Performance & Computational Resource Considerations

Nathaniel Osgood
MIT 15.879

April 27, 2012

The Computational Burden of ABM

- Agent-Based models impose large computational burden
- Key factors:
  - "More moving parts": Lots of values to calculate and manipulate
  - Requirement of running multiple realizations

ABM and Computational Resource Use

- The computational burden of Agent-Based Models limits value delivered
  - Opportunity cost: Reduces time spent in exploration of model results & insights gained
  - Limited time => Less thorough exploration of parameter space => Reduced quality of calibration
  - Inhibits adoption

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Event-limited performance

- AnyLogic utilizes an “event driven” scheduler
  - The more events, the more the scheduler has to “wake up” to do things
  - In addition to the work to be done, there is some bookkeeping involved in the occurrence of an event
- If there is greater occurrence events (either explicit or implicit in e.g. transitions or messages), this will generally adversely affect performance

Hands on Model Use Ahead

Load Sample Model:

SIR Agent Based

(Via “Sample Models” under “Help” Menu)

Suggestions

- Permit disabling of visual elements
  - e.g. Checking “Ignore” on presentation of Agent
- Lower event frequency
  - Use dynamic events
  - Do more on firing of each event
  - Disable when not required
- Where possible, use “bookkeeping” on transitions (increase/decrease counts) rather than statistics
- Use a profiler to find where spending time
- Send events only for “infecting dose” (rather than exposure), where possible

Example of Reducing Events: Messages

- Sometimes there are simple ways to reduce event occurrence
- Example replacement
  - Worse performance: Sending “exposure” messages with rate α, each having a likelihood β of infection upon receipt
  - Better performance: Send “infect” messages with rate $\alpha\beta$
- Such simplifications are context-specific
  - For example, this transformation is much harder if the likelihood of infection given exposure varies by individual

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Statistics

- AnyLogic’s capacity to define “Statistics” over a population provides an easy way to compute population statistics
- Downside: Each computed statistic requires a full iteration through each member of the population
- Example: Classifying people into each of 17 age categories using “Statistics” requires 17 passes through the population!
  - This could in principle be done in a single pass – with each individual just incrementing different bins of a histogram

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Visualization

- The visual presentations of elements of a model takes considerable
  - Time
  - Memory
- Example: Dynamic properties
- Disabling visualization can lead to much faster operation

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Network Construction

- AnyLogic's Scale Free network requires a long time to run
- We have found gains by implementing the Barabasi-Albert algorithm ourselves

Significant Computational Factors

- Event-limited performance
- Statistics
- Visualization
- Network Construction
- Output of data

Database Output

- Batch up data to send to the database
  - Send in one big call to database, rather than multiple calls
- Use local database
- Record smaller subset of data
- Record less frequently
- Record fewer types of data

Model Space Demands

- Models with large populations can require much space
- I believe that the space demands can be particularly large when visualization is enabled
- You can enable space available for models in the "experiment" area
- Ways to reduce space demands
  - Accumulate less data (less frequently/fewer data items)
  - Write data out rather than accumulating in datasets

Exploiting Opportunities for Concurrency Using Distributed Processing

- ABM offers opportunities for parallel processing
- Two particularly manifest opportunities for concurrency require different levels of sophistication to exploit
  - "Embarassingly parallel" & easy to exploit: Concurrency between model realizations. One can readily run different realizations of a model in parallel (e.g. on different machines) & harvest results
  - Also parallelizable, but harder to exploit: Concurrency opportunities between distinct agents. While agent processing could in principle be parallelized, dependencies between agents (e.g. via message sending & joined flows) makes this more challenging to exploit.