Appendix A

How to use the simulation tool

This section describes the features of the simulation tool, and how to use it. The tool was created using Java and uses an external library, called JUNG, (as mentioned before). There are two main user interfaces that the user interacts with in the creation and simulation of the game.

A.1 The main interface

The main interface contains the title 'Spatial Games' and allows the user to set up the game, by two methods:

1. loading a text file which contains the details about the game,
2. specifying the settings using the drop-down menus and the input boxes, and submitting the settings using the submit button.

An image of the main menu interface is depicted in figure A.1.

If the user selects to load a file, the interface will return an error dialogue if the provided file is not a text file, namely it's file extension is must be `.txt`. Furthermore the text file must be in the correct format, and again the user will receive an indication if the file is given in the incorrect format. Details of how to figure the input text file are given in section **A.1.1**. If the user selects to specify the settings using the interface, then the user must enter the correct information in the text fields. Once all this is entered, the user can create the game using the submit button. If the information provided is of an incorrect format, then error dialogues will appear to the user for each text field with the wrong data. Section **A.1.2** specifies the correct format for entering the information into the interface.

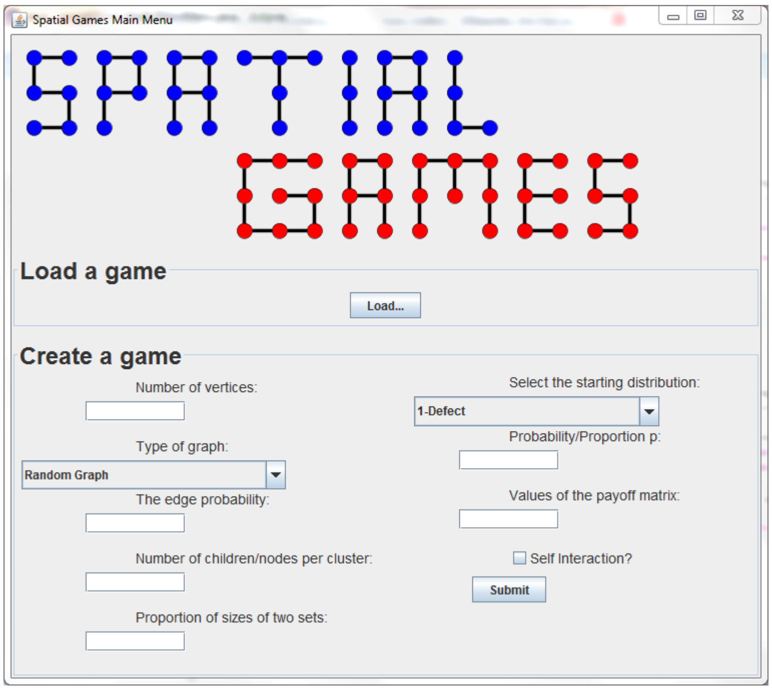


Figure A.1: The main menu of the simulation tool

A.1.1 Configuring the text file

The `.txt` file provided must be in the correct format. For a file to be accepted by the simulator it must:

- specify the number of vertices in the game, using the key phrase ‘`number_of_vertices=`’ followed by the actual number of vertices. Importantly there must be no space between the phrase and the number;
- provide the adjacency matrix of the graph of the game. The key phrase ‘`begin_matrix`’ starts the specification of the adjacency matrix, and so any lines following it must be the adjacency matrix (or empty lines and comments). The adjacency matrix must be specified in the format presented by the example in figure A.2. Furthermore, the adjacency matrix must contain the correct amount of entries, most notably it must match the number of vertices specified by the key phrase from above;
- designate the starting distribution of strategies using the key phrase ‘`begin_distribution`’ with the distribution presented in a line after the indication. The format to present the distribution is illustrated in the example of A.2. It must also contain the correct amount of entries, namely matching the number of vertices specified.

Optionally the user can:

- indicate the game’s payoff matrix using the key phrase ‘`payoff_values=`’ followed by a list of the payoff values in the form `T,R,P,S` (no spaces), where $\mathtt{T} > \mathtt{R} > \mathtt{P} > \mathtt{S}$ and in $\mathbb{R}$. If the user does not provide this, then the game would be

```
number_of_vertices=5

begin_matrix
0 1 1 1 1
% this is a comment
1 0 1 0 0
1 1 0 0 1
1 0 0 1 0
1 0 1 0 0

begin_distribution
1 1 1 0 1

payoff_values=4.5,2,1.25,0
```

Figure A.2: An example of a readable file for the simulator.

created with the default payoff matrix, namely with values 5,3,1,0.

- write comments, using the sign `%` at the start of the line. Importantly comments cannot begin at the middle of a line. Comment lines can intersect the specification of the adjacency matrix.

An example of a correctly formatted file which is readable by the simulator is in figure A.2. The example illustrates the format of the adjacency matrix and the distribution, as well as a comment intersecting the matrix.

A.1.2 Game creation using the interface

To use the interface to create a game, the user has to specify the number of vertices in the text field below '`Number of vertices:`'. This must be a positive integer. For any other input, the interface will issue an error, and will fail to create the game. Further more depending on the graph type chosen, the user must indicate:

- the edge probability, which is the probability given to an edge existing in the graph, in the correct text field. The value must be between 0 and 1, else an appropriate error dialogue is given. This field needs only to be filled if the user is creating a game with the random graph and bipartite graph types;
- the number of children or the size of a cluster. This is only applicable for the regular tree and cluster graphs. The value entered needs to be > 0, else the interface will return an error to the user;
- the proportion to split the vertices into two disjoint sets, under `Proportion of sizes of two sets:`. This is strictly only for the bipartite graph types. The given input must be between 0 and 1;
- the probability that a vertex starts with defection/the proportion of vertices starting with defection. This is used for two starting distributions, p-Proportional

Defect and p-Random Defect. It must be between 0 and 1.

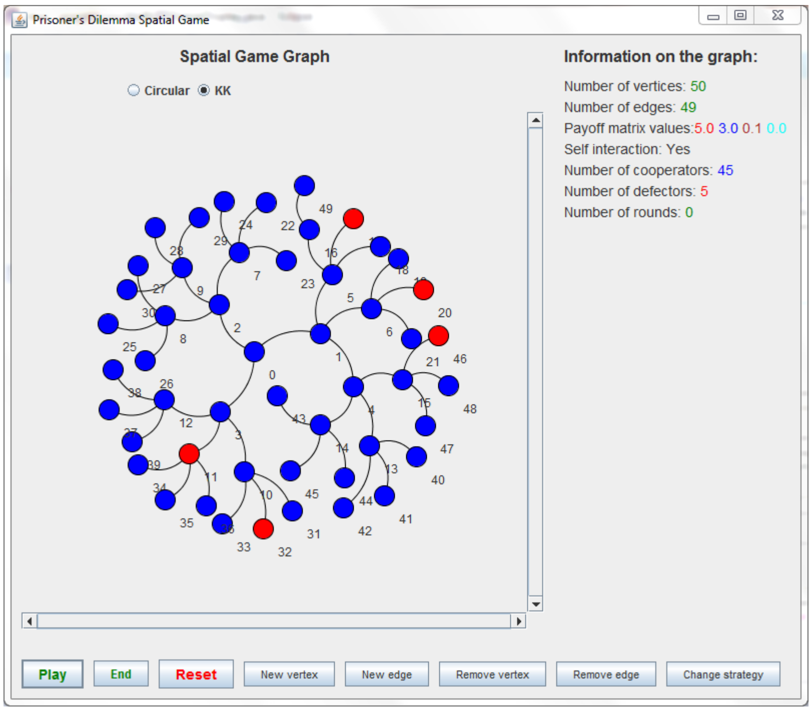
The user can also specify the payoff matrix, and if it is left unfilled then the game will be played with the default matrix. Additionally, the also specify if the game includes self interaction. Once all appropriate fields are filled, the user can create the game by pressing the submit button.

A.2 The game interface

Once the user has submitted a game, the main menu will spawn a new separate interface containing the game. An example of the simulation interface is shown in A.3. At this point the user is given a selection of actions to perform.

At the top of the interface lies the title, as well as two radio buttons called `Circular` and `KK`. These buttons switch between different visual graph layouts, and are purely for the visualisation aspects of the game. Below this option panel lies the graph of the game, providing the visuals of the game. To the right is a panel which contains information about the game, such as the number of cooperators the game has, in its current round. At the bottom of the interface is a panel of buttons, where each button provides an interaction to the graph:

- `Play` - simulates a round of the spatial game, updates the visualisation;
- `End` - updates the game to a stable point/equilibrium;
- `Reset` - resets the game to its original state;
- `New vertex` - adds a player/vertex to the game. The interface opens a dialogue which allows the user to input the new vertex's strategy and neighbours;
- `New edge` - adds new edges to the game. A dialogue is used to specify the new edges to be add to the game;
- `Remove vertex` - removes a specified vertex;
- `Remove edge` - removes a given list of edges from the game;
- `Change strategy` - changes the strategy of a vertex.



Information on the graph:

Number of vertices: 50
Number of edges: 49
Payoff matrix values:5.0 3.0 0.1 0.0
Self interaction: Yes
Number of cooperators: 45
Number of defectors: 5
Number of rounds: 0

Figure A.3: An example game with the game interface