Corso di Laurea in Informatica—Università di Bologna
Progetto per il Corso di Sistemi Complessi
AA 2012/2013

Ozalp Babaoglu Moreno Marzolla
Dipartimento di Informatica — Scienza e Ingegneria, Università di Bologna

1. *Project outline*

Introduction. In this project we will experiment with the basic concepts of game theory. We consider an undirected graph $G=(V, E)$ where each node represents an agent, and edges connect agents that may interact. Each agent v in V has a fitness $F[v]$ which can change after an interaction. The fitness $F[v]$ is a real value, higher is better; negative values are allowed; the initial value is zero.
The system evolves in discrete steps, and therefore can be modeled using the PeerSim cycle-based simulation engine. During a simulation cycle, each agent selects a random neighbor to interact with. When two agents interact, each one pick a *strategy*, and the outcome of the interaction (change of the fitness of both agents) is given by the strategies that have been selected. For simplicity, we assume that there are only three strategies, denoted as {A, B, C}. The outcome of each interaction causes the fitness of the two agents to change according to a pre-defined payoff matrix. The payoff matrix is a 3x3 matrix whose elements describe the outcome of all possible interactions among strategies. We require the payoff matrix to be symmetric. An example of payoff matrix is the following:

	A	**B**	**C**
A	1 / 1	1 / 3	-2 / 0
B	3 / 1	0 / 0	3 / 2
C	0 / -2	2 / 3	1 / 1

The matrix is read as follows: when two agents playing strategy A interact, both get a payoff of 1 and therefore increase their fitness accordingly. If an agent playing strategy A interacts with another playing strategy B, the first one increases its fitness by 1 and the second increases its fitness by 3. If an agent playing A interacts with an agent playing C, the fitness of the first agent is *decreased* by 2, while the fitness of the second one is unchanged. Note that we allow only symmetric matrices, and the diagonal elements myst contain the same payoff for both agents. in other words, if two interacting agents play the same strategy, both get the same payoff.
If there are N agents in the system, then each simulation cycle executes exactly N interactions: each agent selects a random neighbor, and they both update their fitness according to the payoff matrix and the strategy they choose; note that it is possible that the same pair of agents u and v interact twice during a cycle, for example, if u randomly selects neighbor v and v randomly selects neighbor u.

Project Goal. The system dynamics is influenced by two main factors:

- The values in the payoff matrix;
- How agents select a strategy at each interaction.

The goal of this project is to study how the average fitness of each agent is influenced by the factors above. First, we consider a very simple scenario, where each agent always picks the same strategy; then, we consider a more complex scenario in which the chosen strategy can change.

First scenario: constant strategy. Build a simulator which instantiates a network of N peers. Then, in the configuration file specify the following parameters (other parameters may be needed, of course):

- the fraction XA, XB and XC of agents playing strategy A, B and C, respectively (XA + XB + XC = 1);
- the values of the payoff matrix.

As a starting point you may want to consider the following payoff matrix:

	A	**B**	**C**
A	0 / 0	0 / 1	1 / 0
B	1 / 0	0 / 0	0 / 1
C	0 / 1	1 / 0	0 / 0

This matrix corresponds to the well known game rock-paper-scissors. The game is simple: rock beats scissors, scissors beats paper and paper beats rock. Strategy A corresponds to rock, strategy B to paper and strategy C to scissors. How (and why) does the population mix (XA, XB, XC) affect the average fitness of each agent? What about a different choice of payoff matrix?

Second scenario: variable strategies. In this scenario we explore a more interesting situation, where each agent can choose a different strategy at each interaction. We assume that the payoff matrix does not change during the simulation. We can define different behaviors that can be used by an agent to select a strategy; for example:

- *Fixed*: the agent always selects the same strategy (as in the first scenario above);
- *Random*: the agent choose a random strategy;
- *Tit-for-tat*: the agent which does not start the interaction plays the same strategy chosen by the agent starting the interaction.
- *Adaptive*: the agent which does not start the interaction plays the same strategy by its peer only if its peer has a higher payoff

To analyze this scenario, you are required to:

- Implement some of the behaviors above using PeerSim (you may want to define a protocol for each behavior, or to have a single parametric protocol in which you can set the behavior of each agent);
- Create a random graph, and assign to each agent one of the behaviors. Agents will not change their behaviors, therefore an agent set to use “Tit-for-tat” will always do that. The number of agents with each given behavior is a parameter of your simulation.
- Choose a payoff matrix (e.g., the rock-paper-scissors given above;) you may also analyze different payoff matrices of your choice (see below).

What is the “best” behavior, that is, the behavior which produces higher average fitness? Does changing some of the system parameters (e.g., the payoff matrix) result in a different choice of the best behavior?

Possible extensions. You may consider several extensions to the basic requirements described above. For example, you may decide to keep the interaction topology fixed, or change it at each round using the NewsCast algorithm already provided by PeerSim. In the first case, each agent keeps the same set of neighbors during the whole simulation run, while in the second case, each agent has a fresh set of neighbors at each cycle. Is the average fitness affected by having a static or dynamic interaction graph? Does the interaction topology (e.g., number of neighbors) affect the result?

You can propose and analyze your own extensions to the basic model, provided that the extensions are reasonable and you demonstrate that you understand what is going on. In general, we prefer a deeper understanding of a simple model rather than a weak understanding of a complex model. Nevertheless, this should be considered as an “open” project with a few basic requirements and some space for further experimentation.

2. *How and what to submit*

This project must be carried out individually. The project must be handed in electronically, by e-mail to marzolla@cs.unibo.it. The deadline for submission is **1 July 2013, at 23:59**. Please use your official university email address (@studio.unibo.it); the subject of your mail must be "CS Project 2012-2013"; you will receive a confirmation message (this may take a couple of days, please be patient).

IMPORTANT NOTE: it has been brought to my attention that some of the messages I send to @studio.unibo.it are considered spam by the spam filter. Please either disable the spam filter (you can do so with the webmail interface), or explicitly check your spam folder.

Your mail must contain a compressed archive (only .tar.gz and .zip formats are accepted; avoid .rar and other formats) containing the following items:

- Source code you have developed (it is not necessary to include the source code of PeerSim);
- A short paper, in pdf format, describing the experiments that have been performed and a detailed discussion of the results.

The source code must be adequately commented. The paper can be written either in italian or english, and should be organized like a technical paper with a title, abstract and (short) bibliography. You can use any document editor to write the paper, but only send the pdf document to us.
This project must be done individually. No sharing of source code or technical papers is allowed, either in part or full. However you can — and are actually encouraged to — discuss any issue with your fellow students and/or with the course instructors.
We suggest you to write no more than 16 pages using the LaTeX or Word/OpenOffice template from Springer LNCS available at http://www.moreno.marzolla.name/teaching/CS2012/ (this is not a strict requirement). You are required to put your full name, e-mail address and personal ID number (*numero di matricola*) on the paper, on each Java source file and in your mail.

3. *Grading policy*

These are the minimum requirements of your project:

- You must define a PeerSim model for the behavior propagation model described in this document. Extensions of the basic model are encouraged (see below for caveats).
- You must implement the model using the cycle-driven simulation engine of PeerSim. Your simulator must be configurable by means of the standard PeerSim configuration file.
- You must use the simulator to analyze at least the impact of the model parameters on the average fitness of the three agent types. You are free, if you want, to study additional metrics.
- The paper must describe the model (and the extensions you may have implemented), the simulation experiments and the results. You should explain the results, not just show them. The paper might describe implementation details if necessary, but keep in mind that we can look at the source code, so be concise.

Write a simple model, focus on a limited set of experiments and demonstrate that you actually understand the results. You have limited time, so use it wisely.
If you are interested in these topics (e.g., you want to build a better model, or you want to study similar systems), you are encouraged to talk to us when you will be looking for a thesis topic.

4. *References*

[1] David Easley Jon Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, Cambridge University Press 2010. In particular see Chapter 6, "Games"

[2] PeerSim: A Peer-to-Peer Simulator. Available at http://peeersim.sourceforge.net/