Optimising Scattered Memory Access of Uniform Spatial Partitioning

R. Chisholm, Dr P. Richmond, Dr S. Maddock
Department of Computer Science, University of Sheffield, UK
{R.Chisholm, P.Richmond, S.Maddock}@sheffield.ac.uk

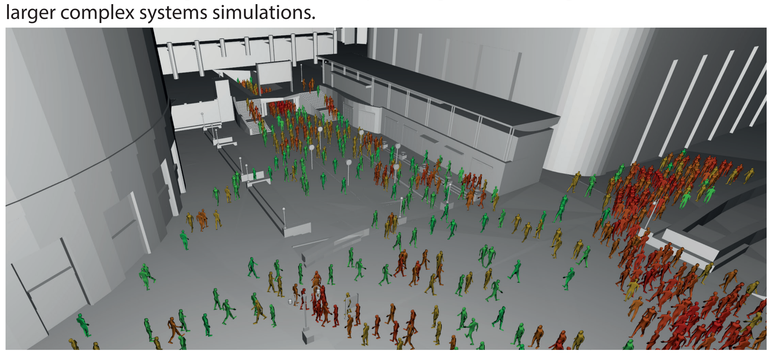
The University Of Sheffield.

Motivation

Many complex systems contain mobile spatial actors that are influenced by their neighbours whether they are particles, people, vehicles or planets. The performance of the underlying data structures used to manage this dynamic spatial data often has a larger impact than that of the model's specific compute requirements.

There are few data-structures capable of handing dynamic spatial data on GPUs. This has led to reliance on uniform spatial partitioning to provide neighbourhood searches. This technique utilises a static data structure which must be reconstructed when any data moves, irrespective of the degree to which the data has changed. This technique is also central to many k-nearest neighbours implementations.

Our research focusing on techniques for optimising GPU near neighbours to facilitate larger complex systems simulations.



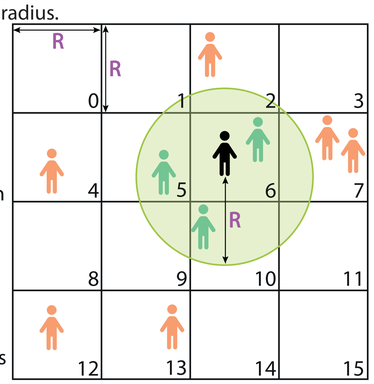
Uniform Spatial Partitioning

The use of spatial subdivision into a uniform grid of bins is a common technique used for surveying near neighbours within a fixed radius.

Uniform Spatial Partitioning

1. Decide the interaction radius: R
2. Partition the environment:
   - Uniform cells of height/width R
   - Clamp agents to environment bounds
3. Search the Moore neighbourhood:
   - Ignoring agents outside of the Euclidean distance R

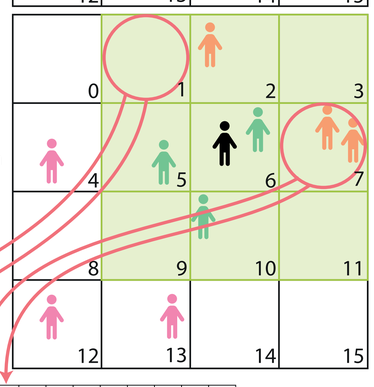
Figures: Top-Right (1-2), Centre-Right (3)



Implementation: Construction

1. Sort the agent data according to their cell indexes into an array
2. Build a boundary index of where each cell's agent storage begins

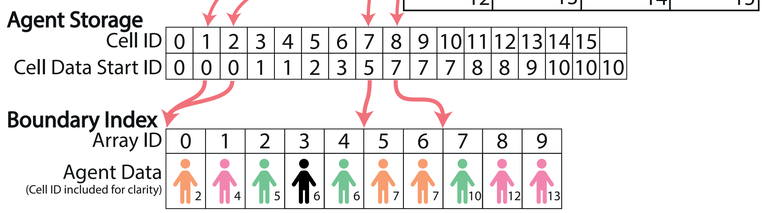
Figure: Bottom (1-2)



Implementation: Search

1. For each cell in the Moore neighbourhood
2. Locate the first index of the cell's data
3. Locate the first index of the next cell's data
4. For each agent within this range:
   - Calculate the Euclidean distance
   - If distance<= R:
     The agent is a valid neighbour

Figure: Bottom (1-3)



Optimising in 2 Dimensions

Strips

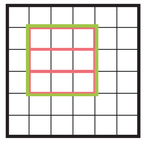
- We search the Moore neighbourhood of bins about a target location, this consists of a 3x3 grid

Figure: Right, Green

- Each row of this grid contains 3 contiguous bins (as stored in memory), therefore we can treat each 3-bin strip as a single bin

Figure: Right, Pink

- This modification decreases the number of bin changes from 9 to 3. This reduces redundant memory reads and the branch divergence that occurs when warp threads are operating on bins of differing sizes.

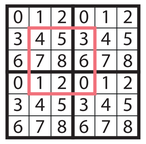


Modular

- In the existing technique each thread starts searching from the same point in the Moore neighbourhood, traversing the remainder of bins in the same order.
- This creates a potential for every bin to be accessed simultaneously during a single iteration of the neighbourhood search.
- Instead the environment is subdivided into bins in 3x3 groups.
  - Each bin within these groups is labelled 0-8, such that any 3x3 Moore neighbourhood placed on the environment will encapsulate 9 bins labelled 0-8.
  - Bins of each Moore neighbourhood are now iterated in label order.

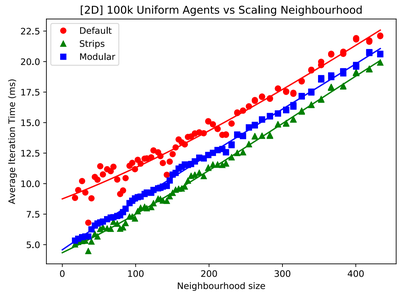
Figure: Right, Pink

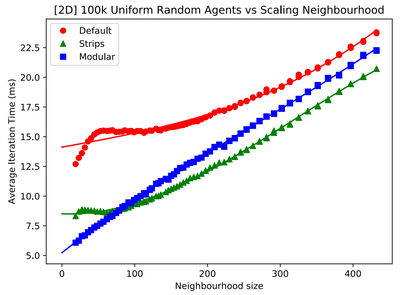
- This modification reduces the initial memory accesses to only access 1/9th of the bins.



Results in 2 Dimensions

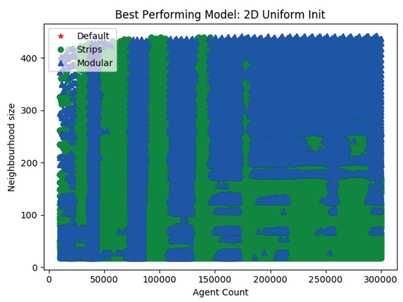
The graphs below show the performance of the existing technique (Default) and the two optimisation techniques Strips and Modular. These are tested on both uniformly (left) and uniform randomly (right) distributed agents with increasing neighbourhood sizes.

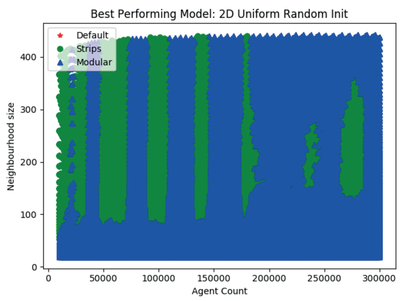




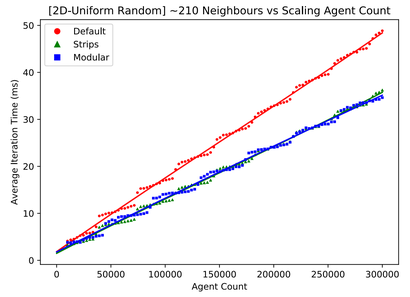
- Both new techniques consistently outperform the existing technique (Default).
- The Strips technique's performance is diminished for small neighbourhood sizes where the agent distribution is uniform random, such that agents are distributed with added noise.
- The Modular technique performs more than 2x faster than the existing technique (Default) with smaller neighbourhood sizes.
- When neighbourhood sizes are greater than 300, the Strips technique's performance improvement stabilises at around 3.5ms.

The scatter plots below show the most performant technique across a 2 dimensional parameter sweep of both problem scale (agent count) and neighbourhood size.





- Uniform agent placement clearly benefits the Strips technique. This ensures that the vast majority of threads avoid divergence.
- As device utilisation increases, the Modular technique's benefit has greater impact.
- Greatest speedup of 2.5x by Modular with 300k agents and 46 neighbours.
- The graph shown right, provides a cross section of the previous scatter.
- Strips performance with increasing agent count, shares the periodisation of the existing technique (Default), whilst Modular's is elongated.
- The elongated periodisation is likely due to Modular's reduction of scatter, reducing the amount of unique data being requested simultaneously.



Applying 2 Dimensional Techniques to 3 Dimensions

Strips

- The Strips technique can be extended to higher dimensions, however Strips still consist of 3 bins, so in 3D we reduce bin changes from 27 to 9.

Modular

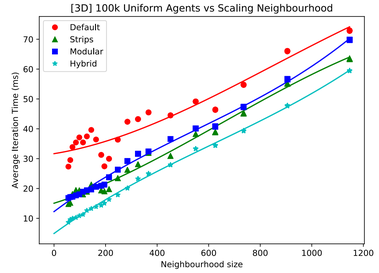
- The Modular technique can be extended to higher dimensions as before. This reduces initial memory scatter to 1/27 in the existing technique's worst case scenario.

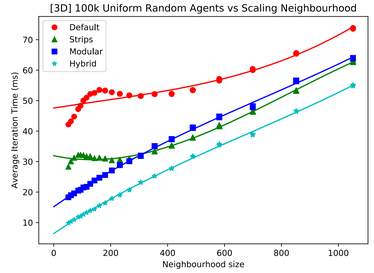
Hybrid

- Due to the benefits of the Strips modification being limited to a single dimension, it becomes possible to apply the Modular modification to the remaining dimensions.
- As the Strips modification already limits the starting bin scatter in one dimension, in combination with Modular it surpasses the benefits of both techniques.

Results in 3 Dimensions

Executing each implementation with 100k agents across a range of neighbourhood volumes presented the results below. This is the same experiment as presented in the central column, however in 3 dimensions with the addition of the Hybrid technique.





- The Hybrid technique's performance ranges from 4 -1.25x faster as neighbourhood size increases.
- A 2D parameter sweep of agent count and neighbourhood size also showed the Hybrid technique to perform fastest in all cases.
- The peak improvement was over 5x faster than default with 40k agents, and ~47 neighbours.
- The same general trend of performance can be observed in 3 dimensions as is visible in 2 dimensions.

Conclusions & Future Research

The work presented here has shown that there are high-level techniques for improving the performance of uniform spatial partitioning on GPUs. These benefits impact complex systems simulations, whereby there is constant demand to increase the size and speed of models.

In this work we have only currently considered the effect of the two parameters; agent count and neighbourhood size. There are many other parameters, relating to agent states (e.g. agent distribution), model specifics (our messages only contained a location vector) and hardware configurations (provided benchmarks were all carried out using a Titan X Pascal).

By the end of this research we hope to provide auto-tuning to achieve optimal performance for a given model and hardware configuration. This will allow complex systems simulation programmers to more easily access optimal performance for any given model relying on neighbourhood communication.