The Giving Game: Documentation

The Giving Game

f

University of Amsterdam f

Julian Ruger

Contents

1. Introduction

In this document the usage of the simulator for the giving game will explained and the code will be documented. The usage of most variables will be explained inside the code itself. This document will focus on how each function is used and specific algorithms will be explained.

2. Instructions

Installation

requirements

Usage

Images to explain each button and graph etc. Python 3.4+

3. Code

This program is written with python. The code consists mostly of object oriented python code.

Back-end

The back end consists of the following files:
The usage of each function in each file and the most important variables will be explained.

Agent.py

The Agent class consists of multiple functions to update information about each agent. Each agent knows only about the things the agent is part of. For example each agent only knows about its own transactions. Each agent has an unique id which can be used as a index for some lists. The index of these lists is the same as the id to make navigating through these lists easy.
The following python packages are used:

- Numpy

```
def receive(self, P, Q):
```

Goods.py

The Good class consists of variables that determine what kind of good it is. Each good has an unique id, which can be used as the index of some lists. The index of these lists is the same as the id to make navigating through these lists easy.
For this class no special packages are used.

Enviroment.py

The Enviroment class consists of multiple functions to create the enviroment of the giving game, to update variables after each transaction and to calculate the community effect. Most of the variables are stored in this class.
The following python packages are used:

- Numpy

```
def create_agents(self):
```

```
def create_goods(self, goods):
```

```
def notify_agents(self):
```

```
def update_balancematrix(self, P, Q):
```

Simulate.py

This file does not use any object oriented code. This file consists of multiple functions that call the necessary functions from other files like Enviroment.py. This file can be seen as the engine for the simulation. The creation of the enviroment, transactions and updating the UI take place in this file.
The following python packages are used:

- Numpy

```
def create_enviroment(N, M, goods_list, M_perishable, perish_period,
                      production_delay, value, parallel, selectionrule):
```

Selection_rules.py

This file does not use any object oriented code. This file consists of the function used to select the next agent. The different algorithms for each selection rule will be explained below.
The following python packages are used:

- Numpy

```
def random_rule(N):
    return randint(0, N-1)
```

```
def balance_rule(balance_matrix, current_agent, N):
    next_agents = []
    highest_balance = balance_matrix[current_agent].min()
    for x in range(N):
        if x != current_agent:
            if balance_matrix[current_agent][x] > highest_balance:
                next_agents = [x]
                highest_balance = balance_matrix[current_agent][x]
            elif balance_matrix[current_agent][x] == highest_balance:
                next_agents.append(x)
    if len(next_agents) > 1:
        return next_agents[randint(0, len(next_agents)-1)]
    return next_agents[0]
```

```
def goowill_rule():
```

Front-end

The front end consists of the following files:
The usage of each function/class in each file will be explained.

GUI_Input.py

The Input class is consists of multiple functions to handle the user input so that the parameters can be used to create the enviroment and start the simulation.
The following python packages are used:

- Numpy
- Qt

```
def setAgents(self, value):
```

GUI_Output.py

This file consists of multiple classes. Each class is used to create a part of the UI. The main class is the Output class.
The following python packages are used:

- Numpy
- Qt

```
class GUI(QtGui.QWidget):
```

Visualisation.py

4. Releases

5. Further research

Requirements

The following Python packages/frameworks are mandatory.

- Qt Framework
- MatplotLib
- Numpy