Manual of LogTP

2024.8

LogTP is an algorithm in MatLab software for computing pairwise stable networks. In this manual, we introduce the concept of pairwise stability and the framework of our algorithm in Section 1. In the following sections, we show more technical details: in Section 2, we introduce the basic version of LogTP for problems with concave and differentiable utility functions. In Section 3, we present a version for mixed-extended problems, where agents all have multi-linear utility functions. In Section 4, we show an accelerated version of LogTP that applies to problems where only direct connections matter.

1 Introduction

1.1 Pairwise stability

In a network formation game, there is a finite set of agents $N = \{1, 2, \cdots, N\}$. The network relations among them are represented by an undirected graph, where the nodes represent the agents and the links stand for pairwise relations. 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 is 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 $u^i(y, x_{-ij}) \leq u^i(x)$.

1.2 Basic idea of LogTP

We develop the algorithm LogTP that computes and selects 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}.*

LogTP is developed 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. Then we devise a logarithmic tracing procedure for this non-cooperative game to select a pairwise stable network. Here we provide an outline of LogTP. For a detailed mathematical background, please refer to Sections 3 and 4 of 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 $B = (1-t)\eta$. It satisfies that $q(s, \alpha^*(s,1)) = (\min\{s^i_{ij}, s^j_{ij}\})_{ij\in L}$, for all $s \in G^2$.

Let $H : G^2 \times [0,1] \to \mathbb{R}^{2L}$ be a mapping of $(s,t) \in G^2 \times [0,1]$ given by the left-hand sides of (3) and (4).

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

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

where $\partial u^i(s^i_{ij}, q_{-ij}(s,\alpha^*(s,t)))$ and $\partial u^i(s^i_{ij}, q_{-ij}(p))$ represent the partial derivatives of $u^i(y, q_{-ij}(s,\alpha^*(s,t)))$ and $u^i(y, q_{-ij}(p))$ at $y = s^i_{ij}$. Similar notations are used for $\partial u^j(s^j_{ij}, q_{-ij}(s,\alpha^*(s,t)))$ and $\partial u^j(s^j_{ij}, q_{-ij}(p))$.

For generic $\sigma \in G^3$, the zero set of H contains a unique differentiable path that starts from the level of $t=0$ and intersects the level of $t=1$. 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 $(\min\{\bar{s}^i_{ij}, \bar{s}^j_{ij}\})_{ij\in L} \in G$ is a pairwise stable network of the network formation game. Usually, we derive different pairwise stable networks when the parameters $p \in G^3$, $\sigma \in G^3$ and $\eta > 0$ vary.

In LogTP, we apply the predictor-corrector method of Allgower and Georg (1990) to numerically trace the path. We show its framework in Algorithm 1. 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.

Algorithm 1: LogTP

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 $H(s, 0) = 0$ to determine the starting point of the homotopy path. 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 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.

- If the problem has a sparse structure, an accelerated version in the folder "comp/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 our algorithm for problems where agents all have concave and differentiable utility functions. In this section, we show its technical details and illustrate how to apply it.

- **main.m**: the main program of LogTP, including parameter setting and the implementation of the predictor-corrector method.

  Parameter settings

  $\epsilon > 0$, which determines the termination of the algorithm;

  $\alpha > 0$, which determines the velocity in the predictor step;

  $\delta_0 > 0$, which determines the starting velocity;

  $p \in G^3$, the prior;

  $\sigma \in G^3$ and $\eta > 0$, weights of the logarithmic penalty terms;

  N, number of players;

  L, number of links;

  $M = 2L$, dimension of variables

  and other parameters in the model to which LogTPc is applied. **These are all global variables!**

  Predictor-corrector method

  We conduct the algorithm shown in Algorithm 1. We initialize the algorithm with Matlab functions "init.m" and "u0.m". In the iterations, we compute the value of the mapping H with Matlab functions "F.m", "def.m" and "homof.m". The equations in (5) and (6) are solved with Matlab function "fsolve".

- **link.m**: to show the set of all possible links.

Input: N, number of players.

Output: a $L \times 2$ matrix lin (**global variable**) whose each row represents a possible link. For example, the row given by (i, j) represents the link between agent i and j. lin can be interpreted as a mapping from $\{1, 2, \cdots, L\} \to L$, which sorts the links.

- **init.m**: to search for the starting point of the homotopy path (solve the equation $H(s, 0) = 0$). Notice that when $t = 0$, the equation $H(s, 0) = 0$ consists of $2L$ independent equations (given in "u0.m"). We solve them in sequence with the Matlab function "fsolve".

  Input: none.

  Output: a vector $s_0 \in G^2$ such that $H(s_0, 0) = 0$.

- **u0.m**: to compute the elements of the mapping $H(\cdot, 0) : G^2 \to \mathbb{R}^{2L}$. In "init.m" we apply the Matlab function "fsolve" to search for the zeros of the mapping given by "u0.m".

  Input: $x \in [0, 1]$, link strength; $i \in \{1, 2, \cdots, L\}, j \in \{1, 2\}$, index for link and player (in the sense of "lin").

  Output: the $2i - 2 + j$ -th element of the mapping H at $t = 0$. (corresponding to the link given by the i-th row in lin and agent $lin(i, j)$)

- **F.m**: to compute the network $q(s, \alpha^*(s, t)) \in G$ involved in the homotopy mapping H.

  Input: $t \in [0, 1]$; $s \in G^2$, the vector of favorite strengths; $p \in G^3$, the prior;

  Output: the network $q(s, \alpha^*(s, t)) \in G$ derived with formula (1) and (2).

- **def.m**: to compute the partial derivatives of the utility functions with respect to a given network.

  Input: $s \in G^2$, the vector of favorite strengths; $x \in G$, a given network;

Output: a $N \times N$ matrix whose (i,j) and (j,i) -th element equals to $\partial u^i\left(s_{ij}^i, x_{-ij}\right)$ and $\partial u^j\left(s_{ij}^j, x_{-ij}\right)$, respectively.

- **homof.m**: to compute the value of the homotopy mapping H. The partial derivative terms are computed with the Matlab function "def.m" with the input variable x set as $q(s, \alpha^*)$ and $q(p)$.

  Input: $(s,t) \in [0,1]^{2L+1}$.

  Output: $H(s,t) \in \mathbb{R}^{2L}$.

- **ahomof.m**: to compute the value of the system (5). This system of equations is solved by Matlab function "fsolve" in the corrector step.

  Input: $(s,t) \in [0,1]^{2L+1}$.

  Output: the value of system (5), in $\mathbb{R}^{2L+1}$.

When applied to a new problem, one has to adjust the formulas in "def.m" apart from necessary adjustments to the parameters in "main.m".

3 LogTPm

The LogTPm applies to a special type of network formation games with multi-affine utility functions. They are called mixed extensions of network formation games, analogous to mixed extensions 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 their mixed extensions since the unweighted pairwise stable networks still satisfy pairwise stability after the extension.

We exploit the structure of mixed-extended problems and adjust the basic version of LogTP accordingly. It follows from (7) that

$$\partial u^i(y, x_{-ij}) = \sum_{g\in\{0,1\}^L} v^i(g)(2g_{ij} - 1) \prod_{k\ell\in L\setminus\{ij\}} (x_{k\ell}g_{k\ell} + (1 - x_{k\ell})(1 - g_{k\ell})). \tag{8}$$

For $ij \in L$, the partial derivatives with respect to x_{ij} are determined by $x_{-ij} \in [0, 1]^{L-1}$ and the payoffs the agents yield from the unweighted networks. Therefore, we denote the partial derivative by $\partial u^i(x_{-ij})$ for short. We compute the partial derivatives with a different method from LogTPc.

- **graphs.m**: to show the set of all possible unweighed networks.

  Input: L, the number of links.

  Output: a $2^L \times L$ matrix gra whose each row represents a possible unweighted network. **(global variable)**

- **values.m**: to compute the payoff vectors in each possible network.

  Input: gra, a $2^L \times L$ matrix recording all possible networks derived from "graphs.m".

  Output: a $2^L \times N$ matrix Va whose each row corresponds to the payoff vector in an unweighted network. **(global variable)**

- **def.m**: to compute the partial derivatives of the utility functions with respect to a given network (with formula (8)).

  Input: $x \in G$, a given network;

  Output: a $N \times N$ matrix whose (i, j) and (j, i) -th element equals to $\partial u^i\big(x_{-ij}\big)$ and $\partial u^j\big(x_{-ij}\big)$, respectively.

The rest parts of the codes are the same as LogTPc and we omit their introductions here. Differently, the codes in "def.m" need not be adjusted when applying LogTPm

to a new model. One only has to adjust the formulas in "values.m" that compute the payoff the agents receive from the unweighted networks.

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 $j \in N\backslash\{i\}$. That is, the agents only value direct connections. The basic idea of ALogTP is to first figure out the links that are sure to be absent or built, which we call robustly absent or built links. These links decompose the whole network into smaller ones, to which we then apply LogTPc (or LogTPm).

We introduce the criteria of robust links. A link $ij \in L$ is robustly 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 robustly built if $\inf_{x\in G} \frac{\partial u^i(y,x_{-ij})}{\partial y} > 0$ and $\inf_{x\in G} \frac{\partial u^j(y,x_{-ij})}{\partial y} > 0$.

We show more technical details of ALogTP.

- **main.m**: the main program of ALogTP.

  Parameter settings

  N, number of players;

  L, number of links;

  lin, the set of all possible links derived from "link.m";

  and other parameters in the model to which ALogTP is applied. (we take the public good provision model of Bramoullé and Kranton (2007) as an example in the codes presented) **(global parameters)**

  Outline

  We first figure out the robust links in the problem with the functions "robust_links.m" and "combine.m". Then we decompose the network into smaller ones with the Matlab function "conncomp" and figure out the agents involved in the subnet-

works with "search_subproblem.m". Finally, we apply LogTP to the subnetworks via functions "solution.m" and "path-following.m".

- **robust_links.m**: to figure out the robustly absent and built links.

  Input: parameters in the model. Take the public provision model of Bramoullé and Kranton (2007) as an example. (in Section 5.4 of our paper) $e \in \mathbb{R}_+^N$, the effort vector; $d > 0$, cost for links.

  Output: two $N \times N$ matrice M and D. A link $ij \in L$ is robustly absent if $M_{ij} = 0$ or $M_{ji} = 0$ and robustly built if $D_{ij} = 0$ and $D_{ji} = 0$.

- **combine.m**: to summarize the results of "robust_link.m" and distinguish between the non-robust links and the robust ones.

  Input: the matrice M and D derived from "robust_link.m".

  Output: a $N \times N$ symmetric matrix $\tilde{D}$. For $ij \in L$, $\tilde{D}_{ij} = 0$ if the link is robustly absent or built and $\tilde{D}_{ij} = 1$ otherwise. The matrix $\tilde{D}$ can be viewed as the adjacency matrix of an undirect graph, whose connected components provide the subproblems. We figure out the connected components via the Matlab function "conncomp". (The input of "conncomp" is the adjacency matrix of a graph and the output is a vector S in ${\mathbb{N}_+}^N$: agent $i \in N$ lies in the S_i -th connected component of the graph.)

- **search_subproblem.m**: to figure out the agents involved in each subproblem. A subproblem contains the agents in a connected component of the graph given by $\tilde{D}$ and the agents connected to them via a robustly built link.

  Input: a $1 \times N$ vector S, the output of "conncomp($\tilde{D}$)"; D, the $N \times N$ matrix derived from "robust_link.m"; num_S, the number of connected conponents.

  Output: *group*, a matrix of N columns. Each row of the matrix corresponds to a subnetwork. Its (i, j)-th element equals 1 if agent j is contained in the i-th subnetwork and 0 otherwise.

- **solution.m**: to figure out a pairwise stable subnetwork with LogTP. To do so, we first generate a vector $Link \in G$ that records the structure of the subnetwork. Precisely, the strengths of the links not included in the subnetwork equal zero. (In problems with a sparse structure, it makes no difference when computing the utility functions and their partial derivatives) The robustly absent or built links have strength 0 or 1, respectively. The strengths of the non-robust links in the subnetwork are set as -1. The number of non-robust links is denoted by num. The subproblem is then tackled with "path-following.m", the main program of LogTP, with input $Link \in G$.

  Input: $group$, a $1 \times N$ vector that records players in a subnetwork.

  Output: $sol \in G$, a pairwise stable subnetwork; num, the number of non-robust links in this subnetwork.

- **insert.m**: to combine the strengths of the non-robust links and robust ones.

  Input: $x \in [0,1]^{num}$, a vector recording the strengths of the non-robust links; $Link \in G$, a vector that records the structure of the subnetwork.

  Output: a network in G derived from $Link$ by replacing the -1s with the elements in x.

- **path-following.m**: the main program of LogTP, which is applied to the subnetworks. Here we handle a homotopy system of $2num$ dimensions, corresponding to the non-robust links.

  Input: $Link \in G$, a network records the structure of the subnetwork.

  Output: $PS \in G$, a pairwise stable subnetwork that we derive from $Link$ by replacing the -1's with the results of LogTP; num, number of the non-robust links.

There are a few adjustments to the codes of LogTP since we have to consider the structure of the subnetwork. That is, we need to combine the strengths of the non-robust links and the robust ones with the Matlab function "insert.m" before the computations. Therefore,

we add an input parameter $Link \in G$ for the functions in LogTP. There are no major changes to the codes except for those in "def.m". We adjust the output of the function "def.m". Now it returns a $num \times N$ matrix whose (i, j) -th element represents the partial derivative of agent j's utility function with respect to the i -th non-robust link in the subnetwork.

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.