UNIVERSITY OF CAMBRIDGE

Agent Based Models using TensorFlow

Sharan Agrawal

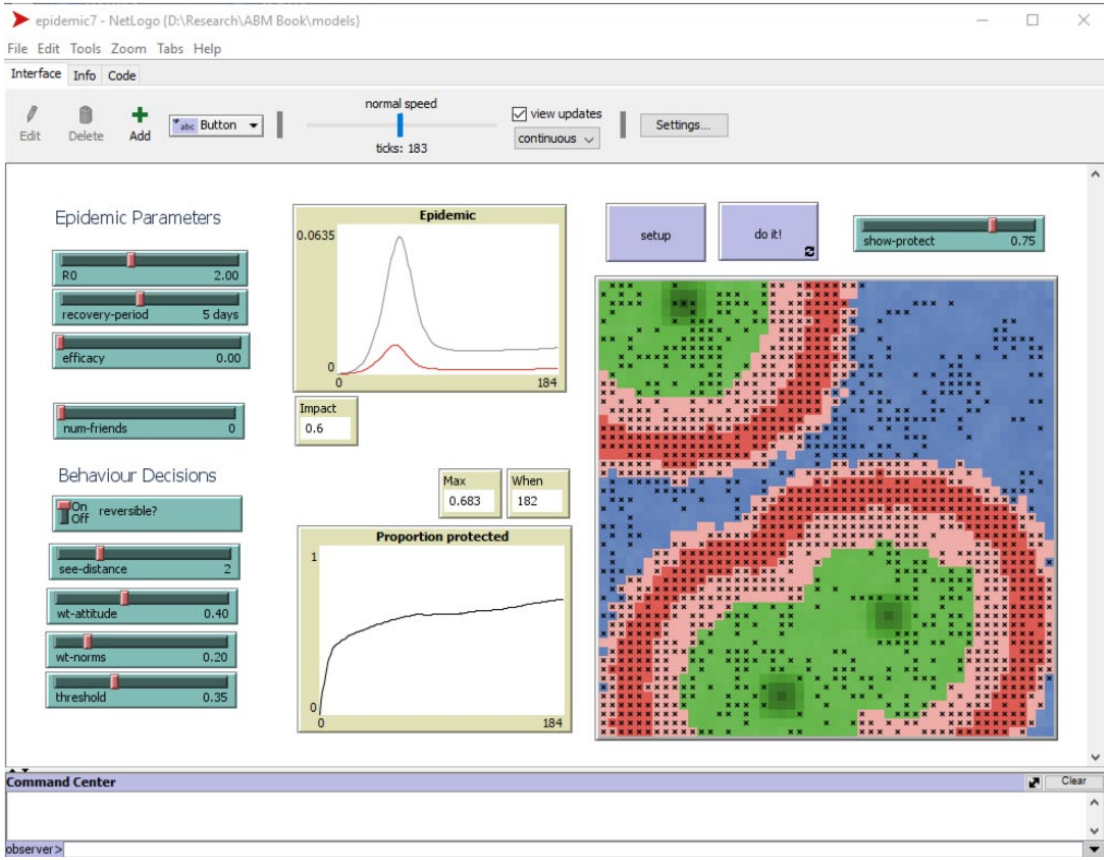
Large Scale Data Processing and Optimization

Background

- Agent Based Models (ABMs) simulate the interactions many autonomous agents in an environment and study the complex emergent behaviours that arise
- Very useful tool when modelling complex systems that are too difficult to model overall, or where complex behaviours result from non-equilibrium interactions between agents
- Recent increases in computation power has led to a surge in popularity, widely used across fields like economics, biology, ecology, epidemiology
- Example: Game of Life by Conway
- Technical challenges: scalability, scheduling, parallelization, synchronization
- Current approaches predominantly model each agent or groups of agents within one process and use MPI to interact with each other
- Systems like FLAME[1] use GPUs to speed up parts of the process

From [2]: Iterative Graph Processing

Background


epidemic7 - NetLogo {D:\Research\ABM Book\models}
File Edit Tools Zoom Tabs Help
Interface Info Code
Edit Delete Add Button
normal speed
ticks: 183
view updates
continuous
Settings...
Epidemic Parameters
R0 2.00
recovery-period 5 days
efficacy 0.00
num-friends 0
Behaviour Decisions
On Off reversible?
see-distance 2
wt-attitude 0.40
wt-norms 0.20
threshold 0.35
Epidemic
0.0635
0
0
184
Impact
0.6
Max
0.683
When
182
Proportion protected
1
0
0
184
setup
do it!
show-protect 0.75
Command Center
Clear
observer>

Dataflow ABMs

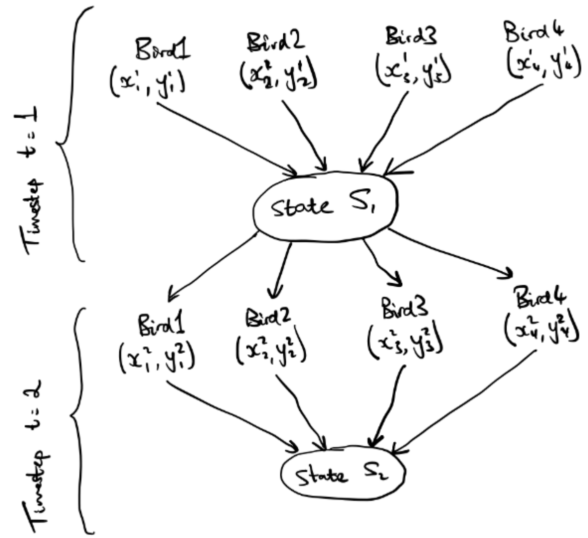
- Let's look at a particular subset of ABM simulations. For now assume a single type of agent. For a simulation running over time t=0, 1, 2…:

  - The state of every agent, $\mathcal{S}_t \in \mathbb{S}$, determined at $t-1$ and modified to $\mathbb{S}_t'$ at the end of t
  - A set of messages $\mathcal{M}_t \in \mathbb{M}$ that are determined at $t-1$, and modified to $\mathcal{M}_t'$ at the end of t
  - A set of operations/actions $A_t : \mathbb{S} \times \mathbb{M} \rightarrow \mathbb{S} \times \mathbb{M}$ which operate on the current state and messages and produce a new set of messages and states

- This turns the ABMs into a dataflow problem. We make the following statements:

  1. Environment is a function of global state
  2. State flows between each agent at every timestep, and is produced by agents
  3. Agent functions are the nodes

- Classic dataflow problem!



How can we address key computational issues?

- Scale:
  - Limits on ABM size now not imposed by message passing between agents
  - Can leverage extensive literature on scaling large scale dataflow by distributing computation and flow across network of distributed devices
- Scheduling:
  - The previous definition only allows synchronous updates with every agent forced to produce an update at every timestep
  - Can leverage literature on how to deal with micro-stragglers
  - Systems like NAIAD allow us to carefully design synchronous and asynchronous execution concepts in dataflow graph

Why would we do this in TensorFlow?

- Take advantage of TensorFlow's immense framework for orchestration of computation and messaging across many devices
  - Both its configurability and ability to determine hardware to run on automatically.
  - Also extensive literature on TensorFlow operation device placement, graph optimizers and compilers, etc.
- Systems like FLAME handle this with custom C code using MPI
  - Much harder for this orchestration and configurability to be usable by non-computer scientists.

What do we need for fully working version?

- A framework implementing Dataflow ABMs would need to do 3 things:

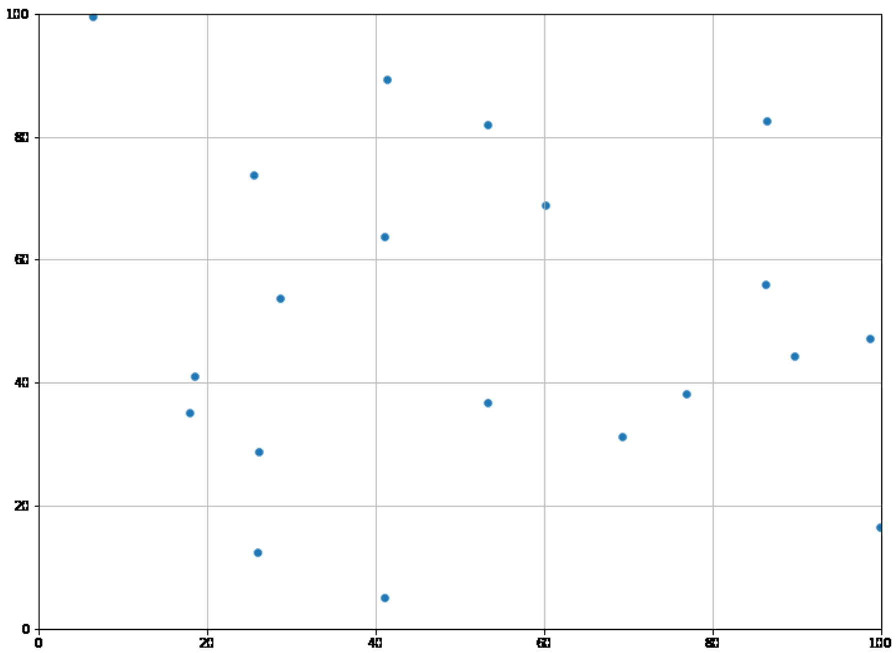
1. Define – allow definitions of ABMs that can be converted into DAG representations, could potentially use xparser for this
2. Collapse – collapse common state into a series of tensors rather than individual edges, and collapse actions on those tensors into single vertices
3. Orchestrate/observe/execute – execute the collapsed graph, let TensorFlow orchestrate across devices or have your own implementation, and observe changes in state variables

Concrete Steps Needed

1. Implement the 3 Stages above for a single agent type, constant number of agents, and a single datatype of state (e.g. int32, etc.)
2. Extend framework to multiple datatypes in state
3. Extend framework to multiple agent types and single datatype in state
4. Extend framework to multiple agents and multiple datatypes
5. Extend to varying number of agents
6. Add messages (transient data that aren't stored in state but modify agent behaviour)
7. Add concepts of timely dataflow to reduce assumptions around synchronous and orderless execution

Project Scope

- Goal: implement Proof of Concept, and then Step 1
- Reach goal: benchmark vs. existing frameworks using circles benchmark
- Achievements so far:
  - POC done for toy models
    - “Bird Flocking” →
  - Batched FRNN completed



Bird Flocking Example

Example:

We can look at the Bird Flocking problem to see how this defines some ABMs:

Here, we have one type of agent, a Bird that contains its location in $\mathbb{R}^2$, with 2 variables, x and y. At each step, each bird determines its velocity v_x, v_y and then updates it's x, y accordingly. It determines velocities by looking at the center of the flock and moving towards it, with some noise ϵ and max speed v_{max}. Say we have N agents, with positions at time t as $\mathcal{P} = \{(x_i^j\ y_i^j) \mid i = 1, \ldots, N,\ T = 0, \ldots M\}$ then our update rules at time $t = j$ would be:

$$x_{mean}^j = \frac{1}{N}\sum_{i=1}^{N} x_i^j \qquad y_{mean}^j = \frac{1}{N}\sum_{i=1}^{N} y_i^j$$

$$v_{x_i}^j = \max\left(x_{mean}^j - x_i^j, v_{max}\right) + \epsilon_i^j \qquad v_{y_i}^j = \max\left(y_{mean}^j - y_i^j, v_{max}\right) + \epsilon_i^j$$

$$x_i^{j+1} = x_i^j + v_{x_i}^j \qquad y_i^{j+i} = y_i^j + v_{y_i}^j$$

Note this system doesn't have any messages, only state and actions, since the messages are the state itself.