IIT Documentation

Open Matlab, go to program folder. Type iit.

The input GUI:

Network definition (top left corner):

Networks can be defined either by (1) selecting mechanisms and defining their connectivity, (2) by the TPM (transition probability matrix). This can be selected in the Network Definition Method List.

Important: Select the correct method BEFORE uploading a TPM or connectivity matrix.

Note: If the network is sparse, uploading a connectivity matrix on top of the TPM will speed the calculation immensely.

Everything else should be fairly self-explanatory.

Options (top left corner):

Starting the GUI, the currently preferred options to calculate Phi are given.

- Complex: either search for complex in a potentially larger system or calculate whole system only.
- Distances should be calculated using earth-mover's distance (EMD) (small phi function and big phi algorithm options).
- External nodes: if Complex is selected, then also subsets of the system are evaluated. The nodes outside the subset can either be frozen to their past state, disconnected (Changing the mechanisms of the elements inside!!), or noised (to max entropy)
  We now consider freezing of outside variable as the correct procedure for external nodes.
- For "large" or non-sparse networks (> 6-10 nodes) the EMD algorithm might not be feasible. Choosing L1norm/2 as the Big Phi Algorithm already reduces computation time considerably while still being a quite accurate approximation, if that is not enough also chose L1norm/2 for small phi. It's way faster, but might lead to quite different results.

Computation Speed options:

- Only active if searching for complexes. Networks that are not strongly connected cannot be complexes. This can be checked in advance, skipping subsets that are not strongly connected. If you are interested in the concepts of the subsets even if there big phi MIP = 0, then choose "No".
- Not using the par-for loop can be better if many instances of the program are run at the same time. But usually one should use the par-for.

Code Structure:

`iit.m` GUI

`iit_run` Main function

% calculate small phi and determine concepts for each subsystem or only the whole system

(-> `big_phi_all.m` FOR loop over subsystems) only if op_ complex = 1

->`big_phi_comp_fb.m` FOR loop over numerators

-> `phi_comp_ex.m` FOR loop over denominators and choosing which denominator has max phi with or without normalization

-> `phi_comp_bf.m/phi_comp_bORf.m` calculate distance between whole and partitioned distributions

-> `comp_pers_cpt.m` calculate distributions

% calculate big phi for each subsystem or the whole system only and determine complex

-> `complex_search.m` FOR loop over subsystems

-> `MIP_search_reentry.m` FOR loop over all possible partitions of the subsystem

-> `phi_comp_ex_unidir.m` FOR loop over denominators and choosing which denominator has max phi with or without normalization for unidirectionally partitioned systems

-> `phi_comp_bORf.m` calculate distance between whole and partitioned distribution for unidirectionally partitioned systems

-> `comp_pers_cpt.m` calculate distributions

Network Struct:

The system properties are saved in the network struct that is passed from function to function.

`network.connect_mat`	The NxN binary connectivity matrix
`network.options`	An array of integers which represent the options (see Options section)
`network.nodes`	A struct array of node structs (see Node Struct)
`network.num_nodes`	The number of nodes, N, in the full system.
`network.tpm`	A States X Node (2N X N) transition probability matrix for the entire system. This matrix does not need to be State X State because of conditional independence.
`network.full_system`	An array of the numbers 1:N.
`network.num_subsets`	The number of subsets in the system, 2N.
`network.current_state`	An integer array of length N that is the current state of the system. This is only used when the software is set to only compute over one state as opposed to taking an average of all states.
`network.past_state`	An integer array of length N that is the past state of the system. This is only used when the software is set to only compute over one state as opposed to taking an average of all states.
`network.num_states`	Total number of states of the system. This is equal to the product of the number of states of each node. The software is almost totally able to handle more than binary nodes, but there are still some changes that need to be made.
`network.noise`	A number in the range [0,.5] that adds noise into the output of every node. That is, the deterministic output will be correct with probability 1 - noise.