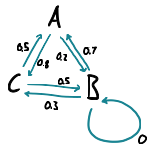
Info Propagator Algorithm:

Program's Process:

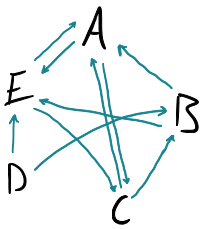
1. Initialize Adjacency matrix (A_{ij}) on N nodes without allowing self-referential edges (ie $\nexists$ loops, simple directed graph) and normalizing outgoing edge weights to 1

1.1 Uniform random initialization:
i.e. all values are random (0,1), then normalized (this is the most general case)



	A	B	C
A	0	0.2	0.8
B	0.7	0	0.3
C	0.5	0.5	0

1.2 Sparse random initialization:
i.e. Yields only a set number of connections upon initiation (which shall generally have equal value)
e.g. with 2 connections per node:



	A	B	C	D	E
A	0	0	½	0	½
B	½	0	0	0	½
C	½	½	0	0	0
D	0	½	0	0	½
E	½	0	½	0	0

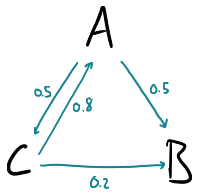
2. After initialization, 'information nuggets' are seeded onto the network nodes. The 'information' or value per nugget is variable to test various rates of network adaptation.

2.1 Either randomly[1], whereupon n nuggets of value v are distributed to a random node on the network,

2.2 or as a function of the diversity of connection:

Probability of node receiving nugget $= \frac{e^{-\beta\sigma}}{\sum_{i}^{N} e^{-\beta\sigma_i}}$, where σ is the standard deviation of a node's outgoing edges, and β we may call the correlation coefficient, as it merely serves to determine how correlated the delivery of a nugget is with its diversity of connections. Though the σ^2 dependence conceals a non-linear relation to the diversity of the node's edges, it still serves as a good metric for diversity of connection in the relevant range of values, i.e. $\sum_i = 1$, and allows for the familiar mechanics of the partition function as an analogue to energy. Note this means σ is NEGATIVELY correlated with diversity of connection edges

e.g.



In this network (outgoing B edges not shown) despite A ≠ C both having 2 non-zero edges, A has a larger σ^2, (and thus $e^{\beta\sigma}$) and is thus is more likely to receive the 'information' nugget.

$$\frac{e^{-\beta\sigma(A)}}{e^{-\beta\sigma(A)} + e^{-\beta\sigma(B)} + e^{-\beta\sigma(C)}} > \frac{e^{-\beta\sigma(C)}}{e^{-\beta\sigma(A)} + e^{-\beta\sigma(B)} + e^{-\beta\sigma(C)}}$$

3. Propagate Information:

After information is seeded, it disseminates from the seeded nodes as the product of the nugget (and thus seed node) value and their outgoing edges.

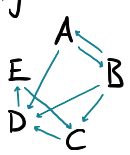
Value of nodes after propagation step:

$$n_j = n_i w_{ij}$$

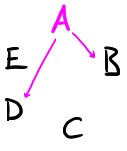
w_{ij} = edge weight from node i to node j

n_i = Value of node i

Example propagation:
Pre-existing Network Structure:



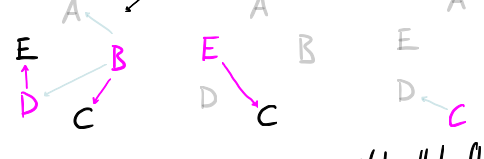
A is seeded w/info:



Presently if B is selected first, it will pass info to D, otherwise not

B does not pass information to A, as A has already passed information

C's only connection has already passed info, so no info is further passed



This process of propagation from nodes with non-zero value is repeated until either all nodes have been visited or there are no remaining connections above a cutoff threshold[2]. Note that after a node receives info in a propagation step, the node may no longer receive information, e.g.

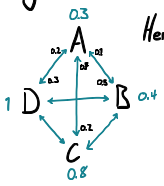
Here, as A is directly connected to C, B's outgoing edge to C will not pass information because C has already received information during that same propagation step. D will however receive info from both B & C during the next propagation step, as both edges transfer from their respective nodes during the same step.

[1] which then should eventually yield a "smoke" network (where the value of info is passed by $\prod$ connections · source value if adaptation is optimal

[2] Actually product of connection (edge) & node value

4. Update Edge Weights

Edges are reweighted by adding the product of their value & their connected node, a change which is then normalized according to the sum of a node's incoming edges.



Here consider reweighting weights associated with A:

A→B: 0.5 + (0.5 · 0.3) = 0.65
A→C: 0.2 + (0.2 · 0.3) = 0.26
A→D: 0.3 + (0.3 · 0.3) = 0.39

Former Values (May sum > 1) — These values will be normalized to the sum of a nodes incoming edges

Node Value, Edge Value

B → A: 0.1 + (0.1 · 0.4) = 0.14
C → A: 0.7 + (0.7 · 1) = 1.4
D → A: 0.2 + (0.2 · 0.8) = 0.36

All these (incoming) values are then normalized

= 0.074
= 0.74
= 0.19

Those sum to 1 initially, and are subsequently renormalized to 1.

$\Sigma = 1.9$

5. Add Node end values to history, reset all node values to 0. Record Adjacency matrix, but do not reset values.

6. Repeat Steps 2-5 for every 'Run' of the algorithm.