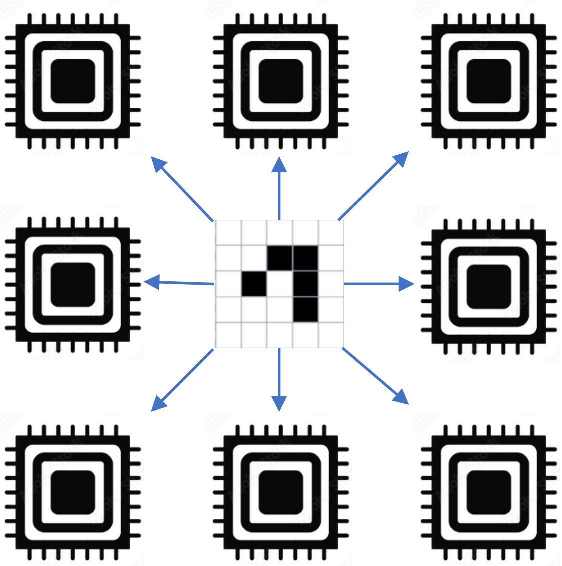
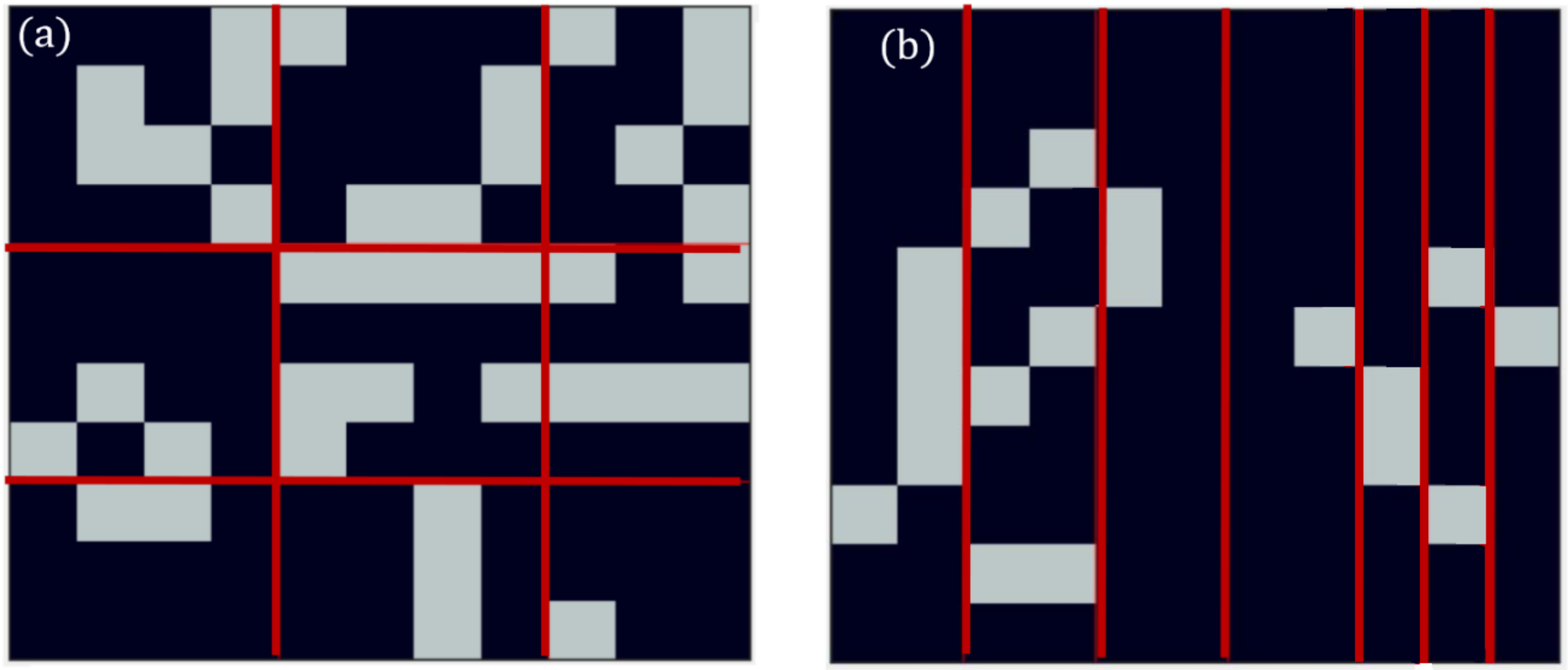
Parallel automata with MPI

Sokratis Anagnostopoulos



Artificial Life course, MATH-642, EPFL

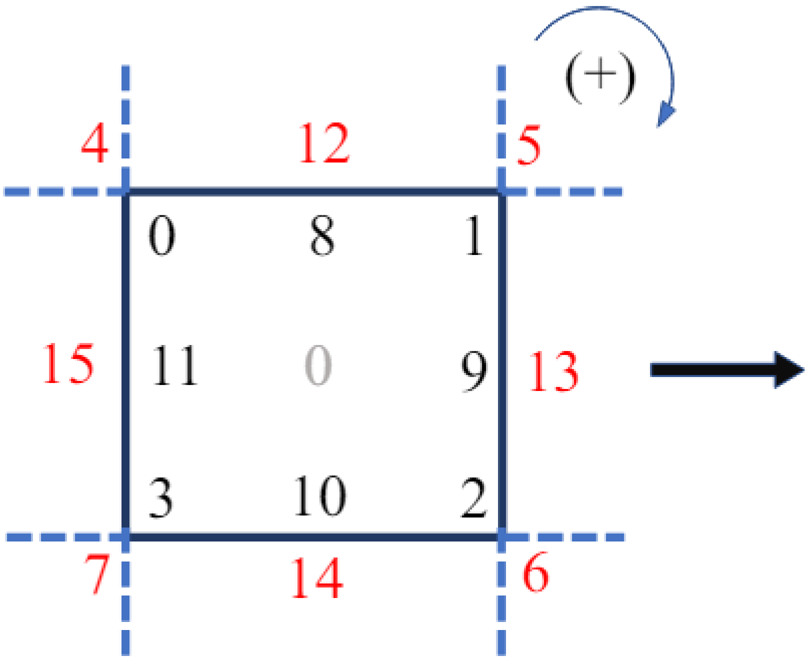
1. Domain decomposition



Snapshots of an 11x11 simulation with 9 cores (a) and 7 cores (b)

2. Core communication

Data types are contiguous in memory. Sending id (black) and receiving id (red) always in clockwise order produce the mapping T

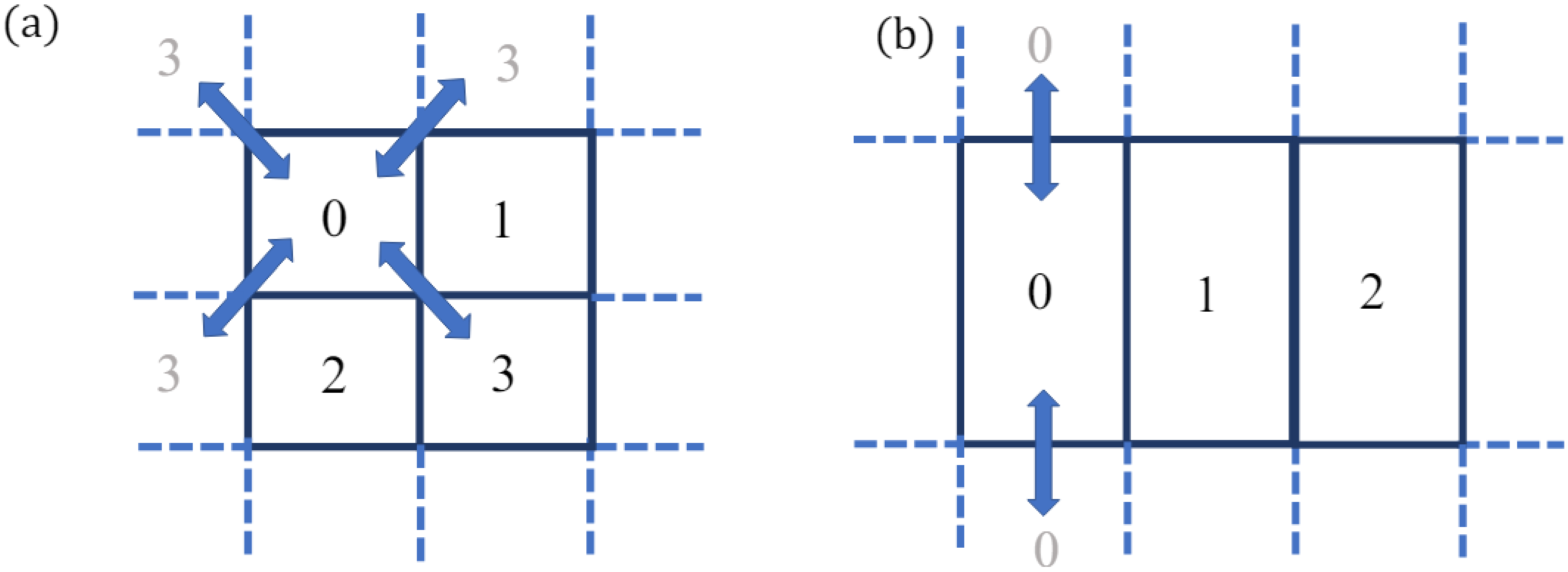


$$T = [0, 8, 1, 11, 0, 9, 3, 10, 2]$$

$$send = type[T[i]], \qquad receive = type[T[i + 4]]$$

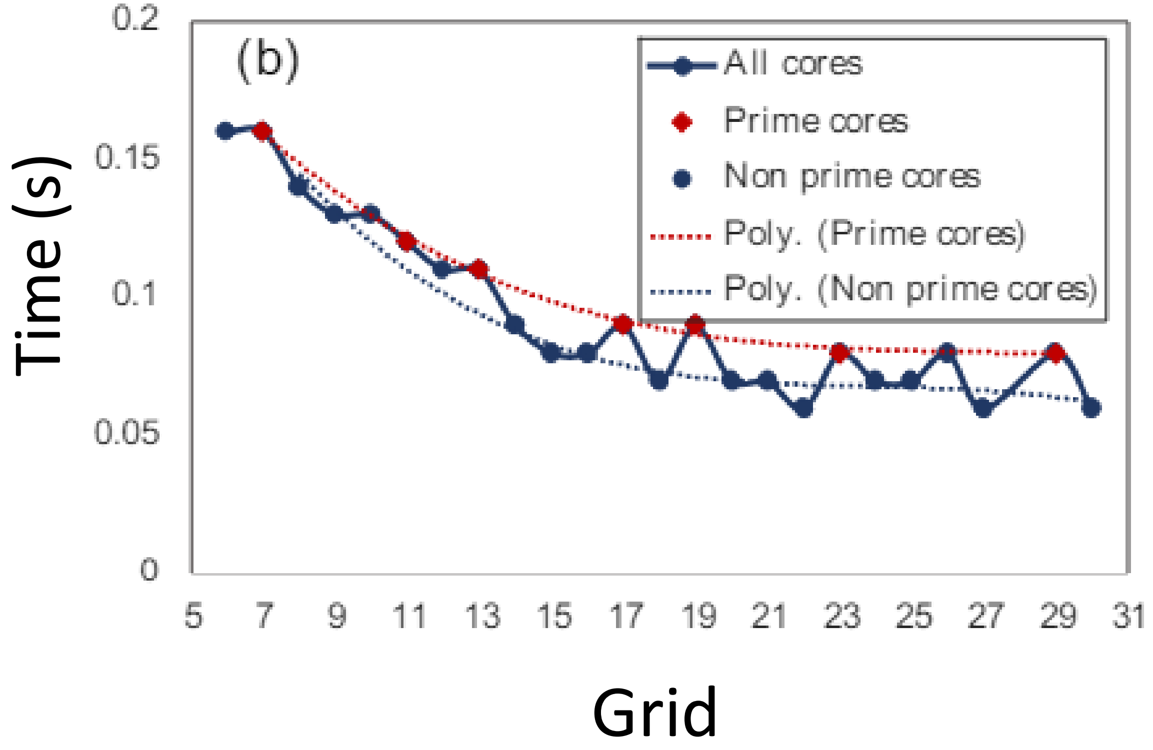
*For the 2D case 16 data types are needed but are efficiently merged in only two groups (vertices and edges)

3. Edge cases



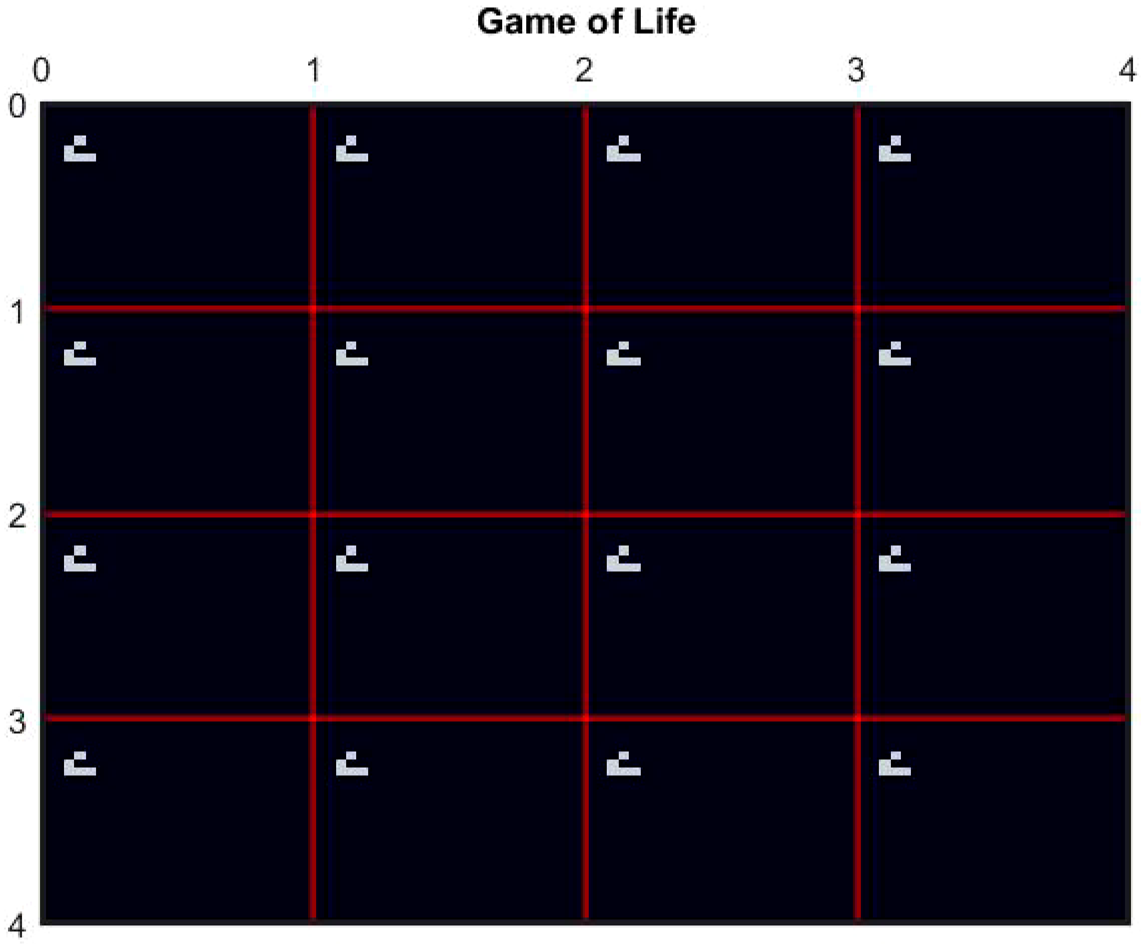
2 by 2 edge case (a) and prime cores (b)

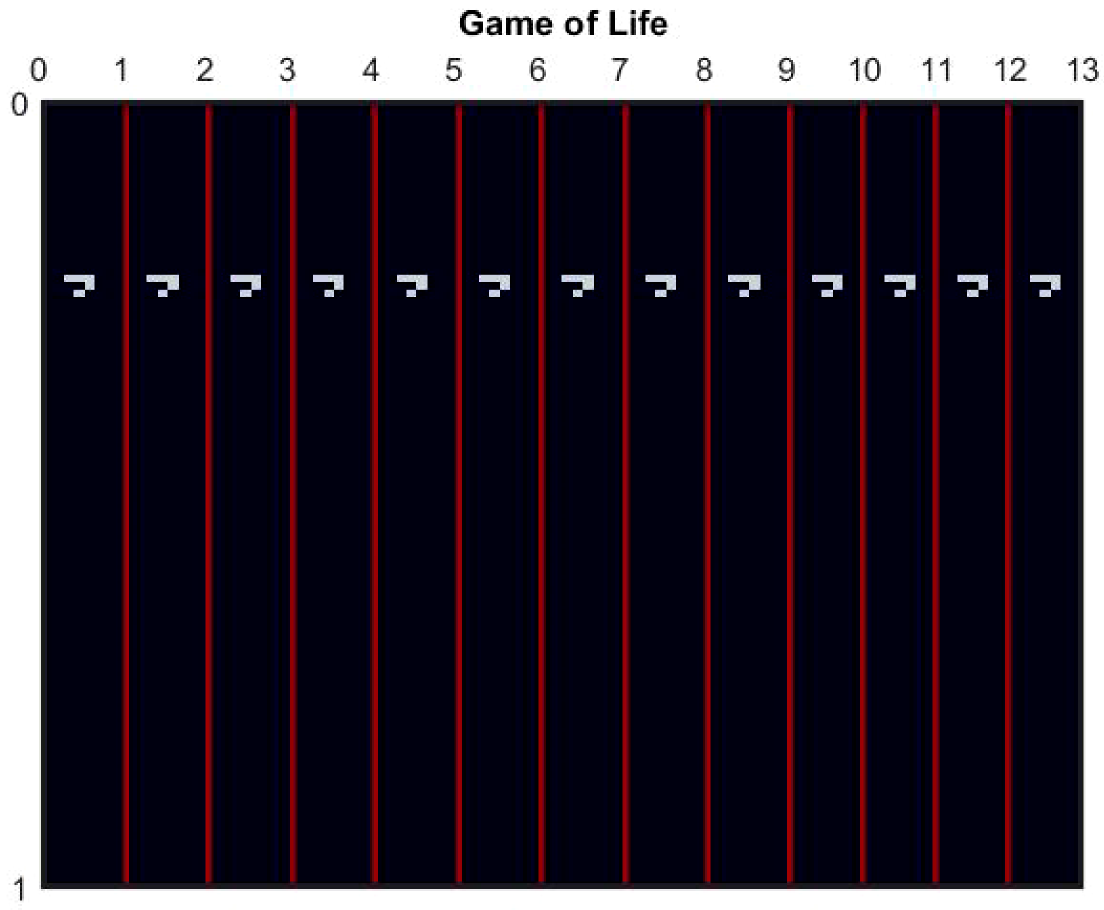
3. Edge cases: prime spikes



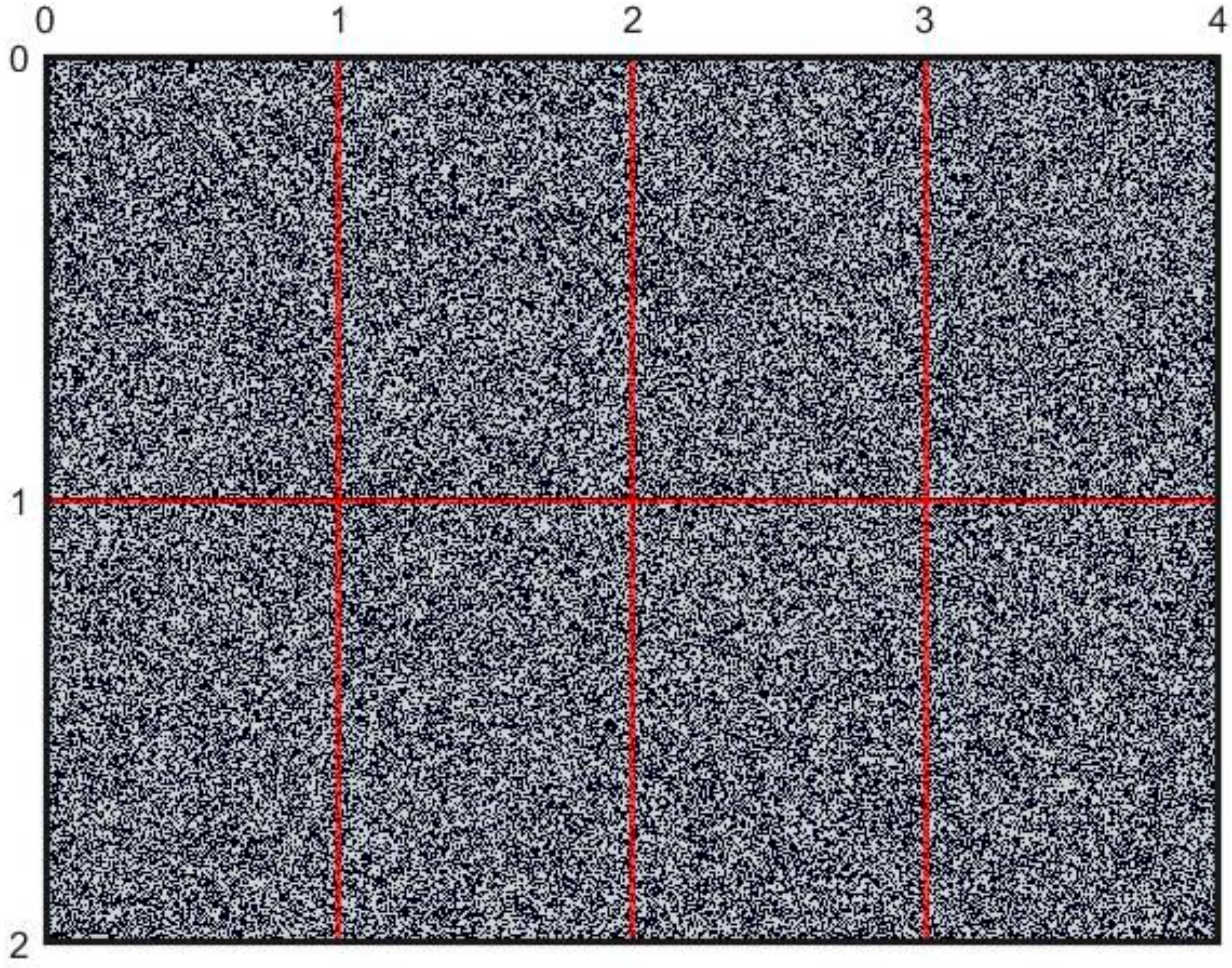
Prime cores are usually less efficient

4. Results: Glider test case (see video folder)

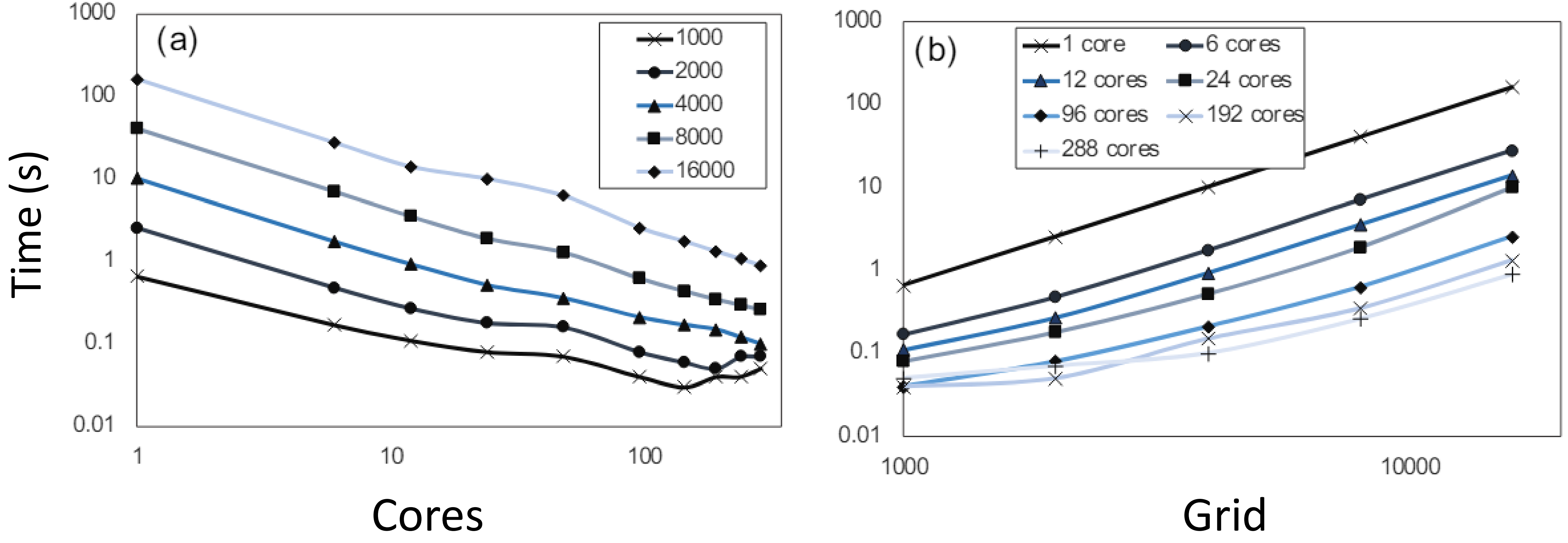




4. Results: Random initialization (see video folder)



4. Results III: time scaling



5. Summary

Parallelizing with MPI produces 1 order of magnitude faster simulations just by adding a few extra cores

Efficient code structure which adapts to other domain decomposition patterns (e.g. arbitrary n-dimensional configurations or triangular grids)

Future work:

- Non-uniform grids (graph connections)
- Try more complex rules
- Adaptive domain decomposition
- GPU acceleration and comparison with CPU

Thank you!