Progress Report: Final Project
CS 442/542, Fall 2023

Kelsey Knowlson
Victoria Lien
Bryce Palmer
Hank Wikle

November 17, 2023

1 Overview

The project code can be found on Lobogit.

2 Python Prototype

We began by creating a simple serial Python prototype. The code represents the entire domain as a boolean vector, with `True` values corresponding to alive cells, and `False` values corresponding to dead cells. The core of the update functionality is a matrix-vector multiplication, in which the state vector is multiplied by a stencil matrix that computes the number of neighbors for each cell. This approach is inspired by 2-dimensional finite differencing methods discussed in Dr. Jacob Schroder's Scientific Computing course [1], and this initial prototype uses a modified version of code provided by Dr. Schroder to generate the stencil matrix.

3 Stencil Matrix Generation

Our next task was simplifying the logic that generates the stencil matrix and translating it into C.

4 Sparse Matrix and Vector Implementation

We implemented three different sparse data structrues in C for use in the project. The first implements a sparse boolean matrix. This data structure implements the COO sparse matrix format, with the modification that no values are stored; if a row and column exist in the structure, its value is assumed to be `True`, otherwise it is assumed to be `False`. The structure begins with a set capacity and uses table doubling [2] to increase capacity as necessary when new entries are added.

The second data structure, which represents the state vector, is a vector of boolean values. This sparse boolean vector was implemented in much the same way as described above, with the exception that instead of storing separate indices for rows and columns, it stores a single set of indices.

Finally, we implemented a sparse vector of `char`, which is intended to be used for the intermediate step of counting neighbors. The decision to use `char` rather than another integral type is motivated by the desire to conserve space in memory, together with the observation that since a cell can have at most 8 neighbors, there is no need to represent values larger than 8. This data structure is implemented similarly to the sparse boolean vector, with an additional member that stores the value of each element.

5 Preliminary CARC Testing

References

[1] J. Schroder, *Lecture 15: Two-dimensional finite differencing*, University Lecture, MATH/CS 471: Introduction to Scientific Computing, University of New Mexico, Oct. 10, 2023.

[2] E. Demaine, S. Devadas, and N. Lynch, *Lecture 5: Amortization*, 6.046J: Design and Analysis of Algorithms, Massachusetts Institute of Technology, 2015. [Online]. Available: `https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/13e2c7d165259712327af0af312a068e_MIT6_046JS15_lec05.pdf` (visited on 11/17/2023).