G54DIA:
Designing Intelligent Agents

Tutorial 4: Multi-Agent Solutions

(based on slides by Julian Zappala)

Brian Logan

School of Computer Science

bsl@cs.nott.ac.uk

Outline of this tutorial

- MAS environments
- MAS design problems: organisation, control, task sharing & communication
- MAS implementation issues
- examples:
  - upgrading the `DemoSimulator`
  - upgrading the `TankerViewer`

MAS environments

- is it possible for more than one agent to be in the same cell?
- is it possible for more than one agent to perform the same action (in the same cell) at the same timestep?
- if not then how are these constraints enforced fairly?
- if multiple agents can be in the same cell/perform an action at the same time, what are the consequences?

MAS organisation

- how many agents?
  - when does cooperation become competition?
  - may require experimentation/calculation …
- what types of agents?
  - homogeneous: all agents are the same
  - heterogeneous: agents have different abilities
  - what roles might agents take?
  - are roles fixed or variable?

Control structure

- anarchy: no control structure
- hierarchical: e.g., one or more “manager” agents control other agents
- distributed: e.g., agents decide collectively or ask other agents to perform tasks

Task sharing

- how are goals/tasks allocated to agents?
- one agent per task?
- more than one agent per task (how many)?
- implications for exploration and exploitation?

Communication

- how is information exchanged:
  - shared memory?
  - message passing (agent to agent, or broadcast)?
  - other approaches, e.g., pheromones? face to face?
- what information is shared?
  - about the environment?
  - about tasks?
  - about this agent, e.g., what it's current plans are?

Implementation issues

- simplest approach is to iterate over a collection of agents
- agents sense the environment and perform actions in series
- if agents are always processed in the same order, this has fairness implications
- e.g., choices of later agents are constrained by those of earlier agents (don't worry about this)

Multi-threaded implementations

- parallel (multi-threaded) implementations are *not* a good idea:
  - library code is not thread-safe
  - race conditions mean that testing and debugging become more difficult

Example: multilibrary & multidemo

- extends the Java Agent Package with multiple tankers (a `Fleet` class that implements a collection of tankers)
- illustrates one of many ways in which a multi-agent solution could be implemented
- one of many ways to upgrade the user interface
- highlights some of the classes you may need to adapt
- gives some hints and code fragments
- not a full solution

Upgrading DemoTanker

```
for (Tanker tank:fleet) {
    // Get the current view of the tanker
    Cell[][] view = env.getView(tank.getPosition(), Tanker.VIEW_RANGE);
    // Let the tanker choose an action
    Action a = tank.senseAndAct(view, env.getTimestep());
    // Try to execute the action
    try {
        a.execute(env, tank);
    } catch (OutOfFuelException dte) {
        System.out.println("Tanker out of fuel!");
        break;
    } catch (ActionFailedException afe) {
        System.err.println("Failed: " + afe.getMessage());
    }
}
```

Upgrading the GUI

- upgrading the GUI is optional but it may be useful for debugging etc.
- possible upgrades include:
  - selecting which agent to view
  - showing other agents in the view
- you'll need some familiarity with Java Swing libraries, e.g:

http://download.oracle.com/javase/tutorial/uiswing/index.html

http://download.oracle.com/javase/6/docs/api/javax/swing/package-summary.html

Selecting a tanker to view 1

```
String[] tankerNames = new String[fleet.size()];

for (int i = 0; i < fleet.size(); i++) {
    tankerNames[i] = "Tanker " + i;
}

//A drop down list to select which tanker to view
tankerList = new JComboBox(tankerNames);
infop.add(tankerList);

//Event handler for drop down list
tankerList.addActionListener(this);
```

Selecting a tanker to view 2

```
@Override
public void actionPerformed(ActionEvent arg0) {
    tank = fleet.get(tankerList.getSelectedIndex());

}
```

- `TankerViewer` must implement the `ActionListener` interface

Displaying multiple tankers 1

```
tankers = new JLabel[SIZE][SIZE];
JPanel pTankers = new JPanel(new GridLayout(SIZE, SIZE));
pTankers.setOpaque(false);

//Add a tanker Icon to each cell in the Tanker Panel
for (int y = 0; y < SIZE; y++) {
    for (int x = 0; x < SIZE; x++) {
        tankers[x][y] = new JLabel();
        tankers[x][y].setIcon(iconfactory.getIconForTanker(Tanker));
        tankers[x][y].setVisible(false);
        pTankers.add(tankers[x][y]);
    }
}

//Add to layered panel in the foreground
lp.add(pTankers, new Integer(1));
```

Displaying multiple tankers 2

```
for (int x = 0; x < SIZE; x++) {
    for (int y = 0; y < SIZE; y++) {
        Icon cur = iconfactory.getIconForCell(view[x][y]);
        cells[x][y].setIcon(cur);
        //Hide previous tankers
        tankers[x][y].setVisible(false);
        //Show current tankers
        for (Tanker t : fleet) {
            if (view[x][y].getPoint().equals(t.getPosition())) {
                tankers[x][y].setVisible(true);
            }

        }
    }
}
```