# Network Analysis with UCINET for ABM of Industrial Districts and Economic Geography

Richard Taylor

Centre for Policy Modelling,

The Business School,

Manchester Metropolitan University,

Manchester, UK.

E-mail: r.i.taylor@mmu.ac.uk

# Outline of Tutorial

Objectives of the course

The software we'll be using

UCINET – JAVA – RePast – RealJ

Graph theory and network analysis

Agent-based modelling

Industrial Districts and economic geography

# Outline of Tutorial

## Day 1

- **Introduction to the software**
- The ID network model
- Running the **model via GUI**
- **Customising the model**
- Java basics
- **The Java-RePast source code**
- **Collecting networks dataset**

## Day 2

- **Introducing UCINET**
- Using DRAW to view your network
- **Concepts of Network Analysis**
- **Characterising your network**
- Static and dynamic networks
- **The Second Industrial Model**
- Presenting the results

# Software

- **Java Development Kit (JDK) version 1.5.0 – object orientated, platform independent, widely used. Arranged into packages.**
- **RePast 2.2 – Set of Java packages for ABM. GUI for visualisation. Bytecode in repast.jar**
- **RealJ IDE – Simple environment for editing java files, compiling and running programs**
- **UCINET – network analysis software**

# Models of IDs and Geography (Revision)

Industrial clusters – networks with cohesive structure

- Economic success
- Whole is greater than the parts
- Characterised by innovation
- Highly differentiated firms
- Diffuse market
- 'Star' or hierarchical networks - one large manufacturer & many subcontracting firms

Economic geography – success of the ID model depends upon geographical proximity of firms

# Social Network Analysis (Revision)

## Graph / network theory

A graph G(V,E) consists of a set of vertices V representing individuals or objects and a set of edges E representing relationships between the individuals or objects

- Directed and undirected graphs, valued graphs
- Degree, indegree, outdegree
- Density, connectedness, subgraphs
- Average path length, cliquishness, clustering
- Small world, scale-free
- Centrality, periphery

# Background on Agent-based modelling

Background in Distributed Artificial Intelligence (DAI)

Properties of Agents

- Autonomy
- Adaptation
- Interactive
- Heterogeneous

Properties of Agent Systems

- Flexible
- Scalable
- Distributed
- Robust

Many of these properties are shared with social systems → argument for the usability of the approach

# Basic principles

- Agents represent the actors in the system, i.e. firms, institutions
- We define agent characteristics as well as their behaviour
- These are implemented as rules in the computer program
- An agent is like an object in OOP ….
- … but normally it has some goals, some means of perceiving its environment, and some kind of reasoning mechanism
- Agents should be embedded within social context

# Examples

- Behavioural norms such as fashion trends or religion
- Group behaviour such as in crowds, traffic or urban spaces
- Environmental models of land use change or water demand
- Consumer behaviour in retail markets
- Auctions and value-chain models

# Getting started with the software

Objective: to become familiar with RealJ and to run the ID network simulation

The JDK consists of the bytecodes for the whole of the Java core, as well as the tools for compiling (javac.exe) and running (java.exe) your own Java programs

RealJ is a text editor for working on Java projects which has some built-in functions for linking with the Java tools

- also known as Integrated Development Environment (IDE)

RealJ splits the workspace into 3 components: text editor, project window, console panel

# Getting started with the software

Objective: to become familiar with RealJ and to run the ID network simulation

Guidelines for the practical:

1. In RealJ, open a new project named “IDNetworkModel.jpr”
2. To this project add the 2 files of the ID model (IDNetworkModel.java & Firm.java)
3. In the project window, set IDNetworkModel to ‘main’ class
4. Also in this window, set the classpath to include “repast.jar”
5. Compile (Build – Compile) and then run (Build – Run Application) it
6. Press ‘play’ on the RePast toolbar and then watch the display appear

# Exploring the ID network model

Objective: to become familiar with the ID network simulation and to explore it via the RePast toolbar (GUI)

The ID network model consists of two types of agent: client and subcontractor firms, and shows the relations among them

The basic steps are:

1. Create the firms
2. Select at random one client and one subcontractor and link them
3. Repeat step 2 for the specified number of iterations

Parameters:

NUM_CLIENTS = 10, NUM_SUBCONTRACTORS = 60,

NUM_LINKS = 100

# Exploring the ID network model

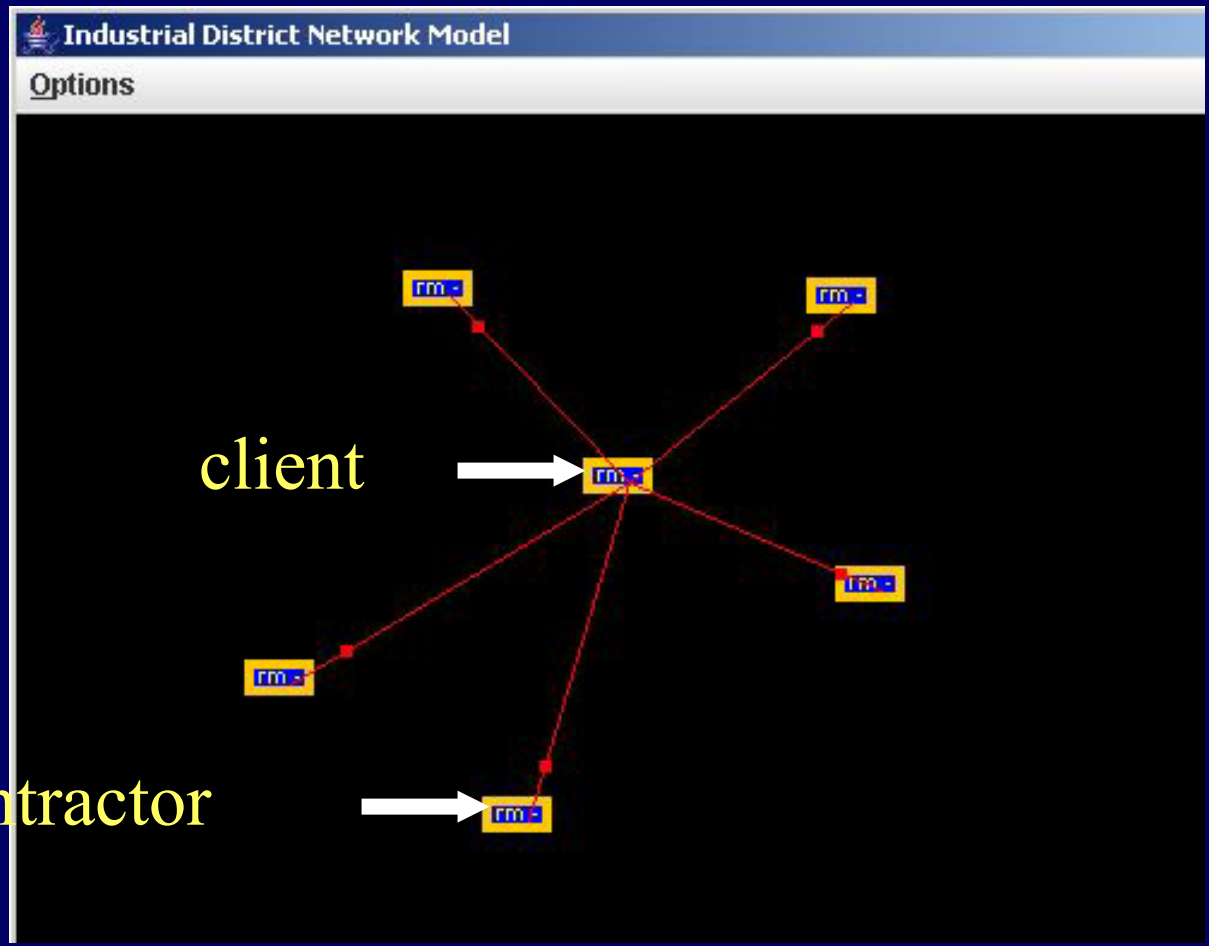


Figure 1. Illustrating the ID network model

```
// Java pseudo code for ID networks model
// create 70 new firms  f(0), f(1) …., f(69)  and add them to an agent list
For (index = 0 to 69) {
        f(index) = new Firm();                          // create the new firm: f(0),f(1), …
        agentList.add( f(index) );                      // add it to the agent list
}
// select one client and one subcontractor at random and link them, 100 x
// with index 0 to 9 the firm is a client; with index 10 to 69 a subcontractor (s/c)
For (linkID = 0 to 99) {
        clientID = RandomInteger(0 to 9);               // select random client ID from 0 to 9
        scID = RandomInteger(10 to 69);                 // select random s/c ID from 10 to 69
        link(linkID) = new Edge();                      // create a new link (Edge): link(0), ...
        f(clientID).addOutEdge( f(scID) );              // add the link out from client
        f(scID).addInEdge( f(clientID) );               // add the link to the subcontractor
}
Display(agentList);                                     // display the firms and their network
```

# Exploring the ID network model

Objective: to become familiar with the ID network simulation and to explore it via the RePast toolbar (GUI)

Guidelines for the practical:

1. Compile (Build – Compile) and then run (Build – Run Application) the ID network model
2. Start it via the RePast toolbar
3. Using the toolbar, reset the simulation and start it again
4. Explore the model by inputting different parameters in the control panel
5. Using a text editor, look at the output data files, “IDNetwork.dl” for your network

# Understanding the source code

Program uses some Java core packages as well as those of RePast.

Some Java basics:

The ‘main’ class takes the name of the model, the same as the RealJ project file (eg. “IDNetworkModel.jpr”)

Use of keyword ‘import’, e.g. :

```
import uchicago.src.sim.gui.*;
```

Other classes which are used by ‘main’ must be included in the header of main or must be added to the project (e.g. “Firm.java”)

Classes are like ‘templates’ for creating Java program ‘objects’

# Understanding the source code

Each class contains some variables to store numerical / symbolic values, as well as some methods for manipulating those variables, of the form:

public int getXValue () { ….}  …

… is ‘called’ with the command getXValue()

New objects are created from the class templates with the ‘new’ keyword, e.g. :

Firm napolianBakery = new Firm();

# Understanding the source code

RePast divides the model implementation into separate parts:

Setup: to set (or reset) any initial parameters to their defaults and to set any objects to 'null'

BuildModel: to create the representational parts of the simulation, i.e., the agents and their environment

BuildDisplay: builds those parts of the simulation that are needed for graphically displaying the simulation to a user.

BuildSchedule: for scheduling 'actions' that change the simulations state i.e., that describe a dynamic simulation of social processes

# Exploring the ID network model

Objective: to understand how the demonstrative source code builds the model and displays it via the RePast GUI

Guidelines for the practical:

1. In the source code, try to identify classes, objects, variables & methods
2. Which parts of it refer to elements of the Java core and which to RePast?
3. Locate the Java docs and the RePast docs and read about the classes
4. Describe what the various parts of the program do (if necessary, use the earlier pseudo code to help you do so).
5. Explore the model by changing the parameters in the source code directly, then compiling and running the program as before