Manual of LogTP

2024.7

1 Introduction

In a network formation game, there is a finite set of agents $N = \{1, 2, \cdots, N\}$ that are considering links with each other. Let $L = \{(i, j) \in N \times N | i < j\}$ be the set of links. For ease of notation, L denotes either the set of links or its cardinal, and we use ij instead of (i, j) to denote the link between nodes i and j. In the framework of unweighted networks, each pair of agents is either connected or not. An unweighted network on N then corresponds to a binary vector $g \in \{0, 1\}^L$, where $g_{ij} = 1$ if i and j are connected and $g_{ij} = 0$ otherwise for every $ij \in L$.

In our paper, we focus on the formation of weighted networks. That is, we associate each link $ij \in L$ with a variable $x_{ij} \in [0, 1]$, which can measure intensity, level of confidence, geographical distance, and so on. A weighted network can then be represented by a vector $x = (x_{ij})_{ij \in L} \in [0, 1]^L$. We denote the set of all possible weighted networks by $G = [0, 1]^L$. For all $ij \in L$, let $x_{-ij} \in [0, 1]^{L-1}$ denote the rest part of the network x while not taking x_{ij} into account. Each agent $i \in N$ has a utility function $u^i : G \to \mathbb{R}$. The concept of pairwise stability is introduced in Jackson and Wolinsky (1996) for unweighted networks and extended to a weighted version in Bich and Morhaim (2020).

Definition 1 (Pairwise stable network)**.** A network $x \in G$ is *pairwise stable* with respect to u if for all $ij \in L$,

1. for every $y \in [0, x_{ij})$, $u^i(y, x_{-ij}) \leq u^i(x)$ and $u^j(y, x_{-ij}) \leq u^j(x)$,
2. for every $y \in (x_{ij}, 1]$, $u^i(y, x_{-ij}) > u^i(x)$ implies $u^j(y, x_{-ij}) \leq u^j(x)$ and $u^j(y, x_{-ij}) > u^j(x)$ implies $u^i(y, x_{-ij}) \leq u^i(x)$.

We develop the algorithm LogTP that helps compute and select pairwise stable networks in weighted network formation games satisfying the following assumption.

Assumption 1. *For every agent $i \in N$ and every link $ij \in L$, the utility function $u^i(x_{ij}, x_{-ij})$ is continuously differentiable and concave with respect to x_{ij}.*

Our algorithm is derived from the logarithmic tracing procedure of Harsanyi and Selten (1988) based on the observation that a pairwise stable network in a weighted network formation game corresponds to a Nash equilibrium of a non-cooperative game. When the form of the utility functions varies, we recommend using different versions of Matlab software.

- For general problems, we recommend the software in folder "LogTPc", introduced in Section 2.
- For a special type of problem with multi-affine utility functions, we recommend the version in folder "LogTPm", introduced in Section 3.
- If the problem has a sparse structure, an accelerated version in the folder "ALogTP" is available. i.e. for each agent $i \in N$, the utility function u^i is only influenced by x_{ij} with $j \in N \backslash \{i\}$. A brief introduction is presented in Section 4.

2 LogTPc

LogTPc is the basic version of software that applies to all network formation games satisfying Assumption 1. In this section, we summarize the rough idea of LogTPc and illustrate how to apply it to a new model. For a detailed mathematical background, please refer to our paper.

Let $p = (p^i_{ij}, p^j_{ij}, p_{ij})_{ij\in L} \in G^3$, $\sigma = (\sigma^i_{ij}, \sigma^j_{ij}, \sigma_{ij})_{ij\in L} \in G^3$ and $\eta > 0$ be given parameters. For $s = (s^i_{ij}, s^j_{ij})_{ij\in L} \in G^2$ and $\alpha = (\alpha_{ij})_{ij\in L} \in G$, let $q : G^3 \to G$ be a mapping given by $q(s, \alpha) = (q_{ij}(s, \alpha))_{ij\in L}$ where

$$q_{ij}(s, \alpha) = \alpha_{ij} s^i_{ij} + (1 - \alpha_{ij}) s^j_{ij}. \tag{1}$$

For $t \in [0,1]$, we define the function $\alpha^* : G^2 \times [0,1] \to G$ by

$$\begin{aligned} \alpha^*(s,t) &= (\alpha^*_{ij}(s,t))_{ij \in L}, \\ \alpha^*_{ij}(s,t) &= \frac{1}{2A}(A + B - \sqrt{(A+B)^2 - 4AB\sigma_{ij}}), \end{aligned} \tag{2}$$

where $A = ts^i_{ij} - ts^j_{ij} + (1-t)p^i_{ij} - (1-t)p^j_{ij}$ and $B = (1-t)\eta$. The solution set of the following system of equations contains a differentiable path that starts from the level of $t = 0$ and intersects the level of $t = 1$.

$$t\partial u^i\big(s^i_{ij}, q_{-ij}(s, \alpha^*(s,t))\big) + (1-t)\big(\partial u^i(s^i_{ij}, q_{-ij}(p)) + \eta(\frac{\sigma^i_{ij}}{s^i_{ij}} - \frac{1-\sigma^i_{ij}}{1-s^i_{ij}})\big) = 0, \tag{3}$$

$$t\partial u^j\big(s^j_{ij}, q_{-ij}(s, \alpha^*(s,t))\big) + (1-t)\big(\partial u^j(s^j_{ij}, q_{-ij}(p)) + \eta(\frac{\sigma^j_{ij}}{s^j_{ij}} - \frac{1-\sigma^j_{ij}}{1-s^j_{ij}})\big) = 0, \tag{4}$$

where $\partial u^i(s^i_{ij}, q_{-ij}(s, \alpha^*(s,t)))$ and $\partial u^i(s^i_{ij}, q_{-ij}(p))$ are the partial derivatives of $u^i(y, q_{-ij}(s, \alpha^*(s,t)))$ and $u^i(y, q_{-ij}(p))$ at $y = s^i_{ij}$. If $\bar{s} = (\bar{s}^i_{ij}, \bar{s}^j_{ij})_{ij \in L} \in G^2$ is the intersection point of the path and the level of $t = 1$, then $(\min\{\bar{s}^i_{ij}, \bar{s}^j_{ij}\})_{ij \in L} \in G$ provides a pairwise stable network of the network formation game. Let $H : G^2 \times [0,1] \to \mathbb{R}^{2L}$ be the mapping given by the left-hand sides of (3) and (4).

In LogTPc, we apply the predictor-corrector method of Allgower and Georg (1990) to numerically trace the path. We show its framework in Algorithm 1. The following Matlab code files play an important role in implementing LogTPc.

- main.m: the main program of LogTP, including the parameter settings and the implementation of the predictor-corrector method.
- links.m: this function returns a $L \times 2$ matrix, each of whose rows corresponds to a link. For example, the row with elements 1, 2 denotes the link between agent 1 and 2.
- def.m: Given $(s,t) \in G^2 \times [0,1]$, this function returns a $N \times N$ matrix whose (i,j)-th element equals $\partial u^i(s^i_{ij}, q_{-ij}(s, \alpha^*(s,t)))$.
- def_p.m: Given $(s,t) \in G^2 \times [0,1]$, this function returns a $N \times N$ matrix whose

(i,j)-th element equals $\partial u^i(s^i_{ij}, q_{-ij}(p))$.

- homof.m: Given $(s,t) \in G^2 \times [0,1]$, this function returns $H(s,t)$.
- ahomof.m: Given $(s,t) \in G^2 \times [0,1]$, this function computes the values of the left-hand sides of system (5).

When applying LogTPc to a new problem, adjustments to the codes are required. One has to adjust the parameters in "main.m" specific to the problem. For example, the social values in the connections model of Jackson and Wolinsky (1995) or the efforts in Bramoullé and Kranton (2007). The formulas in "def.m" and "def_p.m" to compute the partial derivatives of the utility functions should also be updated accordingly.

3 LogTPm

The LogTPm applies to a special type of network formation games with multi-affine utility functions. They are called mixed extension of network formation games, analogous to mixed extension of non-cooperative games. Given a network $x \in G$, $x_{ij} \in [0,1]$ is interpreted as the probability that link ij is built, for every $ij \in L$. The probability that an unweighted network $g \in \{0,1\}^L$ forms equals to

$$P_g(x) = \prod_{ij \in L} (x_{ij} g_{ij} + (1 - x_{ij})(1 - g_{ij})).$$

Each agent $i \in N$ maximizes the expected payoff

$$u^i(x) = \sum_{g \in \{0,1\}^L} P_g(x) v^i(g), \tag{7}$$

where $v^i(g)$ is the payoff agent i receives from the unweighted network g. We can solve unweighted network formation games by studying its mixed extension since the unweighted pairwise stable networks still satisfy pairwise stability after the extension.

It follows from (7) that

$$\partial u^i(y, x_{-ij}) = \sum_{g \in \{0,1\}^L} v^i(g)(2g_{ij} - 1) \prod_{k\ell \in L \setminus \{ij\}} (x_{k\ell} g_{k\ell} + (1 - x_{k\ell})(1 - g_{k\ell})).$$

That is, given $x \in G$, the partial derivatives of the utility functions are uniquely determined by the payoff the agents receive from the unweighted networks. In this version of software, we first compute the payoffs $v^i(g)$. To do so, we require the following two code files.

- graphs.m: this function returns a $2^L \times L$ matrix, each of whose rows corresponds to an unweighted network that may form.
- values.m: this function returns a $2^L \times N$ matrix, whose i-th row gives the payoff vector of all agents in the unweighted network given by the i-th row of the output of "graphs.m".

With the payoffs $v^i(g)$, we can easily derive the partial derivatives of the utility functions. The rest part of the code is the same as the basic version. When applied to a new problem, one only has to adjust the formulas in "values.m" apart from necessary adjustments to the parameters.

4 ALogTP

Inspired by the insightful approach of Leung (2020), we develop ALogTP, an accelerated version of LogTP, that applies to problems with a sparse structure. i.e. For each agent $i \in N$, the utility function u^i is only influenced by x_{ij} with $j \in N \setminus \{i\}$. Its basic idea is to first figure out the links that are sure to be absent or built. These links decompose the whole network into smaller ones, to which we then apply LogTP.

A link $ij \in L$ is surely to be absent if $\sup_{x \in G} \frac{\partial u^i(y, x_{-ij})}{\partial y} \leq 0$ or $\sup_{x \in G} \frac{\partial u^j(y, x_{-ij})}{\partial y} \leq 0$ and it is surely to be built if $\inf_{x \in G} \frac{\partial u^i(y, x_{-ij})}{\partial y} > 0$ and $\inf_{x \in G} \frac{\partial u^j(y, x_{-ij})}{\partial y} > 0$. In the implementation of ALogTP, we require the following code files:

- main.m: the main program of ALogTP, including the parameter settings, decomposition of networks, and the implementation of LogTP to the subnetworks.

- robust_links.m: this function returns two $N \times N$ matrix M and D that help figure out the links surely to be absent or built. A link $ij \in L$ is surely to be absent if $M_{ij} = 0$ or $M_{ji} = 0$. This link is sure to be established if $D_{ij} = 0$ and $D_{ji} = 0$.

- combine.m: this function combines M and D and returns a new $N \times N$ matrix $\tilde{D}$. For $ij \in L$, $\tilde{D}_{ij} = 1$ if the link is surely to be absent or built and $\tilde{D}_{ij} = 0$ otherwise. With this matrix, we can decompose the network into smaller ones (via the Matlab function "conncomp").

- search_subproblem.m: this function generates a matrix of N columns from the output of the matlab function "conncomp". The number of its rows equals the number of subnetworks. Each of its rows indicates the agents involved in the corresponding subnetwork.

- solution2.m & path_following.m: in these two functions we apply LogTP to the subnetworks.

Similar to the basic version, one has to adjust the parameters in "main.m" and the formulas in "def.m" and "def_p.m" when applying ALogTP to a new problem.

References

Allgower, E. L. and Georg, K. (1990). *Numerical Continuation Methods: An Introduction*. New York: Springer.

Bich, P. and Morhaim, L. (2020). On the existence of pairwise stable weighted networks. *Mathematics of Operations Research*, 45:1393–1404.

Bramoullé, Y. and Kranton, R. (2007). Public goods in networks. *Journal of Economic Theory*, 135:478–494.

Harsanyi, J. C. and Selten, R. (1988). *A general theory of equilibrium selection in games*. Cambridge: MIT Press.

Jackson, M. O. and Wolinsky, A. (1995). A strategic model of social and economic networks. Technical report, Northwestern University, Center for Mathematical Studies in Economics.

Jackson, M. O. and Wolinsky, A. (1996). A strategic model of social and economic networks. *Journal of Economic Theory*, 71:44–74.

Leung, M. P. (2020). Equilibrium computation in discrete network games. *Quantitative Economics*, 11:1325–1347.

Algorithm 1: LogTPc

Require:

$\epsilon > 0$, which determines the termination of the algorithm;
$\alpha > 0$, which determines the velocity;
$\delta_0 > 0$, which determines the starting velocity;
$p = (p_{ij}^i, p_{ij}^j, p_{ij})_{ij \in L} \in G^3$, the prior;
$\sigma = (\sigma_{ij}^i, \sigma_{ij}^j, \sigma_{ij})_{ij \in L} \in G^3$ and $\eta > 0$, weights of the logarithmic penalty terms;
$l_0 = (0, 0, \cdots, 0, 1) \in \mathbb{R}^{2L+1}$, the initial prediction direction;
$k = 0$, the number of iterations;

Ensure:

A pairwise stable network;

1: Initialization: Compute the unique solution $s_0 \in G^2$ of the equation $H(s, 0) = 0$. Let $z_0 = (s_0, 0) \in G^2 \times [0, 1]$.

2: Predictor step: Set $z_k' = z_k + \delta_k l_k$.

3: Velocity test and corrector step:

Let $t(z)$ denote the value of t at the point $z \in G^2 \times [0, 1]$.

Make sure z_k' is feasible: if $t(z_k') < 0$ or $t(z_k') > 1$, set $\delta_k = 0.9\delta_k$ and return to the predictor step.

Make sure z_k' is a good guess: if $\|H(z_k')\| > \alpha$, set $\delta_k = 0.9\delta_k$ and return to the predictor step.

Corrector step: solve the following system of equations starting from z_k' and obtain z_{k+1}.

$$\begin{aligned} H(z) &= 0, \\ l_k^T(z - z_k') &= 0. \end{aligned} \tag{5}$$

4: If $t(z_{k+1}) > 1 - \epsilon$, apply it as the starting point to solve the equations

$$\begin{aligned} H(z) &= 0, \\ t(z) &= 1. \end{aligned} \tag{6}$$

The result provides a pairwise stable network.

Otherwise, set $\delta_{k+1} = \delta_0$, $l_{k+1} = \frac{z_{k+1} - z_k}{\|z_{k+1} - z_k\|}$, $k = k + 1$ and return to the predictor step.