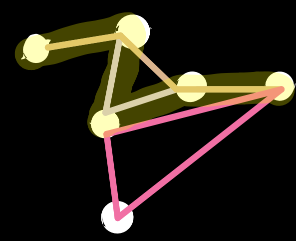
Graph T_5



1) Paths, diffusion, and navigation

1.1) path lengths

→ shortest path length (Dijkstra's algo)

→ local efficiency

→ global efficiency

1.2) diffusion processes on graphs

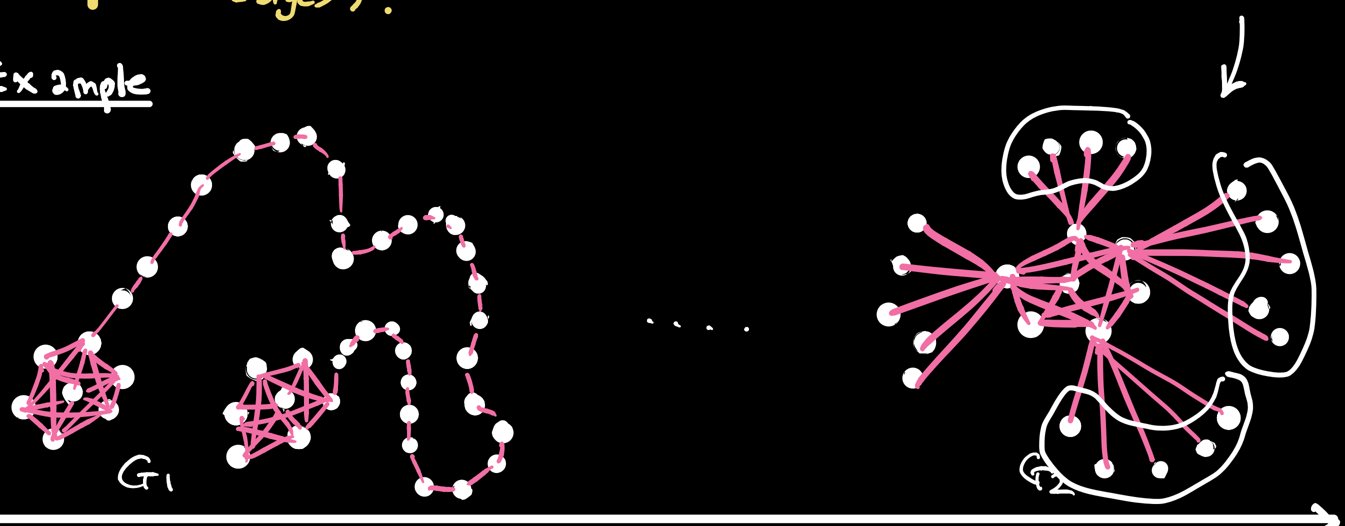
→ diffusion efficiency

"

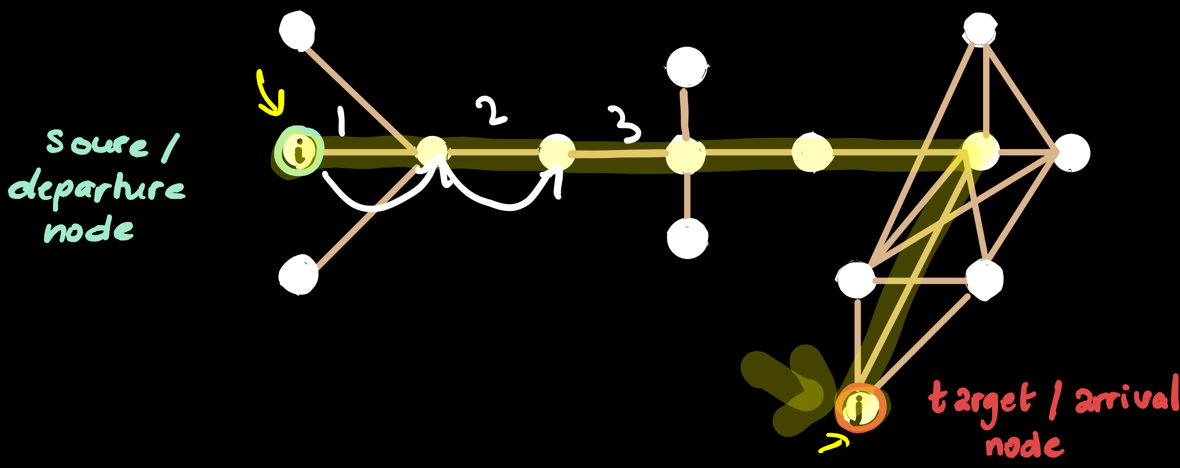
* How the topological properties of graphs constrain the way in which information flows through the graph?
* which specific graph properties can be used to gain insight into information processing by graph units (nodes / edges)?

"

<u>Example</u>



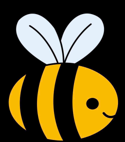
information flow Efficiency measures



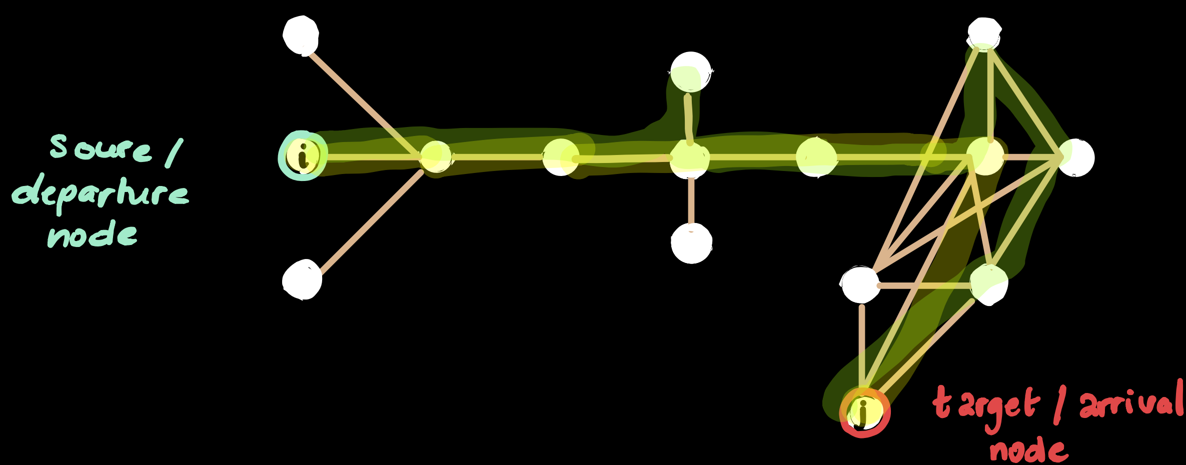
How to get from node i to node j
in a rapid and efficient way?

> Path length and related metrics based on shortest paths seem like reasonable measures of the capacity of a network to support rapid, integrated, and efficient communication.

→ shortest path length = { hop count in binary networks
minimum sum of edge weights



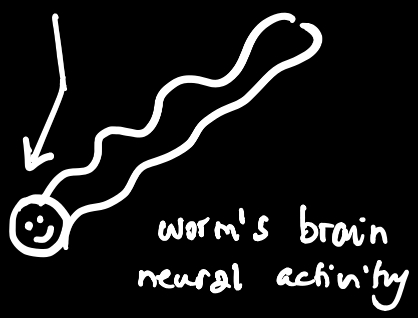
> “Now what is the underlying assumption about information flow in a graph?”



Claim

Signals / information propagate in real-world graphs via the shortest paths between graph elements.

→ The most effecient path might not be the shortest!



* A worm does not have access to a map of shortest paths of its own nervous system.
* How does a neuron "know" that the best and cheapest way for it to send a signal to another neuron is via a two-hop path mediated by a third neuron?

Underlying assumption about the neural graph of a worm's brain:
The routing information along shortest paths mandates that each node has a global knowledge of the graph to find the appropriate path.

> "So what is the best model to use for measuring or quantifying effective information flow or communication within a graph?"

THE THREE TOURISTS

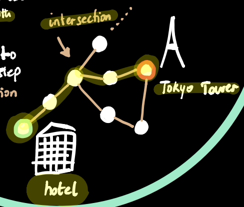
Scenario 1

A walker tracing out a path in a graph.

e.g., Alice is visting Japan. She wants to go to Tokyo Tower from her hotel.

① Alice wants to use the shortest path possible.

② How she get to choose her next step at the intersection Knowing she has a map?



Alice has a clear destination.
Knows the optimal route to get there.

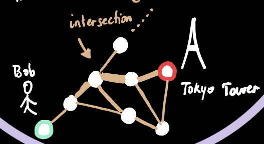
Scenario 2

A walker explores the roads without any road map.

E.g., Bob randomly chooses a direction whenever he arrives at an intersection.

But soon he realizes that doing so has him walking in circles.

He biases his random choices so he is more likely to exit an intersection along the busiest/widest road.



Bob has no clear target
There is no apparent order to his journey.

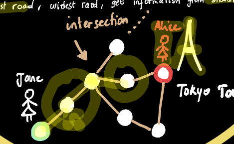
Scenario 3

A walker Knows his destination but does not Know how to get there.

E.g., Jane wants to meet Alice at the Tokyo Tower.

① To compensate for this lack of global knowledge, Jane makes decisions based only on the information at hand → local properties of her current location.

(busiest road, widest road, get information from another walker)



The locally informed decisions cannot guarantee that Jane will ever reach her destination.

But she makes an informed guess based on the available information.

Model 1 → routing via a shortest path based on global knowledge.

Model 2 → "diffusing" through the graph in a random way.

Model 3 → navigation to a target destination according to local information.

We will consider graph measures quantifying the efficiency of information flow under:

Shortest path routing
- efficiency
  - global
  - nodal
  - local
- Shortest path identification (Dikstra's algo)

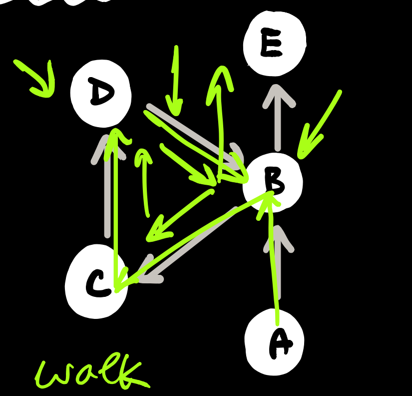
Diffusion of information flow
- diffusion efficiency
- more over the next lectures (fusing graphs learning graph-based metrics, graph embedding, ...)

walks, trails, paths, and cycles

- A walk is a sequence of edges that an entity can "walk" along, in a continuous manner, without having to ever "jump" from one node to another node and thereby break the contiguity of the sequence.
  A walk can visit the same node(s) or edge(s) more than once.
- A trail is a walk where all edges are unique.
- A path is a trail in which all edges and nodes are unique.
- A cycle is a closed walk where first and last nodes are the same.
  ↳ can be embedded within a larger walk.

paths ⊂ trails ⊂ walks ⊃ cycles

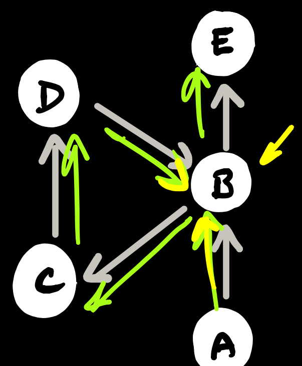
Example :



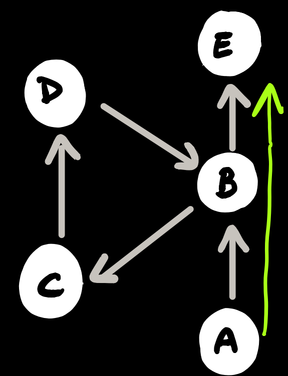
walk

* { A-B-C-D-B-C-D-B-E }

* { B-C-D-B } → cycle



* { A-B-C-D-B-E }
trail



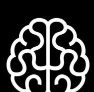
* { A-B-E }
path

Remark

In some of the literature, the term "path" is used as a generic description of trails, walks, paths, and cycles.
In this case, a simple path means a path in which all nodes and edges are unique.

Shortest path routing

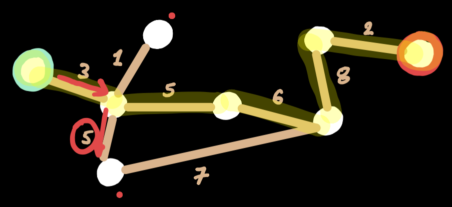
Shortest paths are fundamental to graph theory.
Many works focused on shortest paths as the principle routes along which information is communicated in a graph.



What about our brains?
↳ Whether or not neural information is routed via the shortest paths of a nervous system nonetheless remains an unresolved question in neuroscience.

Diffusion of information flow

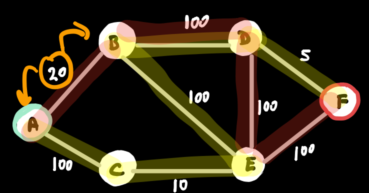
" What exactly do we mean by a shortest path ?
– Beware! Once you give it a definition, you build on it and act on it –



Definition

A shortest path, a geodesic, in a graph is an ordered set of edges linking two nodes in a graph for which the sum of the weights of its edges is minimal.

" What if the edges with the highest weights mediate the strongest and most efficient flow of information in a graph? "

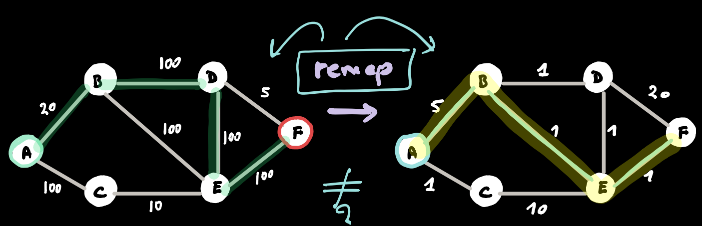


longest path ?

A → C → E → B → D → F = 315
A → B → D → E → F = 320 ✓

Idea → remap the longest path to the shortest path.
↳ similarity-to-distance remapping to the edge weights before computing the shortest paths.

$w_{ij} \leftarrow 1/w_{ij}$



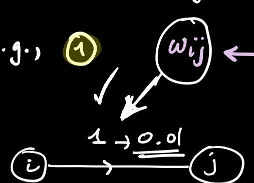
Longest path = {A, B, D, E, F}
$\ell = 320$

≠ ? =

Shortest path = {A, B, E, F}
$\ell = 7$

There is no theoretical justification for a remapping based on the reciprocal of the edge weights ($1/w_{ij}$), so we might consider other remapping functions.

E.g., ① $w_{ij} \leftarrow 1/w_{ij}^{\beta}$, $\beta > 0$



Larger values of β accentuate the confidence we have in edges with larger weights representing stronger connections.

② if $0 < w_{ij} < 1$

$$\begin{cases} w_{ij} \leftarrow -\log w_{ij} \\ w_{ij} \leftarrow 1 - w_{ij} \end{cases}$$

FINDING SHORTEST PATHS

- One of the classicals in finding shortest paths.

  Dijkstra conceived his algorithm in 1956, apparently while he was shopping with his fiancée and thinking about the shortest way to travel from Rotterdam to Groningen.

  - He published his algorithm in a two-page note 3 years later (Dijkstra, 1959).
  - The paper is in the top 1% of most cited papers ever published across all fields of sciences.

" Is it the first algorithm to compute shortest path lengths between two nodes in a graph? "

No! Matrix methods were developed in the early 1950s by Shimbel (1953) and Luce (1950) for binary graphs.

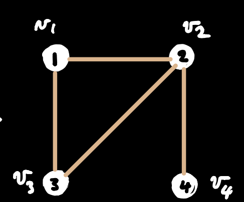
Matrix methods

Given the binary adjacency matrix A of a graph, then A^k characterizes all walks in a network that traverse exactly k edges (hops).

- For a graph with n nodes and with binary edges, the entries $n \times n$ adjacency matrix are defined by:

$$A = \begin{cases} A_{ii} = 0 \\ A_{ij} = 1 & \text{if there is an edge } e_{ij} \\ A_{ij} = 0 & \text{if there is no edge} \end{cases}$$

$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 1 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 \end{array} \longrightarrow$$

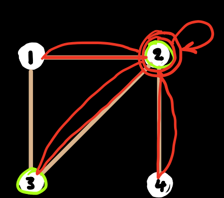


- The number of walks of length 2 between two nodes:

$$N_2(v_i, v_j) = A^2(i,j)$$

E.g.,

$$A^2 = \begin{array}{c|cccc} & & 2 & 3 & \\ \hline & 2 & 1 & 1 & 1 \\ 2 & 1 & \boxed{3} & \boxed{1} & 0 \\ & 1 & 1 & 2 & 1 \\ & 1 & 0 & 1 & 1 \end{array}$$



$N_2(v_2, v_3) = 1$; $N_2(v_2, v_2) = 3$

Theorem

The number of walks of length k joining any two nodes v_i and v_j of a binary graph is given by the (i,j) entry of the matrix A^k:

$$N_k(v_i, v_j) = A^k(i,j)$$

Proof. This is easily proved by induction on k, as you are invited to do. □

DIJKSTRA'S ALGO

KEY RULES

- Dijkstra's algorithm is iterative.
- At each iteration, we update the path lengths from a user-defined initial node to all other nodes in the graph.
- Each node i is assigned a single value ℓ_{0i} that represents its path length from the initial node 0. $0 \cdots\cdots \rightarrow 5 \leftarrow$
- We aim to iteratively decrease the path length values assigned to each node.
- Initial node is assigned a path length of zero, while all other nodes are assigned ∞.
- We start by visiting the neighbors of the initial node. The algorithm stops when all nodes are visited.

Key steps:

0/ Initialize path length in each node.

1/ From the starting node, $\boxed{\text{visit}}$ the node with the smallest known distance / cost.

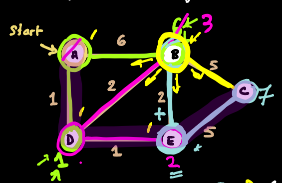
① create a list to keep track of each visited nodes.

② create a list to keep track of unvisited nodes.

2/ Once we've moved to the smallest-cost node, check each of its neighboring nodes.

3/ Calculate the distance/cost for the neighboring nodes by summing the cost of the edges leading from the start node.

4/ If the distance/cost to a node we are checking is less than a known distance, update the shortest distance for that vertex.



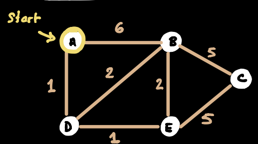
node	shortest distance from A	previous node
* A	0	
B	~~6~~ $\boxed{3}$	~~A~~ D
C	7	E
* D	$\boxed{1}$	A
* E	$\boxed{2}$	D

visited = [A, D, E, B, C]

unvisited = [ ]

A ← D ← E ← C

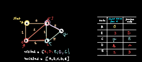
Solution:



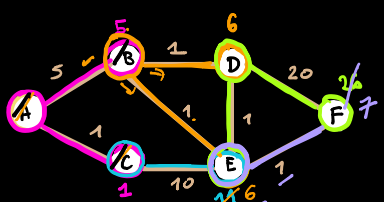
node	shortest distance from A	previous node
A	0	
B	3	D
C	7	E
D	1	A
E	2	D

visited = [A, D, E, B, C]

unvisited = [ ]



Exercise: run Dijkstra's algorithm step-by-step on the following graph.



node	Shortest distance	previous node
* A	0	
B	$\boxed{5}$	A
* C	$\boxed{1}$	A
D	$\boxed{6}$	B
E	~~11~~ $\boxed{6}$	~~C~~ B
F	~~26~~ 7	~~D~~ E

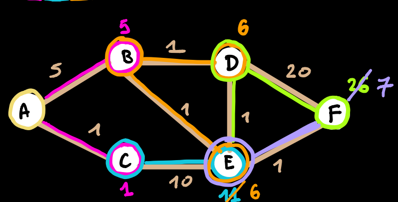
Smallest

visited = [A, C, B, D, E, F]

unvisited = [~~A~~, ~~B~~, ~~C~~, ~~D~~, ~~E~~, ~~F~~]

* A node is marked as visited when all its neighbors have been checked.

* Always visit the unvisited node with the smallest distance (in the table).

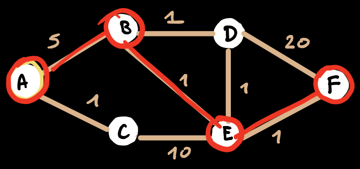


node	Shortest distance	previous node
A	0	
B	$\boxed{5}$	A
C	$\boxed{1}$	A
D	$\boxed{6}$	B
E	~~11~~ $\boxed{6}$	~~C~~ B
F	~~26~~ 7	~~D~~ E

D or E

visited = [A, C, B, D, E, F]

unvisited = [ ]



node	shortest distance	previous node
A	0	
B	$\boxed{5}$	A
C	$\boxed{1}$	A
D	$\boxed{6}$	B
E	~~11~~ $\boxed{6}$	~~C~~ B
F	~~26~~ 7	~~D~~ E

D or E

$l = 7$

what is the shortest path from A to F and its length?

Dijkstra's algorithm can handle weighted directed graphs but not negatively weighted edges!

Why is that?

How can we solve this? [Hint: Bellman-Ford algo] HW 2 :)

SHORTEST PATH ROUTING EFFICIENCY

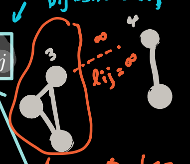
1. Characteristic path length

The charecteristic shortest path length L is the average shortest path length between all possible pairs of nodes in a directed or undirected graph:

$\left[1/l_{ij}\right] \to 0$ ∞

Dijkstra's algo

$$L = \frac{1}{N}\sum_i \boxed{l_i} = \underbrace{\frac{1}{N(N-1)}}_{\text{number of node pairs excluding self-pairing}} \sum_{i \neq j} \boxed{l_{ij}}$$



$\frac{L_{C_1} + L_{C_2}}{2}$

average shortest path from node i to all other nodes

shortest path length from node j to node i.

Intuition

A short characteristic path length L means that information can, on average, be routed between pairs of nodes using only a few connections.

" Can we compute L for fragmented graphs (with more than one connected component)?

No since a path will not exist between at least one pair of nodes → $L = \infty$

Harmonic mean (Newman, 2003) estimates the average shortest path length between all possible node pairs in a fragmented graph.

$$\boxed{L'} = N(N-1)\Big[\sum_{i \neq j} \boxed{\frac{1}{l_{ij}}}\Big]^{-1}$$

harmonic mean is finite unless the graph has no edges at all!

reciprocal path is summed
→ if i & j belong to different components ⇒ $l_{ij} = \infty$ ⇒ $\frac{1}{l_{ij}} = 0$

✓ or ✗

$$L' = N(N-1)\Big[\sum_{i \neq j} \frac{1}{l_{ij}}\Big]^{-1}$$

✗ Harmonic mean is sensitive to outliers (i.e., node pairs with exceptionally long shortest paths.

$L' = 0.\ldots\ldots 1$

✓ Harmonic mean is a hub-centric measure of integration as it emphasizes the path lengths originating from hub regions and down-weights peripheral nodes.

E.g., shortest paths lengths in a graph: {2, 2, 3, 12}

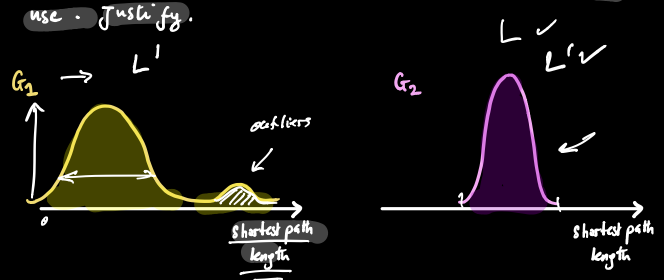
[2 → 3]

↳ L = 4 and $\underbrace{L' = 2.1}$

more characteristic of the majority of path lengths in G.

The characteristic path length is a global measure of a graph's capacity to integrate information using shortest path routing. Using harmonic mean is desirable in fragmented graphs and is not unduly influenced by a small proportion of node pairs with exceptionally long shortest paths.

* Given the following distributions of shortest path lengths in two different graphs, specify which efficiency measure to use. Justify.



2. Global efficiency

A graph's global efficiency, E_{glob}, is the reciprocal of the harmonic mean of its path lengths:

$$E_{glob} = \frac{1}{L'} = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{l_{ij}}$$

harmonic mean

(Latora and Marchiori, 2001) contend that E_{glob} is the efficiency of information exchange in a parallel system in which nodes concurrently exchange information via shortest paths. ✓

≠

whereas $\frac{1}{L}$ is a better measure of the efficiency of a sequential system in which information is processed serially. ✓

3. Nodal efficiency

it is defined for each node j in G.

"hub" "closeness centrality"

$$E_{nodal}(j) = \boxed{\frac{1}{N-1}} \sum_{i} \boxed{\frac{1}{l_{ij}}}$$

normalized

sum

reciprocal of the shortest path length from node j to node i

* Hub nodes tend to have the highest nodal efficiency, facilitating their role in efficiently integrating and distributing information.

"Any difference between closeness centrality and E_{nodal} of a node in a graph G?"

3. Local efficiency

(Latora and Marchiori, 2011) define a node-specific local measure:

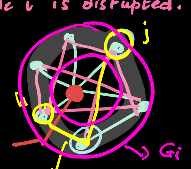
$$E_{local}(i) = \frac{1}{N_{G_i}(N_{G_i}-1)} \underbrace{\sum_{j,h \in G_i} \frac{1}{l_{jh}}}$$

subgraph comprising all immediate neighbors of node i.

how efficiently the neighbors of a node can communicate when that node i is disrupted.

reflects the extent to which a node is integrated between its immediate neighbors ($\neq E_{glob}$)

local scale



$\ell_{jh} = 2$

$j, h \in G_i$