With increasing neighborhood size there is much less possibility for keeping up the initial fragmentation which can only remain due to the spatial structure on the grid. When an agent is allowed to interact with all the other agents on the grid like in Fig. 10d, in the final state there will always be only one cultural region (as long as in the initial state there is not a single agent who shares not a single feature with any other agent on the grid). Therefore the spatial limitation of the interaction between agents is crucial for cultural zones/regions to remain. This also seems realistic when looking at the actual world.

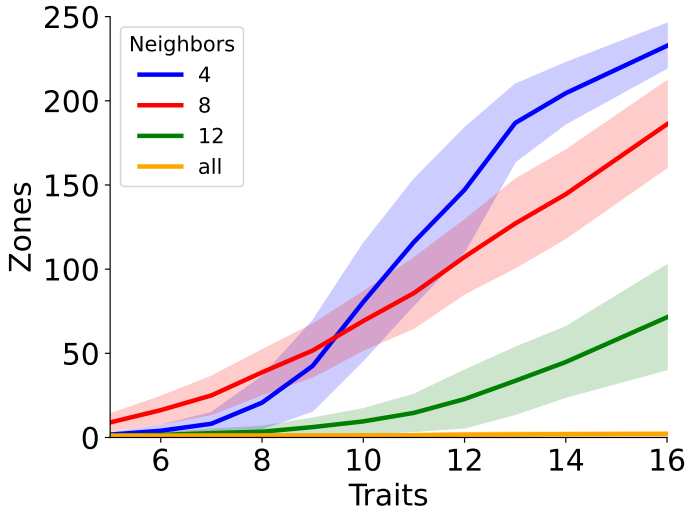

Figure 11. Final amount of zones over range of traits for different types of neighborhood configuration. Agents with 3 features on a 20x20 grid. Results averaged over 100 simulations.

Fig. 11 depicts this impact of the neighborhood size. The previously studied effect of more zones with more possible traits is again observable. With increasing neighborhood size the fragmentation gets lost leading to a smaller amount of zones in the final state.

A last aspect that affects the dynamics is the implementation of periodic boundaries. With this the agents on the border have more possible interaction partners which elevates the chance for having a neighbor with one or more shared features and therefore increasing the interaction probability. So the stable state will generally end up with less zones compared to the case of fixed boundaries but the overall properties, like the increasing number of zones for a higher number of possible traits, will remain the same.

4. Implementation in to a python application

The implementation of the model into python was done using mainly three packages; NumPy for the numerical realization of the model, PyQt for creating the GUI and matplotlib for the visualization of the grid. These three parts will be discussed in the following sections.

4.1. Realization of the model

The easiest part was the implementation of the actual model. Since the agents are on a grid the straightforward way was to save the grid in a matrix where each entry represents an agent. The entry for the agent was then a vector, the cultural vector, with its entries, the features, being one of the possible traits represented by a number, as already demonstrated in section 2.

Using NumPy's random module, a random agent on the grid was selected and given the neighborhood size an appropriate neighbor using the same module was chosen. The rest took place as already described in the prior sections, the similarity was computed and the possible interaction was performed.

4.2. Visualization

The visualization at first seemed like a fairly easy task too. Using matplotlibs matshow function, the grid could be displayed like in Fig.1. The problem with this was that each cultural vector should have a unique