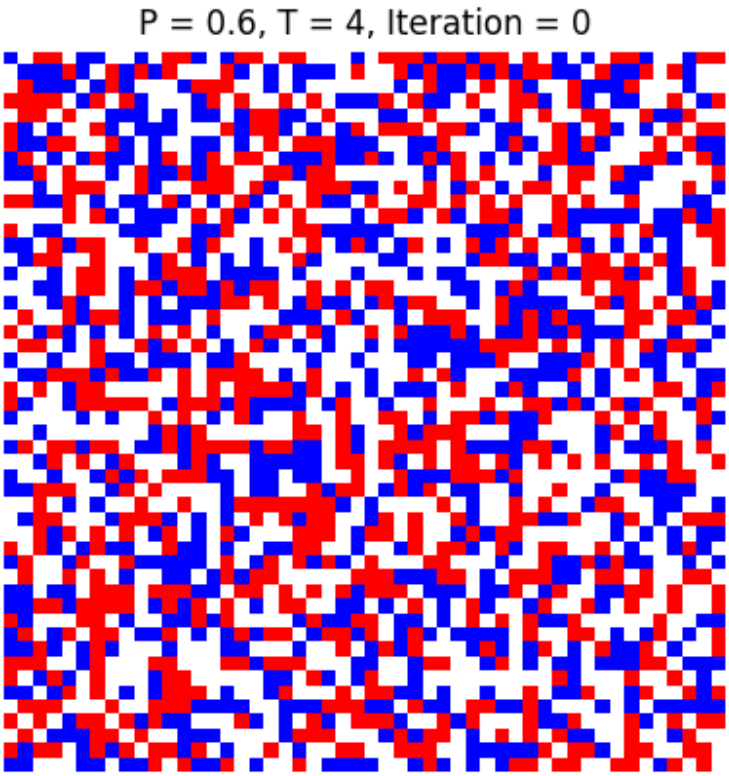

P = 0.8, T = 3, Iteration = 10


P = 0.8, T = 3, Iteration = 15

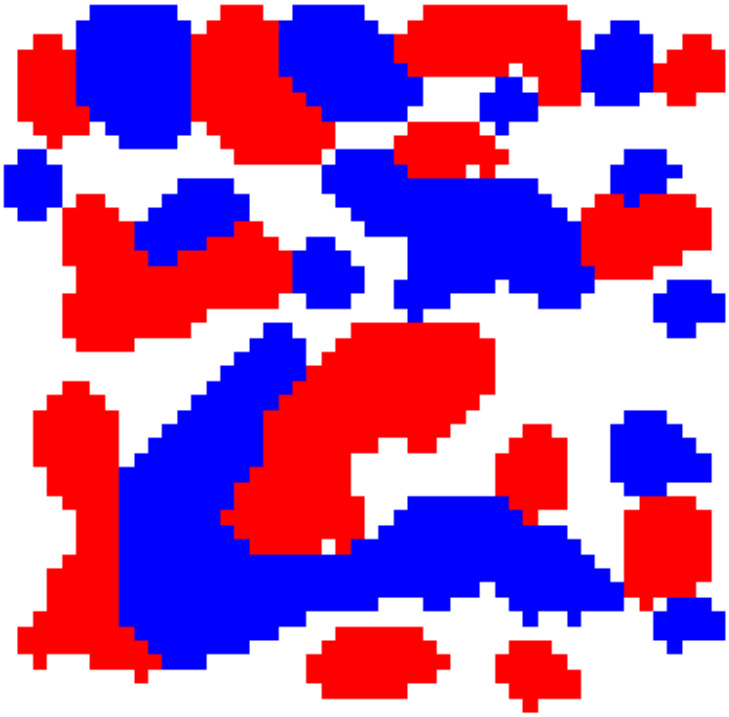


This model seems to reach resolution faster than the previous model. This is likely because the higher population fraction means there are less locations to move around in. If there are less locations to move around in, the model will reach stability faster. Small islands still exist in this model, since while t = 3, strange edge pieces can be satisfied in really small groups.

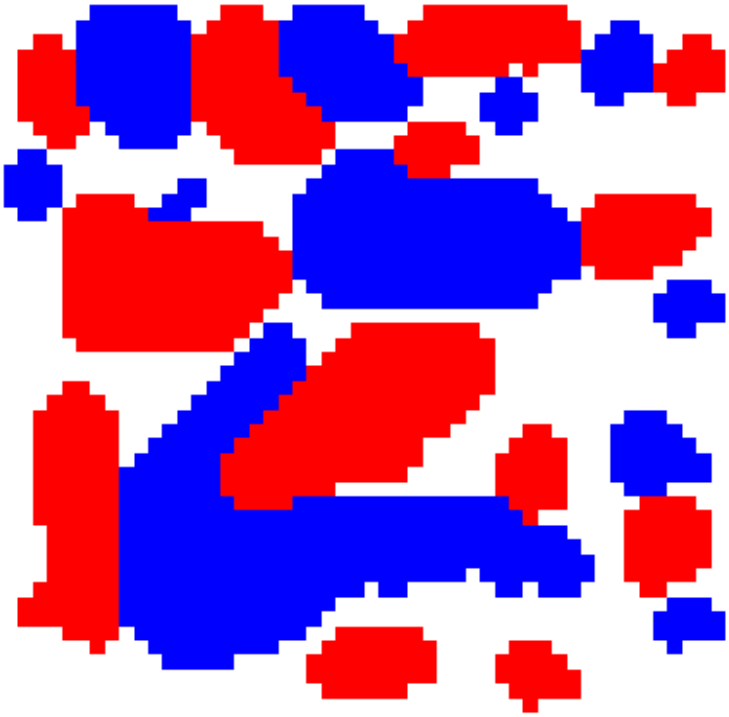
For P = 0.6, T = 4:

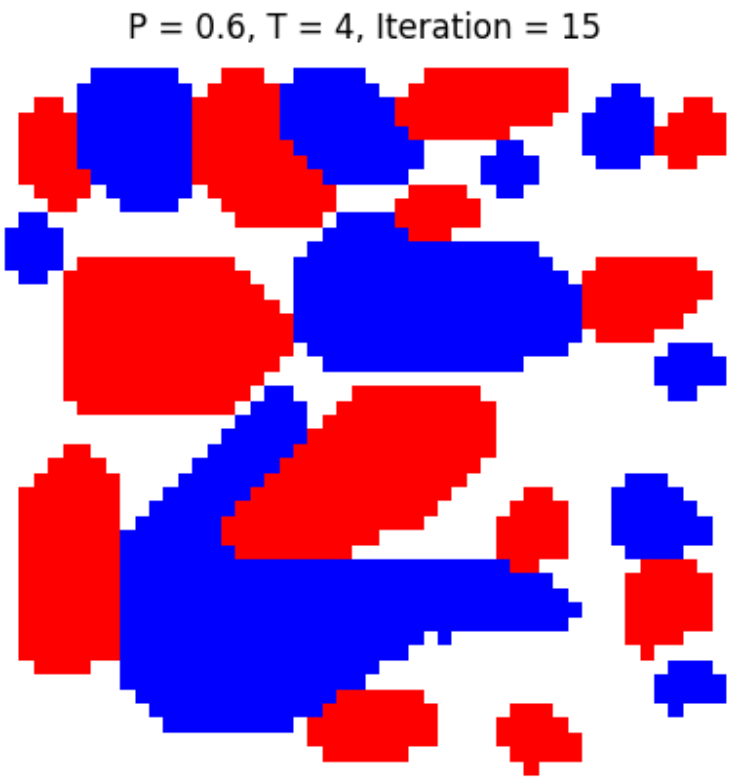

P = 0.6, T = 4, Iteration = 0

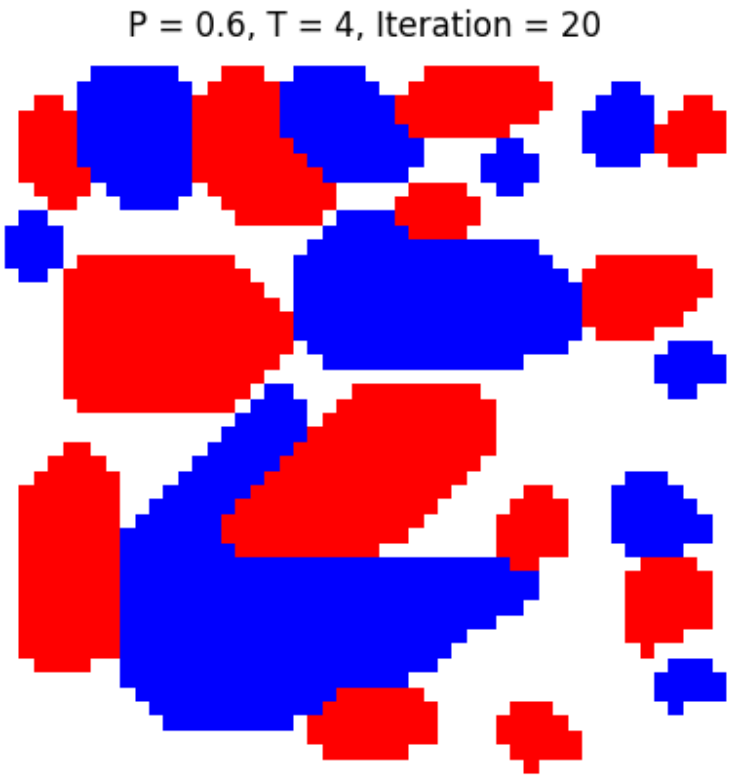
P = 0.6, T = 4, Iteration = 5



P = 0.6, T = 4, Iteration = 10

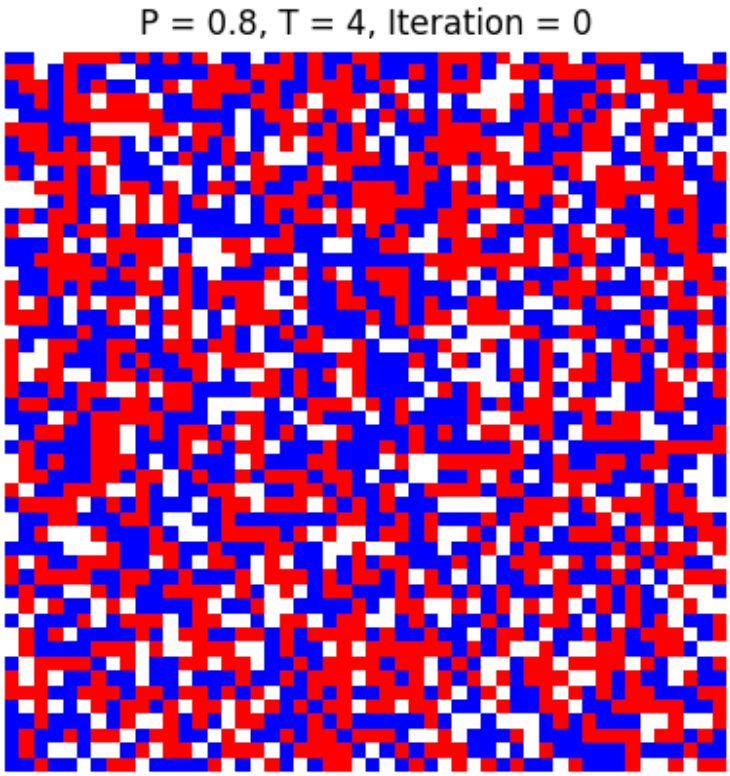



P = 0.6, T = 4, Iteration = 15

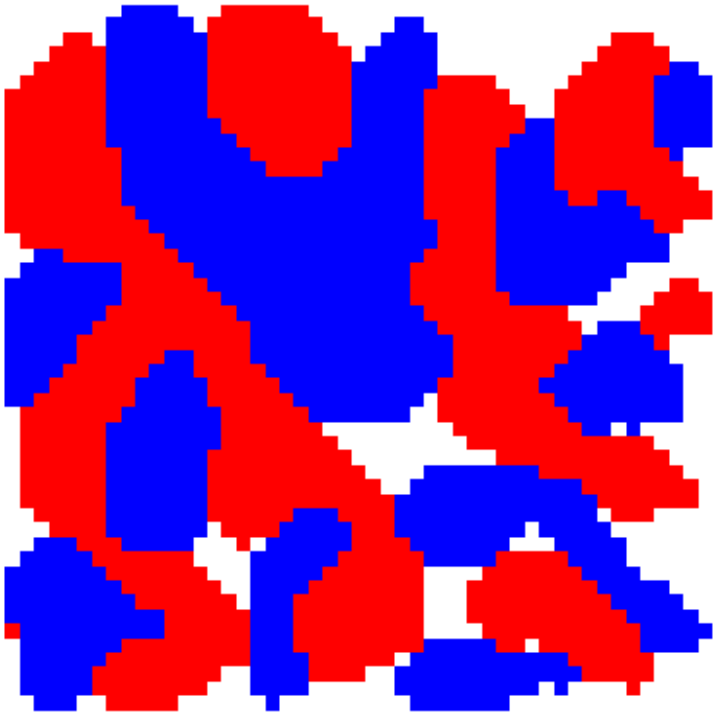


This one was really strange. The low population density led to a lot of smaller blobs forming, and the high satisfaction requirement meant there were not very many places that agents could move. This led to lots of blobs reaching stability because there were no longer any locations to move to.

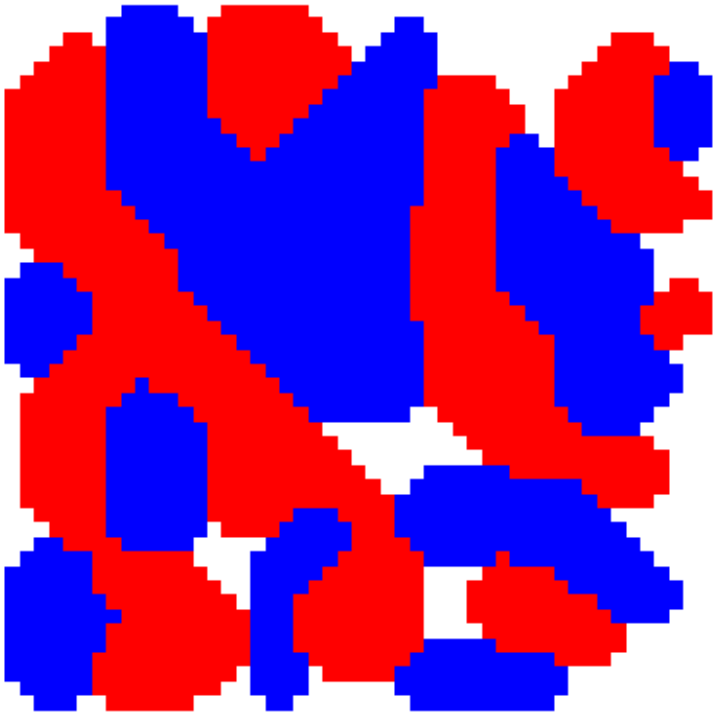
For P = 0.8, T = 4:


P = 0.8, T = 4, Iteration = 0

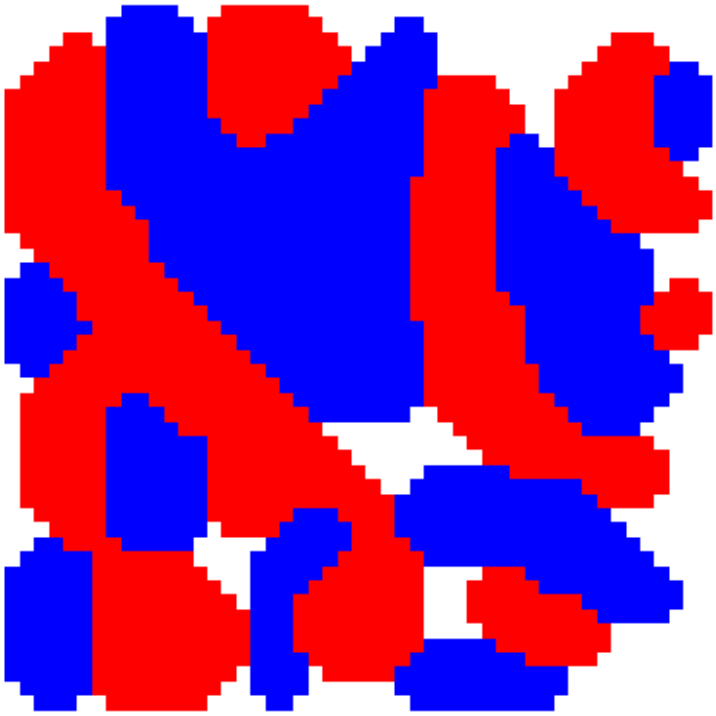
P = 0.8, T = 4, Iteration = 5

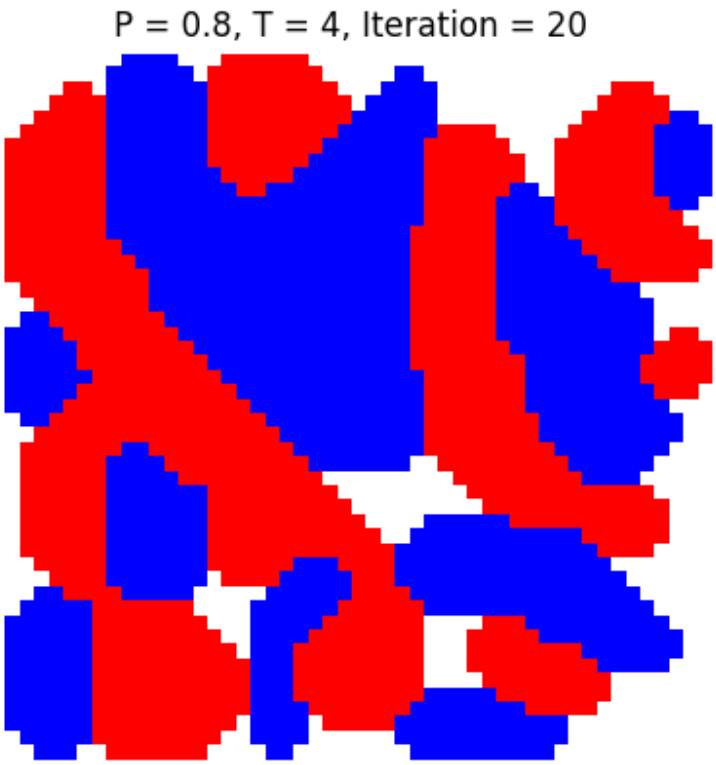


P = 0.8, T = 4, Iteration = 10



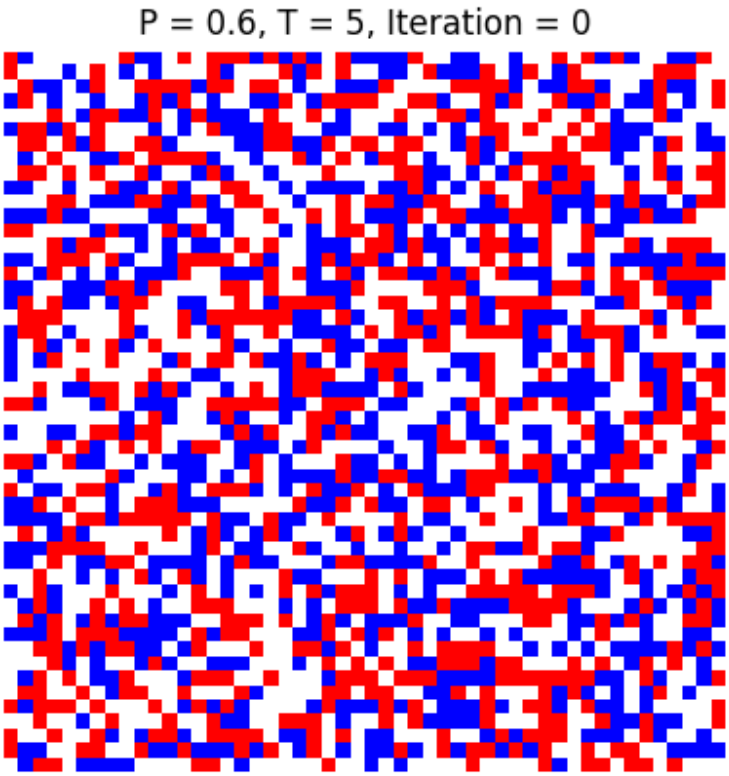
P = 0.8, T = 4, Iteration = 15

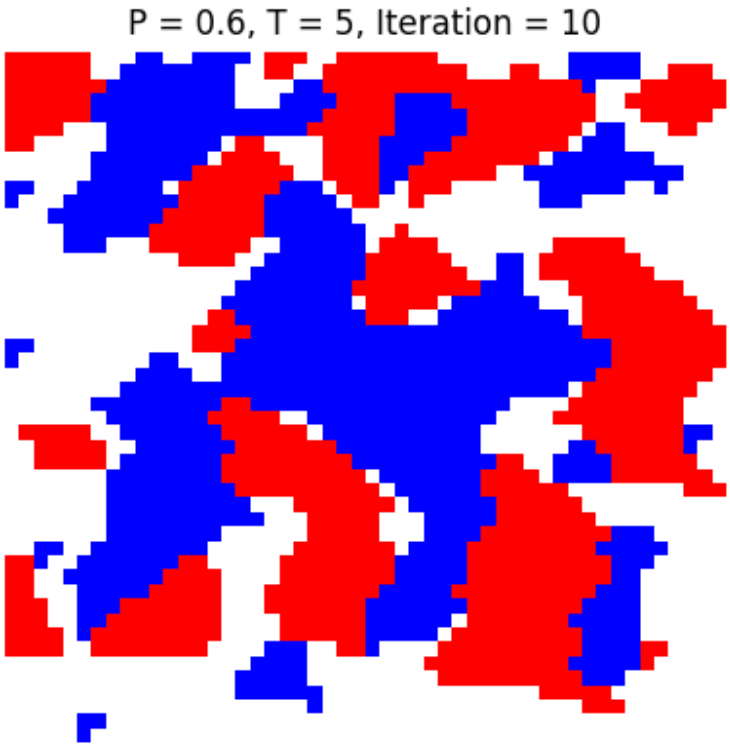


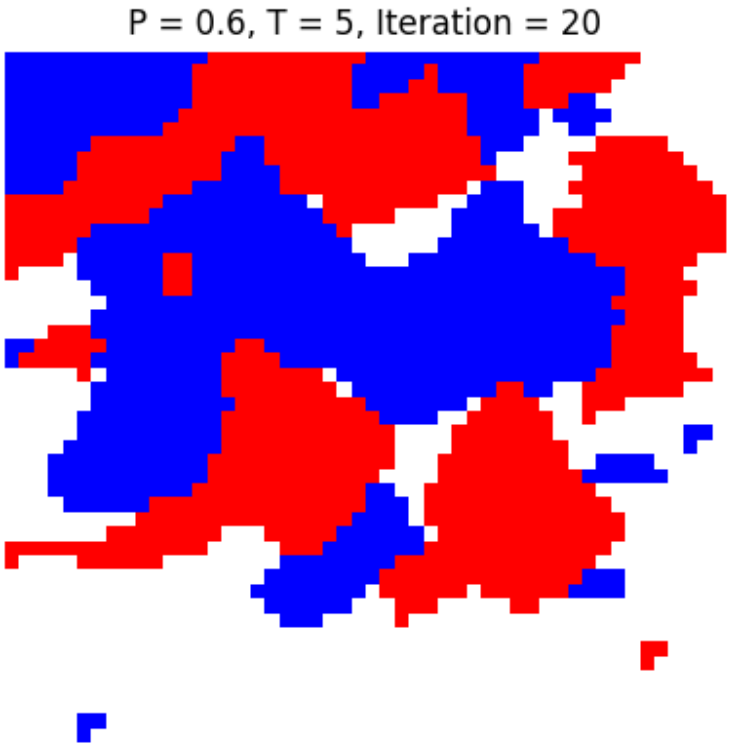


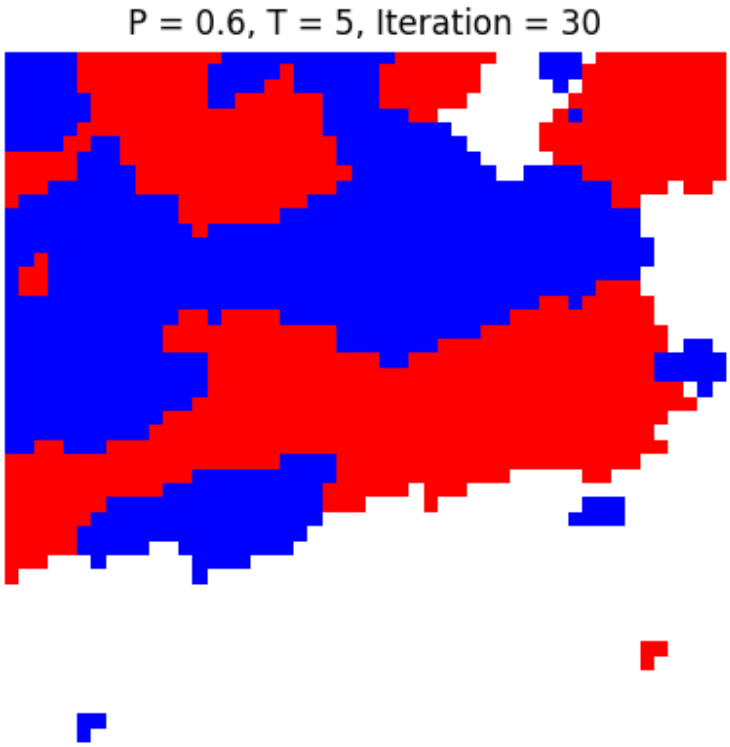
The higher t value continues to result in stability being found quickly, as a result of unavailability of satisfying locations to move to.The higher population density ended this simulation even faster than the 0.6 simulation.

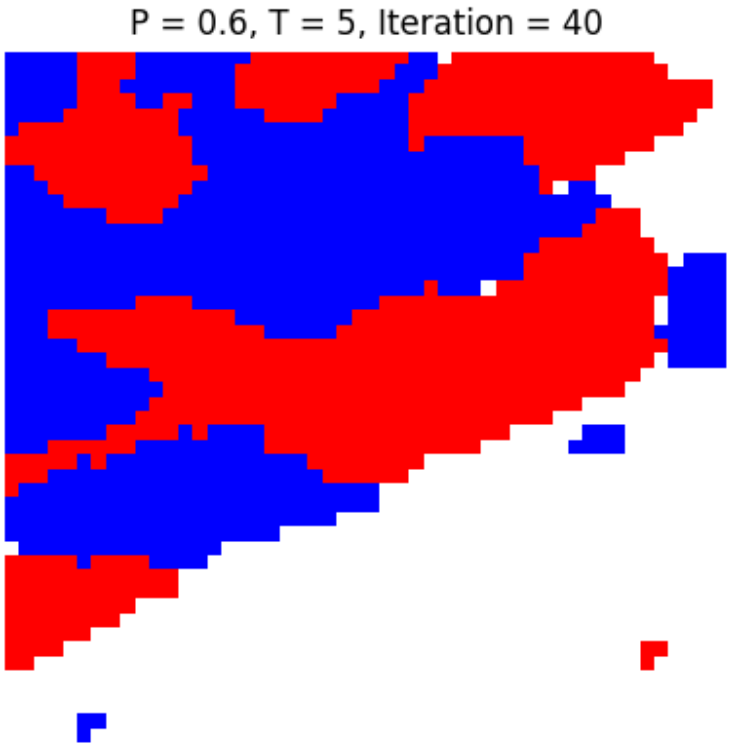
For P = 0.6, 80% of agents have T = 3, rest T = 5 (plots labeled T = 5):


P = 0.6, T = 5, Iteration = 0


P = 0.6, T = 5, Iteration = 10

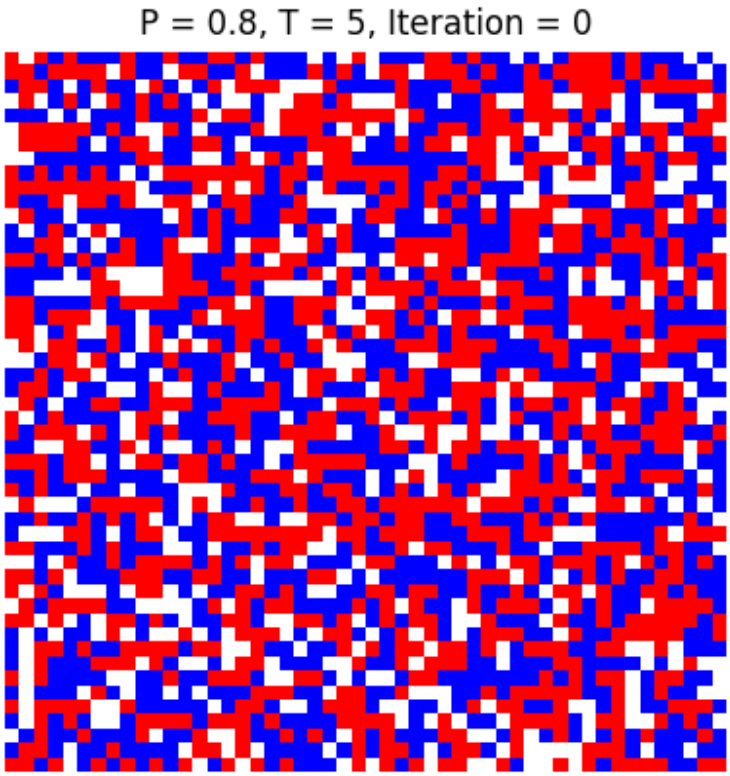

P = 0.6, T = 5, Iteration = 20

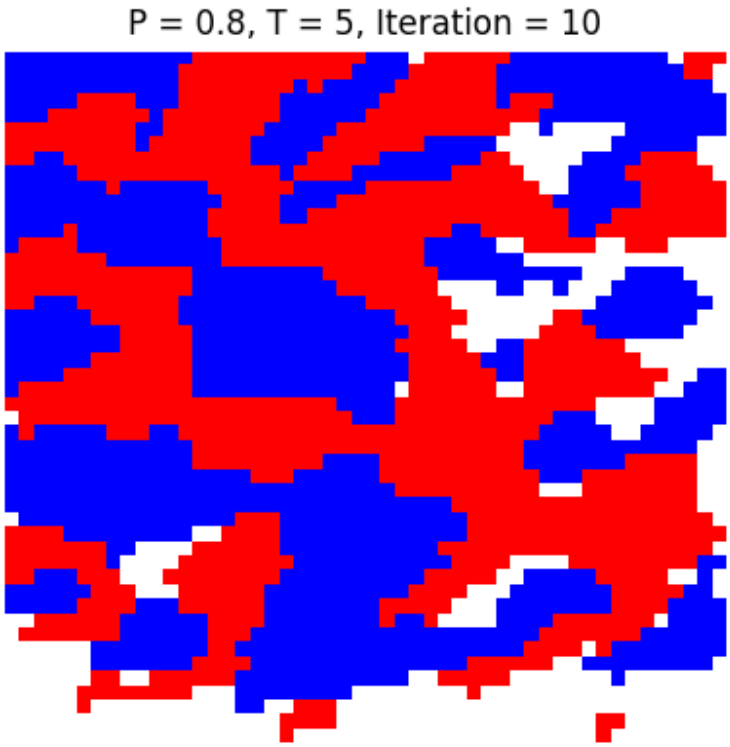

P = 0.6, T = 5, Iteration = 30

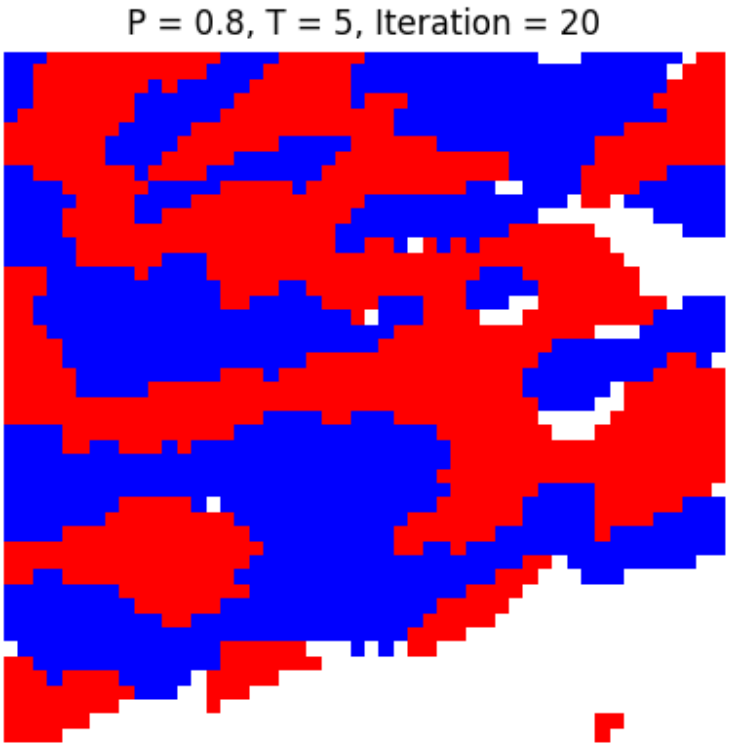


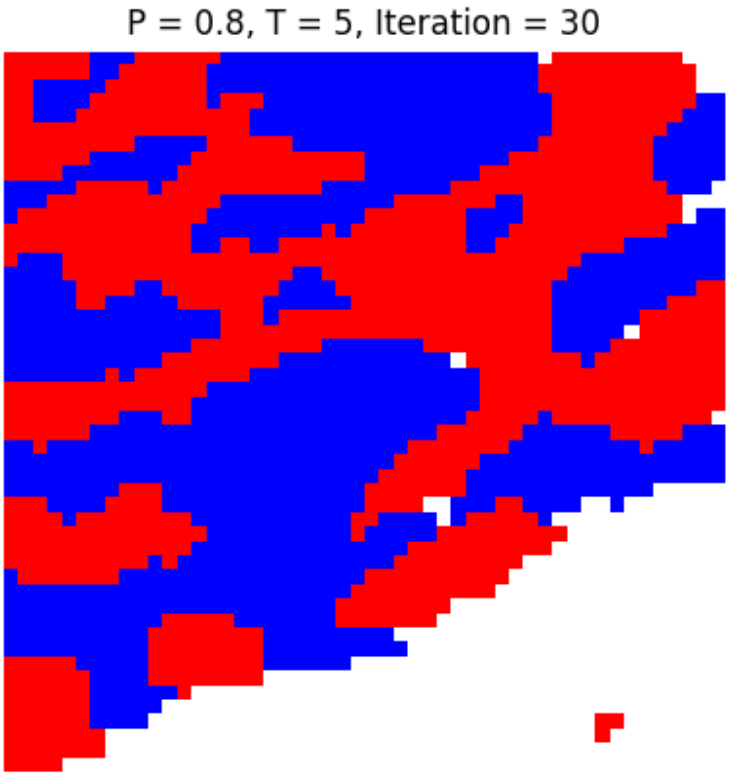
Now that agents with t = 3 are back, some of the smaller islands are forming again. This model takes longer to converge, but creates a much more distinct separation of the groups, likely due to the t = 5 agents making strong centers.

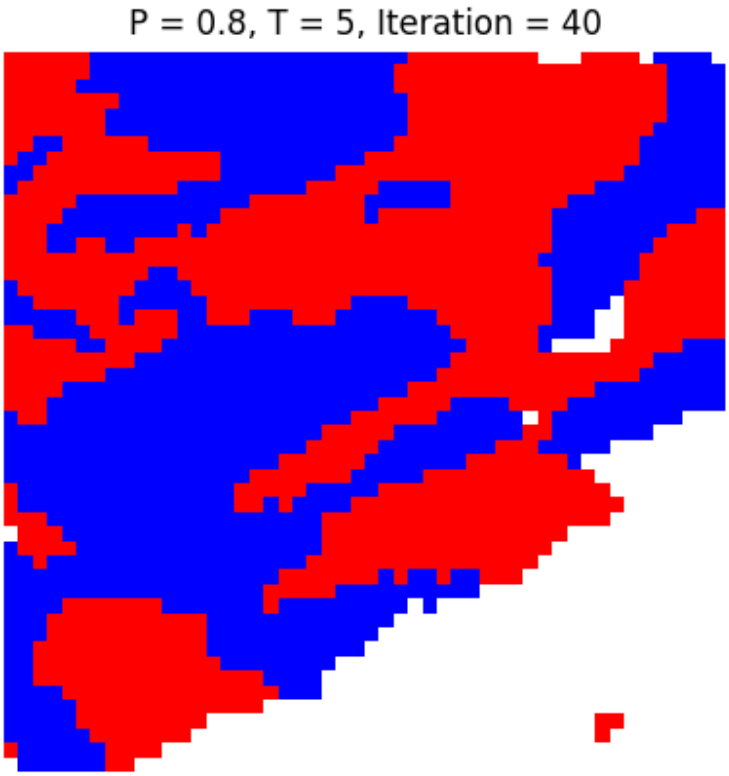
For P = 0.8, 80% of agents have T = 3, rest T = 5 (plots labeled T = 5):


P = 0.8, T = 5, Iteration = 0


P = 0.8, T = 5, Iteration = 10


P = 0.8, T = 5, Iteration = 20


P = 0.8, T = 5, Iteration = 30



This one converged a bit faster than the p = 0.6 simulation with agents of t = 3 and t = 5. The higher population density limiting options is likely why. Additionally, this converged to more distinct groups than any simulation with lower t values, so it is clear that even having a fifth of the agents with t = 5 really does make a big difference.

Sources:

https://stackoverflow.com/questions/7229971/2d-grid-data-visualization-in-python

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.imshow.html