NetworGame 2.0 User Guide

(Version 04/07/2013)

Nodes having a major influence to break cooperation define a novel centrality measure: game centrality

Gabor I. Simko[1,2] and Peter Csermely[1,*]

1 Department of Medical Chemistry, Semmelweis University, Tűzoltó str. 37-47, H-1094 Budapest, Hungary,
2 Vanderbilt University, Nashville, TN USA
*Corresponding author, E-mail: *csermely.peter@med.semmelweis-univ.hu*

Summary

This User Guide describes the use of the NetworGame 2.0 program package, which is freely downloadable from the web-site: www.linkgroup.hu/NetworGame.php.

Table of Contents

1. Specification of the NetworGame 2.0 program

1.1. NetworGame 2.0 core functionality

The 2.0 version of the NetworGame program is an updated version of the NetworGame 1.0 version published in a preliminary conference report [1] also available in our web-site (www.linkgroup.hu/NetworGame.php). The 2.0 version utilizes our experiences gained with the 1.0 version. The 2.0 version of the NetworGame program is a console application for simulating repeated spatial 2-player games based on a configuration file. NetworGame 2.0 allows the adaptive choice of the simulation length and the number of simulations by using its built-in statistics module, as well as the manual specification of these parameters (see Section 1.4.: "Simulation termination conditions and number of simulations"). A configuration file contains the following simulation settings:

- network description,
- payoff matrix,
- simulation termination conditions and number of simulations,
- strategy update rule,
- initial strategies,
- payoff schema,
- settings of semi-synchronicity,
- settings for edge weights,
- initial seed for random number generator,
- printing commands.

1.2. Network description

The network description is the name of an external Pajek [2] file containing the network of nodes. NetworGame 2.0 handles all networks as undirected graphs, even if the original Pajek network was directed. Determination of edge weights will be described in Section 1.9.

1.3. Payoff matrix

The payoff is a 2x2 matrix defining the payoff values for the strategy combinations.

1.4. Simulation termination conditions and number of simulations

Let C_T denote the proportion of cooperating nodes at time T, and W_T denote an averaging window over the cooperation levels in a simulation with $length = 50$ (this window size proved to be a good compromise between the accuracy of the results and the speed of convergence). For example, W_{100} denotes the average of $C_{51} \dots C_{100}$.
Then, two types of termination conditions are supported:

1. termination at a fixed time step,
2. termination at time step T, such that $|W_T - W_{T-50}| < maxError$, i.e. W_T changed less than a specified threshold ($maxError$) in the last 50 steps.

Let $W_T[i]$ stand for the W_T of the i^{th} simulation in a number of parallel simulations, where the simulations are indexed $1 \dots n$.
The number of parallel simulations n can be:

1. set to a fixed value of simulations,

2. increased until the standard error for the sample mean over $W_T[1] \dots W_t[n]$ is below a specified threshold ($maxError/2$). Thus, the 95% confidence interval is within $sample\ mean\ \pm\ maxError$.

1.5. Strategies and strategy update rules

The NetworGame 2.0 version allows two strategies, cooperation or defection. Strategy update rules determine how players choose their strategy in the next round of the game. Strategy update rules are implemented as plug-ins making the development of custom strategy update rules possible. The NetworGame 2.0 program package comes with three standard strategy update rules as described in the following sections in detail.

Proportional strategy update rule plug-in

Player i copies the strategy of its neighbor j with probability:

$$p_{i,j} = \frac{G_j}{\sum_l G_l}, \quad l,j \in \{N(i) \cup i\}$$

where G_x is the average payoff received by player x in the previous round of the game. It is the user's responsibility not to use negative payoff values with this strategy update rule, because that would result in negative probabilities [3].

Best takes over strategy update rule plug-in

In the best takes over strategy update rule player i adopts the strategy of the agent selected from i and its neighbors, who had the highest payoff in the last round of the game.

Fermi-rule (logit replicator dynamics) plug-in

Player i randomly selects one of its neighbor j, and copies the previous strategy of j with a probability W, where

$$W = \frac{1}{1 + e^{-\frac{(G_j - G_i)}{K}}}$$

and G_i and G_j are the payoffs received by player i and j in the last round, and K is the noise (irrationality) in the system [4]. At $K = 0$ players are completely rational, at $K = \infty$ players are completely irrational. In our built-in implementation, $K = 0.1$ is fixed, but it is straightforward to modify this value in the source code.

The NetworGame 2.0 program allows the insertion of custom written strategy update rule plug-ins. For their description see Section 2.

1.6. Initial strategies

Initial strategies are generated according to their distribution specified in the simulation settings, where the user specifies only the probability of being a cooperator or defector in the initial round, and NetworGame generates the initial strategies for the nodes accordingly. Moreover, NetworGame 2.0 supports the manual specification of individual initial strategies for any nodes. This setting becomes the initial strategy of the specified node in the first step. Note, that in any round, the nodes play the same strategy against all their neighbors, and update their strategy only at the end of the round. Therefore, it is not possible to set different initial strategies against different neighbors.

1.7. Payoff schema

Three types of payoff schema are supported:

1. “intact” schema, the commonly used accumulated payoff schema,
2. “degree” (also called “averaging”) schema, where the accumulated payoff is divided by the degree of the given node [5,6],
3. “Luthi” schema, having results of replicator dynamics invariant to affine transformations of the payoff matrix as described in [7].

1.8. Settings of semi-synchronicity

In the program both synchronous update and semi-synchronous update [8] are supported. The probability p of semi-synchronous update gives the probability of updating the strategy of a node at the end of a round. We get the asynchronous update in the limit $p \to 0$ and synchronous update by $p = 1$, respectively.

1.9. Settings of edge weights

If the network has edge weights, the software can use them to determine the probabilities of games between neighboring nodes connected by the given edge. NetworGame 2.0 uses a linear interpolation to calculate the game probabilities, where edge weights corresponding to probability $p = 0$ and $p = 1$ should be set initially. Game probabilities within this edge weight region are linear interpolations between 0 and 1 set by the actual edge weight. Smaller edge weights than the pre-set minimum, or larger edge weights than the pre-set maximum have a game probability of 0 or 1, respectively.

1.10. Initial seed for random number generator

For random number generation, NetworGame 2.0 uses the Xorshift 128bit algorithm [9] with a period of 2^{128}-1. Xorshift is a powerful and verified pseudorandom number generator, which passes the Diehard Battery of Tests. The sequence of random numbers depends on a user-defined initial seed value. Importantly, the simulations produce the same results, if the seed value is unchanged.

1.11. Printing commands

Settings affecting the output of the program:

1. print the cooperation levels / standard deviances step-by-step
2. print the last strategy of the nodes
3. test nodes one-by-one for a given initial strategy (for example in calculating Game Centrality)
4. test edges one-by-one for a given initial strategy.

2. Strategy update rule plug-ins

The NetworGame 2.0 program allows the insertion of custom written strategy update rule plug-ins, but the construction of these plug-ins obviously needs programmer expertise.

Strategy update rules are stored in dynamic libraries (.dll in Windows, and .so in Linux) that are activated through their interfaces. All plug-ins must provide an init() method for initialization, and an update() method that returns the next strategy of a node. Optionally, plug-ins may contain a deinit() method that can be used for de-initialization.

3. References

1. Farkas IJ, Korcsmáros T, Kovács IA, Mihalik Á, Palotai R, Simkó GI, Szalay KZ, Szalay-Bekő M, Vellai T, Wang S, Csermely P (2011) Network-based tools in the identification of novel drug-targets. Science Signaling 4: pt3.
2. Batagelj V, Mrvar A (1998) Pajek - Program for large network analysis. Connections 21: 47–57.
3. Tomassini M, Luthi L, Giacobini M (2006) Hawks and doves on small-world networks. Phys. Rev. E 73: 016132.
4. Szabó G, Tőke C (1998) Evolutionary prisoner's dilemma game on a square lattice. Phys. Rev. E 58: 69–73.
5. Wu ZX, Xu XJ, Wang YH (2005) Does the scale-free topology favor the emergence of cooperation? http://arxiv.org/abs/physics/0508220.
6. Szolnoki A, Perc M, Danku Z (2008) Towards effective payoffs in the prisoner'sdilemma game on scale-free networks. Physica A 387: 2075–2082.
7. LuthiL, Tomassini M, Pestelacci E (2009) Evolutionary games on networks and payoff invariance under replicator dynamics. Biosystems 96: 213–222.
8. Saif MA, Gade PM (2009) Prisoner's dilemma with semi-synchronous updates: evidence for a first order phase transition. J. Stat. Mech. Theor. Exp. 7: P07023.
9. Marsaglia G (2003) Xorshift RNGs. J. Stat. Soft.8: 1-6.