Lecture 13: Shortest Paths and Small-world Networks

Modelling Complex Systems (Spring 2026)

Jordi-Lluís Figueras

Matematiska Institutionen, Uppsala Universitet

May 7, 2026

Starting with your discoveries

In the last discussion, the class did something very valuable: you turned a network question into algorithms.

The problem was:

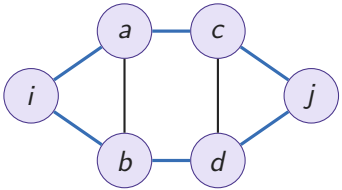
Counting shortest paths

Given a graph and two nodes i and j, how many shortest paths connect i to j?

Students found two correct routes to the answer:

- a matrix-power method using the adjacency matrix;
- a propagation method that is essentially breadth-first dynamic programming.

This is exactly the kind of mathematical discovery we want in this course.



two shortest paths from i to j

Lecture goals

In this lecture we will:

- formulate precisely the shortest-path counting problem;
- prove why powers of the adjacency matrix count walks;
- use the first nonzero matrix power to count shortest paths;
- explain the propagation algorithm found in class;
- show that both methods are the same matrix-vector propagation from e_i;
- connect this to the number of shortest paths using a specified edge;
- resume the Watts-Strogatz model from Lecture 12;
- explain how a few shortcuts create small-world networks.

Counting Shortest Paths

The algorithms you found in class

Problem and notation

Let $G = (V, E)$ be an unweighted graph. For now, think of an undirected graph; the directed case only changes which neighbors are allowed.

For two nodes $i, j \in V$:

- $\text{dist}(i, j)$ is the length of a shortest path from i to j;
- σ_{ij} is the number of shortest paths from i to j.

If no path connects i and j, then we set $\sigma_{ij} = 0$.

Let M be the adjacency matrix:

$$M_{uv} = \begin{cases} 1 & \text{if } u \text{ is adjacent to } v, \\ 0 & \text{otherwise.} \end{cases}$$

Student algorithm 1: powers of the adjacency matrix

The first idea was:

Matrix-power algorithm

Compute $M, M^2, M^3, \ldots$ until the entry $(M^n)_{ij}$ is nonzero.

Then:

$$d = \min\{n \geq 0 : (M^n)_{ij} > 0\}$$

is the distance from i to j, and

$$\sigma_{ij} = (M^d)_{ij}$$

is the number of shortest paths.

The key point is that matrix multiplication automatically sums over all possible intermediate nodes.

Reference: Wikipedia contributors, "Adjacency matrix," especially the discussion of matrix powers.

Why matrix powers count walks

For $n = 2$,

$$(M^2)_{ij} = \sum_k M_{ik} M_{kj}.$$

The term $M_{ik} M_{kj}$ equals 1 exactly when

$$i \to k \to j$$

is a walk of length 2.

In general,

$$(M^n)_{ij} = \sum_{k_1, \ldots, k_{n-1}} M_{ik_1} M_{k_1 k_2} \cdots M_{k_{n-1} j}.$$

Each product is 1 if the sequence

$$i, k_1, k_2, \ldots, k_{n-1}, j$$

is a walk of length n, and it is 0 otherwise.

From walks to shortest paths

A matrix power counts walks, not only simple paths. So why does the first nonzero power count shortest paths?

Let

$$d = \min\{n \geq 0 : (M^n)_{ij} > 0\}.$$

Then there is no walk from i to j of length smaller than d.

Moreover, any length-d walk from i to j cannot repeat a vertex. If it did, the repeated part would form a loop that could be removed, producing a shorter walk from i to j.

Therefore the length-d walks counted by $(M^d)_{ij}$ are exactly the shortest paths from i to j.

Student algorithm 2: propagation from the source

The second idea was to propagate numbers outward from the source node.

Propagation algorithm

Put the number 1 on node i. At each step, assign numbers to the still unnumbered nodes that touch the current wavefront. The number assigned to a new node is the sum of the numbers on adjacent already-numbered nodes.

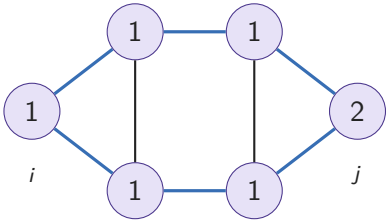
Stop when node j receives a number.

The number on j is σ_{ij}.

This is breadth-first search plus a dynamic programming count.

Reference: Wikipedia contributors, "Breadth-first search." The counting labels are a dynamic-programming extension.

Propagation example



The layers from i are:

$$\{i\}, \quad \{a, b\}, \quad \{c, d\}, \quad \{j\}.$$

The recurrence gives:

$$\sigma_{ii} = 1,$$

$$\sigma_{ia} = \sigma_{ib} = 1, \qquad \sigma_{ic} = \sigma_{id} = 1,$$

and finally

$$\sigma_{ij} = \sigma_{ic} + \sigma_{id} = 2.$$

The two shortest paths are the two blue routes through the top and bottom branches.

Why the propagation algorithm works

Let

$$L_r = \{v \in V : \text{dist}(i, v) = r\}$$

be the r-th layer around i.

Every shortest path from i to a vertex $v \in L_r$ must arrive from a neighbor in L_{r-1}.

Therefore

$$\sigma_{iv} = \sum_{\substack{u \sim v \\ u \in L_{r-1}}} \sigma_{iu}.$$

This is exactly the rule the class discovered: new labels are sums of labels on the previous wavefront.

By induction over r, the number written on each node is the number of shortest paths from i to that node.

Matrix powers one vector at a time

Let e_i be the vector with a 1 at node i and zeros elsewhere.

Instead of forming all matrices M^n, propagate one vector:

$$x_0 = e_i, \qquad x_{n+1} = M x_n.$$

Then

$$x_n = M^n e_i.$$

For an undirected graph, $M = M^T$, so

$$(x_n)_j = (M^n e_i)_j = (M^n)_{ji} = (M^n)_{ij}.$$

Thus the first n for which $(x_n)_j > 0$ is $\mathrm{dist}(i, j)$, and that first value is σ_{ij}.

Why this is the same propagation rule

The vector update says that, for each node v,

$$(x_{r+1})_v = \sum_u M_{vu}(x_r)_u = \sum_{u \sim v} (x_r)_u.$$

This is exactly the instruction:

add the numbers on adjacent nodes.

The difference is only what we store:

- the matrix-power method thinks about all entries of M^n;
- the vector method updates only the column started from i;
- the BFS propagation records each node when it first becomes reachable.

So the two student methods are the same linear propagation, viewed at two different scales.

References for the two algorithms

The two classroom algorithms correspond to standard graph-algorithm ideas.

- Matrix-power method: Wikipedia contributors, "Adjacency matrix," especially the section explaining powers of the adjacency matrix.
- Propagation method: Wikipedia contributors, "Breadth-first search." The shortest-path counts are obtained by adding a dynamic-programming count to the BFS layers.

Useful links:

- `https://en.wikipedia.org/wiki/Adjacency_matrix`
- `https://en.wikipedia.org/wiki/Breadth-first_search`

Why this matters for edge betweenness

In Lecture 12 we defined edge betweenness using

$$\sigma_{st} \qquad \text{and} \qquad \sigma_{st}(e),$$

where $\sigma_{st}(e)$ is the number of shortest paths from s to t that use the edge e.

The class discussion gives the computational building block:

- count all shortest paths between two nodes;
- count how many of those paths use a specified edge;
- sum these fractions over all pairs to measure global bottleneck importance.

So the algorithms you found are not side calculations. They are exactly what network centrality needs.

Shortest paths that use one edge

Let e be an edge, and let $d = \text{dist}_G(i, j)$.

The student idea was the complement principle:

$$\text{paths using } e = \text{all paths} - \text{paths avoiding } e.$$

For shortest paths this becomes

$$\sigma_{ij}^G(e) = \sigma_{ij}^G - \#\{\text{shortest paths from } i \text{ to } j \text{ in } G \text{ avoiding } e\}.$$

This is the right idea. The only precision is that "shortest" must mean shortest in the original graph G.

Computing the complement correctly

Remove the edge e and call the new adjacency matrix M'.

Keep the original distance

$$d = \mathrm{dist}_G(i, j).$$

Then the number of original shortest paths avoiding e is

$$\left((M')^d\right)_{ij}.$$

Therefore

$$\boxed{\sigma_{ij}^G(e) = \sigma_{ij}^G - \left((M')^d\right)_{ij}.}$$

If removing e does not change the distance, this is the same as running the shortest-path count on the modified graph. If removing e makes the distance longer, we must not subtract the new longer shortest paths.

A useful lesson from the algorithms

The two student algorithms look different, but they express the same structure.

Method	Main object	What is being counted
Matrix powers	M^n	walks of length n
Propagation	BFS layers	shortest paths by distance layer
Complement	G versus $G \setminus e$	shortest paths using an edge

The common idea is to combine graph distance with counting.

This prepares the transition back to small-world networks: shortcuts matter because they change distances and the set of shortest paths.

Small-world Networks

Resuming the Watts-Strogatz model from Lecture 12

Where we resume

At the end of Lecture 12, we introduced two network statistics:

- the clustering coefficient, which measures local triangular structure;
- the average path length, which measures global reachability.

Many real networks have both:

Small-world property

High local clustering together with short average path length.

The Watts-Strogatz model explains how this can happen: keep most local edges, but create a few long-range shortcuts by rewiring selected local edges.

Regular, random, and small-world

Network type	Clustering	Typical distances
Regular lattice	high	long
Random graph	low	short
Small-world network	high	short

The surprising regime is the third one.

A small number of long-range connections can greatly reduce average distance without destroying most local triangles.

This is the same qualitative mechanism we saw in bottleneck examples: one extra route can change global movement more than local degree suggests.

The Watts-Strogatz construction

Start with N nodes arranged on a ring.

Choose an even integer k. Connect each node to its $k/2$ nearest neighbors on each side.

Then inspect each local edge once and rewire it with probability p, avoiding self-loops and duplicate edges.

The parameter p controls the amount of randomness:

- $p = 0$: regular ring lattice;
- small $p > 0$: a few shortcuts;
- $p = 1$: highly randomized network.

The small-world regime occurs for small but positive p.

How a rewired edge is chosen

Label the nodes $0, 1, \ldots, N-1$. For each node i, consider only the $k/2$ edges from i to the nodes

$$i + r \pmod{N}, \qquad r = 1, \ldots, k/2.$$

This convention lists each undirected lattice edge exactly once.

For each such edge (i, j):

1. with probability $1 - p$, keep (i, j);
2. with probability p, remove (i, j) and choose a new endpoint m uniformly at random from the allowed nodes;
3. add (i, m), where allowed means $m \neq i$ and (i, m) is not already an edge.

Thus the endpoint i is fixed, the other endpoint is moved, and the total number of edges remains $Nk/2$.

Question: choosing a free endpoint

Fix a node i. Suppose we have a list of all occupied edges with one endpoint equal to i.

We want to choose a new edge (i, m) that is not on this list. Consider the following procedure:

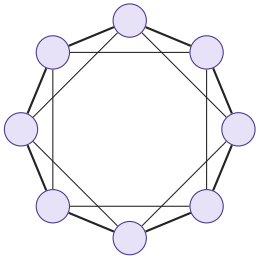
1. choose a candidate node m uniformly at random;
2. check whether (i, m) is already on the occupied-edge list;
3. if (i, m) is not on the list, accept m;
4. if (i, m) is on the list, choose a new candidate m' and repeat.

At the end, are we choosing the accepted endpoint uniformly at random from the allowed nodes?

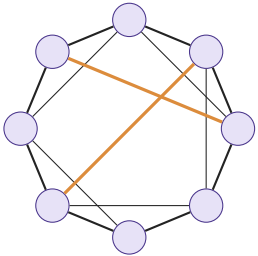
Will this process stop in finite time?

Ring lattice versus shortcuts

Ring-like network



With shortcuts



Shortcuts act as long-range bridges. They can reduce path lengths before they significantly reduce local clustering.

What happens as p increases?

Let $C(p)$ be the average clustering coefficient and let $\ell(p)$ be the average shortest path length.

The qualitative Watts-Strogatz observation is:

- for very small p, $\ell(p)$ often drops rapidly;
- for the same range of p, $C(p)$ remains relatively close to $C(0)$;
- only for larger p does the graph behave like a random graph with much lower clustering.

This produces a window where

$$C(p) \text{ is still high,} \qquad \ell(p) \text{ is already small.}$$

That window is the small-world regime.

Discussion: why is this social?

In a small-world regime we have

$$C(p) \text{ high}, \qquad \ell(p) \text{ small}.$$

Discuss why this is a simple model for society.

Guiding questions:

- What does high clustering mean for friendships, families, workplaces, or local communities?
- What does small average path length mean for reaching people outside one's local circle?
- Which social ties play the role of shortcuts?
- Why can a society be locally clustered and still globally well connected?

Why shortcuts have such a strong effect

In a ring lattice, reaching a distant node may require many local steps.

A shortcut creates a new possibility:

$$\text{local movement} \quad + \quad \text{jump across the ring} \quad + \quad \text{local movement.}$$

One shortcut helps many pairs of nodes, not only its two endpoints.

In terms of shortest-path counting:

- new shorter paths appear for many pairs;
- some old shortest paths stop being shortest;
- shortcut edges can have high edge betweenness.

This connects the opening algorithms with the small-world mechanism.

Numerical experiment

Use the Watts-Strogatz script from Lecture 12:

```
../lec12/scripts/smallWorldWattsStrogatz.py
```

It varies the rewiring probability p and prints:

- average clustering coefficient;
- average shortest path length;
- diameter of the largest connected component;
- fraction of nodes in the largest connected component.

Guiding question:

Small-world transition

How small can p be while still making path lengths much shorter?

Why small worlds matter for dynamics

Small-world structure affects many processes on networks.

Examples:

- epidemics can spread rapidly once shortcuts connect distant local clusters;
- information can travel quickly without every node having many contacts;
- synchronization may be accelerated by long-range coupling;
- failures or cascades may jump between communities through bridge links.

The general lesson is that network topology can change the dynamics even when the local rules stay the same.

Exercise

Connect today's two themes.

Take a small ring lattice and add one shortcut edge e.

For several pairs of nodes i, j:

1. count σ_{ij} before adding e;
2. count σ_{ij} after adding e;
3. decide whether a shortest path now uses e;
4. identify one pair for which the distance changes but local clustering near most nodes does not.

This is the small-world idea in its most concrete form.

What to remember

- $(M^n)_{ij}$ counts walks of length n from i to j.
- The first nonzero power gives the distance, and its value gives the number of shortest paths.
- Updating $x_{n+1} = Mx_n$ from $x_0 = e_i$ is the one-vector version of the same calculation.
- The propagation algorithm works because shortest paths move one BFS layer at a time.
- To count shortest paths using an edge, subtract original-length shortest paths that avoid that edge.
- The Watts-Strogatz model interpolates between a regular ring and a random graph.
- A few shortcuts can strongly reduce average path length while preserving high clustering.

Suggested bibliography

For students who want to read more:

- Wikipedia contributors, “Adjacency matrix,” especially the discussion of matrix powers.
- Wikipedia contributors, “Breadth-first search.”
- Wikipedia contributors, “Watts-Strogatz model,” especially the algorithm section.
- D. J. Watts and S. H. Strogatz, “Collective dynamics of small-world networks,” *Nature*, 393, 440–442, 1998. Paper PDF.
- M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, 2010.
- A.-L. Barabasi, *Network Science*, Cambridge University Press, 2016.
- NetworkX documentation.

Acknowledgment

These slides were prepared with the assistance of OpenAI Codex / ChatGPT 5.5 as a drafting and editing tool.
All mathematical, pedagogical, and course-specific content remains under the responsibility of the lecturer.