GENESIS Social Simulation
Modelling Progress

Andy Turner

http://www.geog.leeds.ac.uk/people/a.turner/

International Symposium on Grid Computing 2010
Applications on Humanities & Social Sciences I
Taipei, Taiwan (2010-03-10)
Overview

• Introduction
• Geographic Dynamic Simulation Models
  – A demographic model
  – Developing a traffic model
• Scaling Up to City Size Models
  – Java OutOfMemoryError Handling
• Considerations and Further Work
Introduction

- Andy Turner
  - [http://www.geog.leeds.ac.uk/people/a.turner](http://www.geog.leeds.ac.uk/people/a.turner)
  - Blogger
    - [http://www.geog.leeds.ac.uk/people/a.turner/personal/blog/](http://www.geog.leeds.ac.uk/people/a.turner/personal/blog/)
  - GENESIS
    - [http://www.geog.leeds.ac.uk/people/a.turner/projects/GENESIS/](http://www.geog.leeds.ac.uk/people/a.turner/projects/GENESIS/)
- GENESIS
  - Generative e-Social Science for Socio-Spatial Simulation
  - a second phase research node of the UK National Centre for e-Social Science
  - [http://www.genesis.ucl.ac.uk/](http://www.genesis.ucl.ac.uk/)
My Paper, Presentation Slides, and Notes from this meeting can all be found via the following URL:

http://ur1.ca/ncsc
Geographic Dynamic Simulation Models

• **Geographic** means it is about interaction on or near the surface of earth
  – It is about people and their environment
• **Dynamic** means this represents change over time
  – Continuous change may be modelled
    • In an event based or scheduled way
      – Where each event triggers other events
    • With a specific temporal resolution
• A **simulation** is the run of a model
  – Based on input data and configuration
• The **model** is the simplification of reality
Two Agent Based GENESIS DSM

1. Demographic model
   - Run with time steps of a day and run for years
   - Initially aspatial

2. Traffic model
   - Run with time steps of seconds and run for days
   - Inherently spatial
Result can be reproduced although models are stochastic in nature

- Results are seeded by a pseudo-random number generator
  - Results can be easily reproduced provided the same input data and model configuration
  - A range of results can be generated for different random seeds

- Iterators are used to go through collections of Objects during processing
  - However, the order in which objects are retrieved via the iterator does not have an effect in that the data after going through the iteration is the same each time.

MoSeS meets NEC 10th March 2008
Basic Demographic Model

• Deals with birth and death
• Starts with an initial population
  – Comprised of males and females
• Males and females may have different age specific fertility and mortality rates
• At each step I simulate:
  – Death
  – Birth
  – Pregnancy
  – Miscarriage
Basic Demographic Model Detail

• All living Person (Agents) are tested to see if they die at each step:
  – Tests are done by asking for the next number from a pseudo-random sequence and comparing the number with respective age and gender specific mortality rates

• In the first simple model
  – Miscarriage rate was fixed for all ages, but was reasonably high
  – Gestation period was fixed at 266 days
  – At birth a single the Agent is formed and there is a 50% chance of it being either male or female
Example Output

- _Year 299
  - _TotalDeathsInYear 11442
  - _TotalBirthsInYear 12049
  - _TotalConceptionsInYear 14238
  - _TotalMiscarriagesInYear 2138

- [Output Directory](#)
Demographic considerations

• How realistic do the simulations results become?
• To begin with the initial population is not seeded with pregnancies
  – There are a large number of Persons added into the simulation 266 days after it starts
  – Over time miscarriage helps to even out the number of new births on each day
• Sharply increasing Fertility probabilities at a specific age result in a secondary cohort effect
  – The first newborn population tend to all have babies on the same day too
Example Platform

- Intel(R) Core(TM) 2 Duo CPU P9400 @ 2.4GHz
- 2.39 GHz, 1.95 GB RAM
- Ran from within Netbeans 6.7.1
- Java opts
  - -Xmx512m -Xms512m
Example Run Simulation

Parameters

• _MaximumNumberOfAgents = 1000000
• _MaximumNumberOfAgentsPerAgentCollection = 10000
• InitialFemalePopulation = 1000
• InitialMalePopulation = 1000
• TotalYears 300
Example Run Data

• No input data
• All data generated by simulation model
  – 1162563 Files
    • 1173895 Directories
• Size 1.92 GB (2067216392 bytes)
• Size on disk 4.48 GB (4819922944 bytes)
Developing a Traffic Model

• To model peoples movements on an individual level requires a way to store the location of each agent

• As a first step, agents were positioned in a confined region on a Euclidean 2D plane and made to move around this randomly by repositioning at each time tick
Further developing a Traffic Model

• Next the concept of a destinations was developed
  – Rather than necessarily having a different destination at each time tick, an Agent might be assigned a destination beyond its maximum range for movement in a time tick

• Scheduling Agents movements
• Collecting and using other data
A model for Leeds seeded with restricted UK Census data

• Focus on commuting
  – Journeys from home to work and back home again
  – Data
    • 2001 UK census special travel statistics data which gives the home origin and work destination at an Output Area (OA) level
      – There are over 200 thousand OAs in the UK
    • Uses Open Street Map (OSM) Road data
      – http://www.openstreetmap.org/
  – 3rd party libraries
    • Traveling Salesman OSM routing library
      – http://wiki.openstreetmap.org/wiki/Traveling_Salesman
    • GeoTools
      – http://www.geotools.org
Example output

- Output Directory
Memory Handling

• This is the crux of the paper I am presenting here
• I use memory handling to scale my model so I can run with billions of agents on an average machine
• Warning, the next few slides show Java code and I shall try to explain them so you can all understand
A method to get the amount of available JVM memory

```java
public long getTotalFreeMemory() {
    long result;
    try {
        long maxMemory = _Runtime.maxMemory();
        long allocatedMemory = _Runtime.totalMemory();
        long freeMemory = _Runtime.freeMemory();
        result = freeMemory + (maxMemory - allocatedMemory);
        return result;
    } catch (NullPointerException _NullPointerException) {
        if (_Runtime == null) {
            init_Runtime();
        }
        return getTotalFreeMemory();
    }
}
```
A method to try to prevent OutOfMemoryError being thrown

```java
protected HashSet<Long> ensureThereIsEnoughMemoryToContinue(
    AgentCollection a_AgentCollection) {
    HashSet<Long> result = new HashSet<Long>();
    while (_TestMemory.getTotalFreeMemory() < 20000000L) {
        Long swapped_ID =
            _AgentCollectionManager.swapToFileAgentCollectionExcept(
                a_AgentCollection);
        result.add(swapped_ID);
    }
    return result;
}
```
A typical public method

```java
public AgentCollection get_AgentCollection(boolean handleOutOfMemoryError) {
    try {
        AgentCollection result = get_AgentCollection();
        _Environment.ensureThereIsEnoughMemoryToContinue(
            result,
            handleOutOfMemoryError);
        return result;
    } catch (OutOfMemoryError a_OutOfMemoryError) {
        if (handleOutOfMemoryError) {
            ...
        } else {
            throw a_OutOfMemoryError;
        }
    }
}
```
A simple case of handling OutOfMemoryError

```java
_Environment.clear_MemoryReserve();
AgentCollection result = null;
try {
    result = get_AgentCollection();
    _Environment._AgentCollectionManager.swapToFileAgentCollectionExcept(
        result);
    _Environment.init_MemoryReserve(
        result,
        false);
    _Environment.ensureThereIsEnoughMemoryToContinue(
        result,
        handleOutOfMemoryError);
    return result;
} catch (OutOfMemoryError b_OutOfMemoryError) {
    b_OutOfMemoryError.printStackTrace();
    System.exit(OutOfMemoryErrorHandler.OutOfMemoryErrorExitStatus);
    //throw b_OutOfMemoryError;
}
return result;
```
A method for initialising a memory reserve to free up if an OutOfMemoryError is encountered

```java
public final void init_MemoryReserve(
    AgentCollection a_AgentCollection,
    boolean handleOutOfMemoryError) {
try {
    init_MemoryReserve();
    a_AgentCollection._Environment.ensureThereIsEnoughMemoryToContinue(
        a_AgentCollection,
        false);
} catch (OutOfMemoryError a_OutOfMemoryError) {
    if (handleOutOfMemoryError) {
        ...
    } else {
        throw a_OutOfMemoryError;
    }
}
}
```
a_AgentCollection._Environment.clear_MemoryReserve();
try {
    AgentCollectionManager a_AgentCollectionManager
        = a_AgentCollection.get_AgentCollectionManager();
a_AgentCollectionManager.swapToFileAgentCollectionExcept(
        a_AgentCollection,
        false);
a_AgentCollectionManager = null;
System.gc();
//swapToFileData();
a_AgentCollection._Environment.init_MemoryReserve(
        a_AgentCollection,
        false);
} catch (OutOfMemoryError b_OutOfMemoryError) {
    System.err.println(
        "Fatal OutOfMemoryError in " +
        this.getClass().getName() +
        ".init_MemoryReserve(AgentCollection,boolean");
    System.exit(OutOfMemoryErrorExitStatus);
    //throw b_OutOfMemoryError;
}
Methods for creating and freeing up a memory reserve

```java
private final void init_MemoryReserve(
    int size) {
    if (this._MemoryReserve == null) {
        this._MemoryReserve = new int[size];
        Arrays.fill(
            this._MemoryReserve,
            Integer.MIN_VALUE);
    }
}

public final void clear_MemoryReserve() {
    this._MemoryReserve = null;
    System.gc();
}
```
public Long swapToFileDialogException(
    AgentCollection a_AgentCollection) {
    Iterator a_Iterator = this._AgentCollection_HashMap.keySet().iterator();
    Long b_AgentCollection_ID = null;
    AgentCollection b_AgentCollection = null;
    while (a_Iterator.hasNext()) {
        b_AgentCollection_ID = (Long) a_Iterator.next();
        b_AgentCollection =
            (AgentCollection) this._AgentCollection_HashMap.get(b_AgentCollection_ID);
        if (b_AgentCollection != null) {
            if (b_AgentCollection != a_AgentCollection) {
                b_AgentCollection.write();
                b_AgentCollection = null;
                break; // to break out of while
            }
        }
    }
    this._AgentCollection_HashMap.put(
        b_AgentCollection_ID,
        b_AgentCollection);
    return b_AgentCollection_ID;
}
Agents and Data Structures

• Agents
  – Each of these has a unique numeric ID
  – Have a file location

• AgentCollections
  – These are HashSets of Agents
  – Have a file location

• IDCollections
  – These are HashSets of Agent numeric IDs
Data Structure Settings

```c
ager._MaximumNumberOfAgents = 1000000000000L;
ager._MaximumNumberOfAgentsPerAgentCollection = 10000;
ager._MaximumNumberOfObjectsPerDirectory = 100;
```

- 100000000000
- 1000000000
- 10000000
- 100000
- 10000
- 1000
- 100

6 directory levels can store 1000000000000 (one hundred thousand million) files.
Considerations and Further Work

• I am happy to help anyone wanting to get my code running on their laptops while I am here
  – There is no set up documentation at the moment and not all what is needed is in the subversion repository, but it is all online and open source.

• Meetings with colleagues at ASGC and ASCSR...
More to do...

- There is no end to the task of detailing a digital version of what has happened and what might happen let alone considering post casting what might have happened
  - Social simulation is fun, geographical, so please get involved if you are interested and we can work together on this
Acknowledgements

• GENESIS is funded by the UK ESRC
  – RES-149-25-1078

• Thanks for help and support from:
  – The international e-Research community
  – The University of Leeds
    • School of Geography
    • Centre for Computational Geography
Thank you

http://url1.ca/ncsc

MoSeS meets NEC  10th March 2008