Graph Neural Network

Art of Machine Learning - ECE408/208

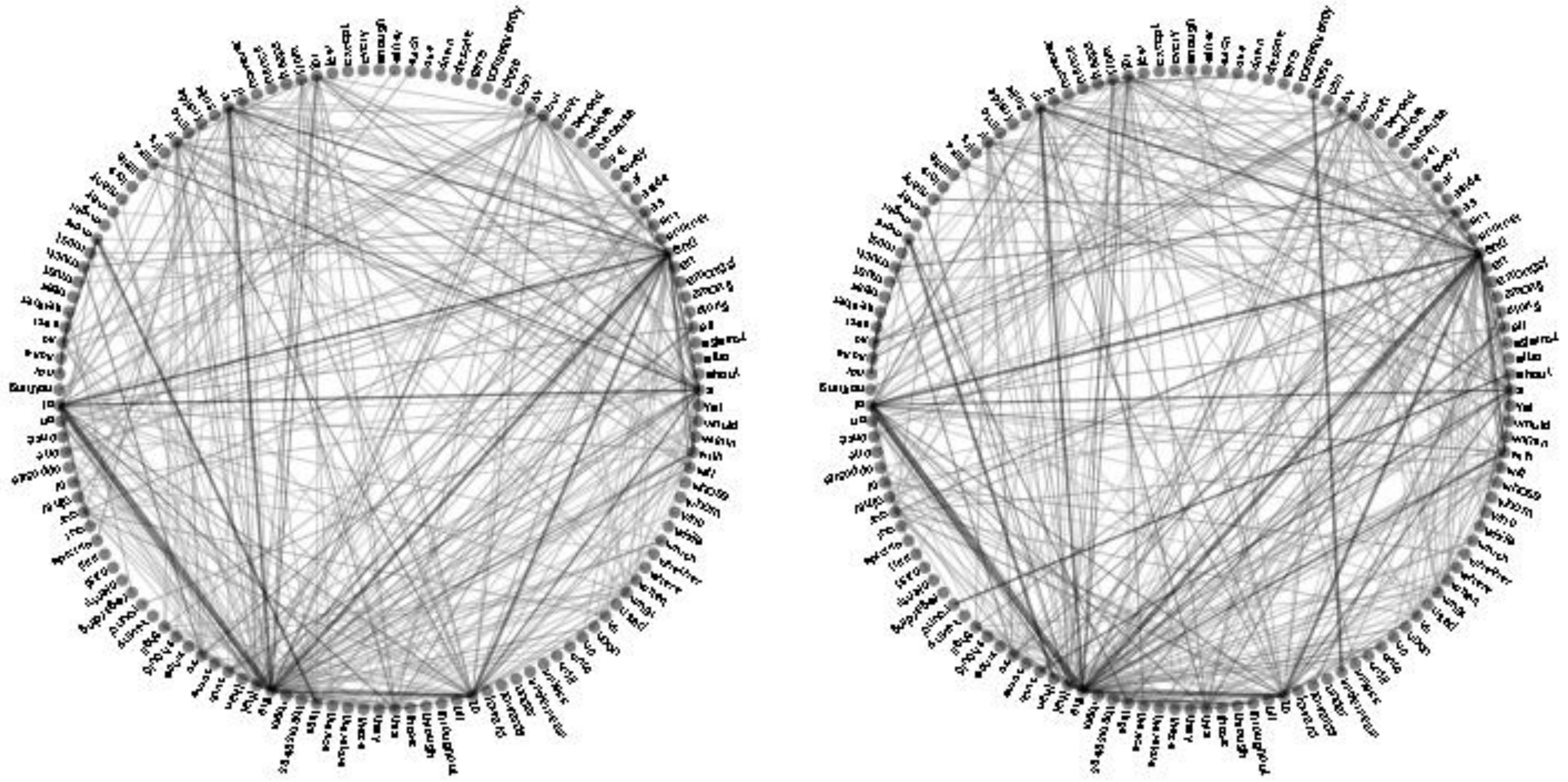
Hamed Ajorlou
Electrical and Computer Engineering Dept.

April 2024

Machine Learning on Graphs: The Why

- Graphs are generic models of signal structure that can help to learn in several practical problems

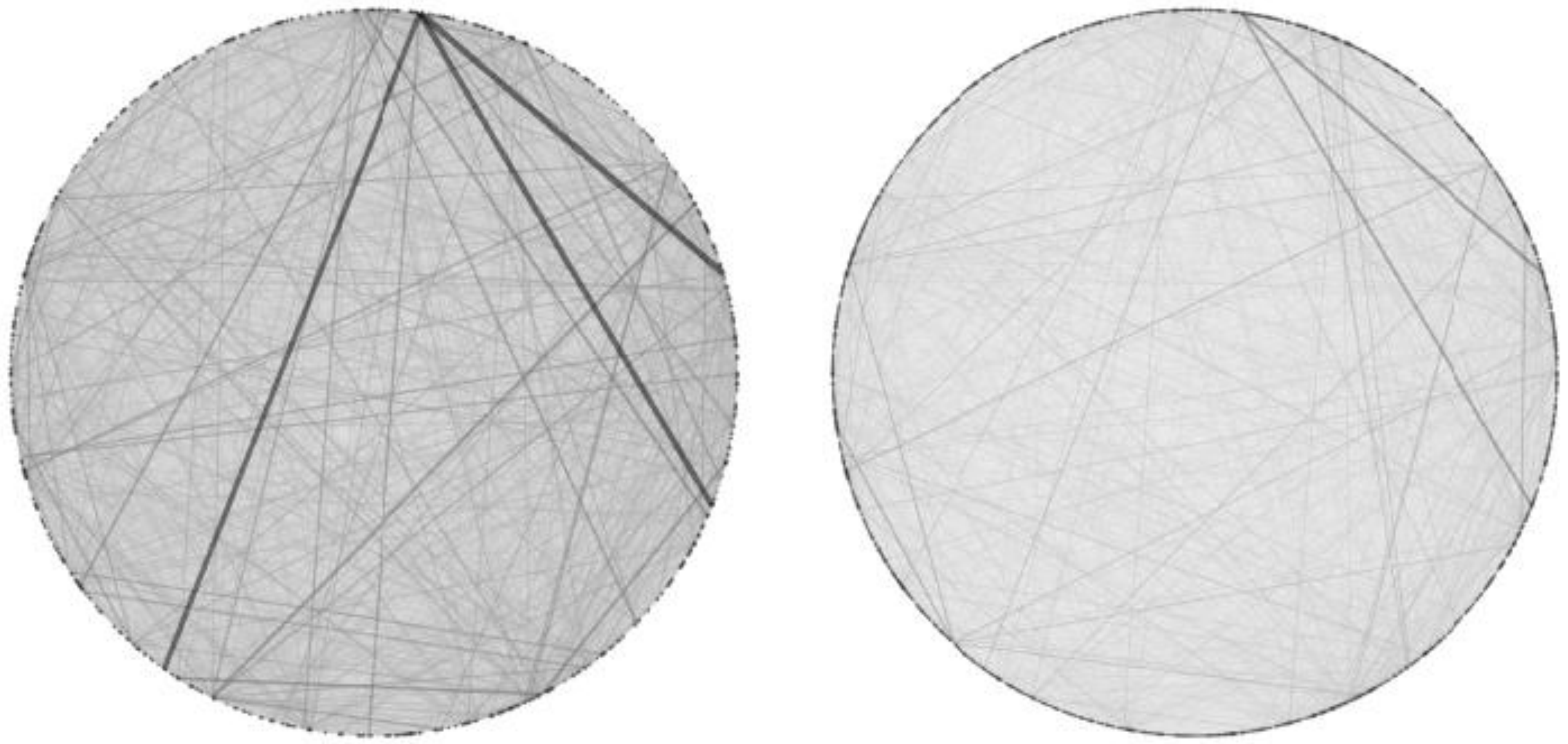
Authorship Attribution



Identify the author of a text of unknown provenance

Segarra et al '16, arxiv.org/abs/1805.00165

Recommendation Systems



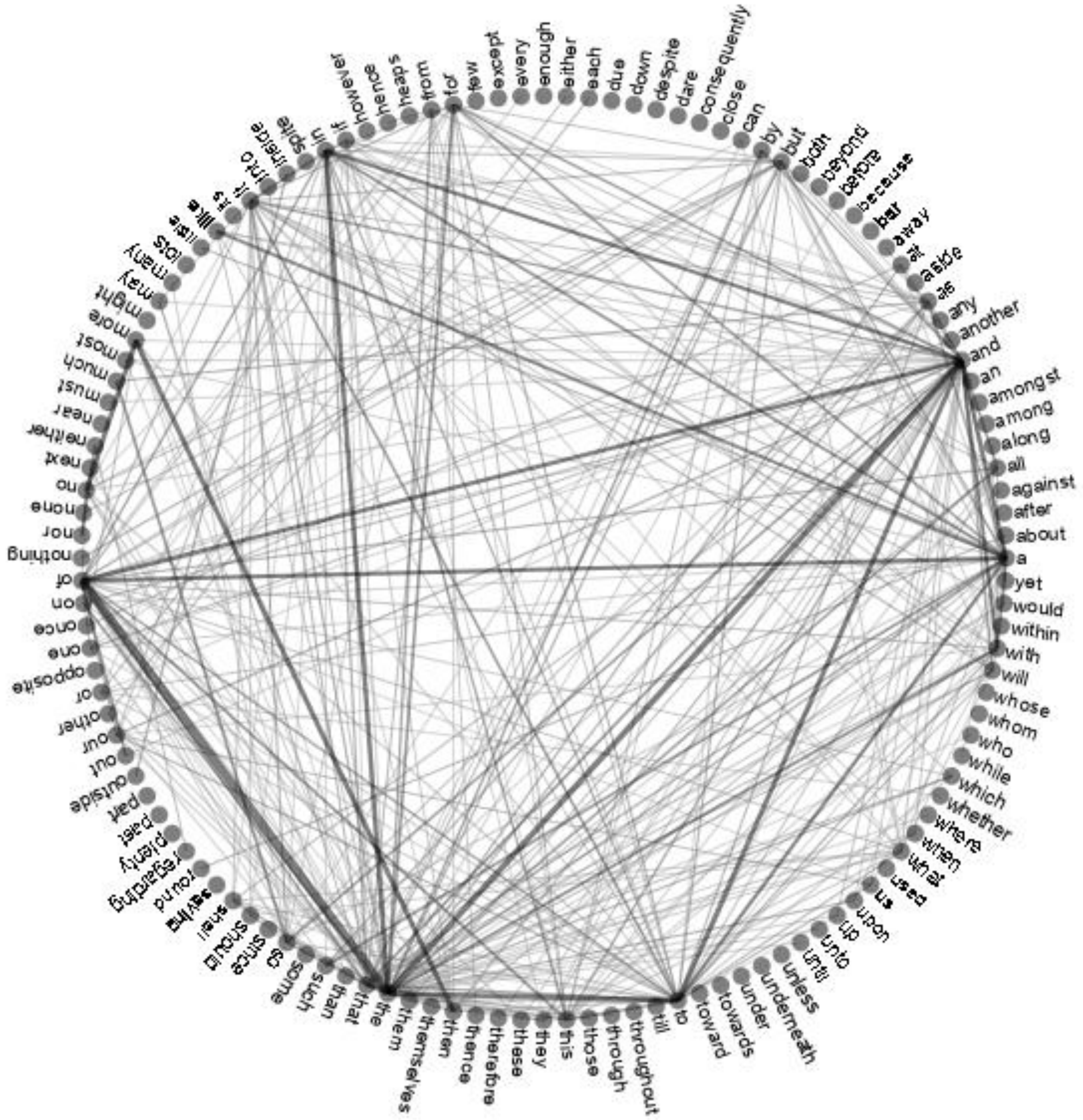
Predict the rating a customer would give to a product

Ruiz et al '18, arxiv.org/abs/1903.12575

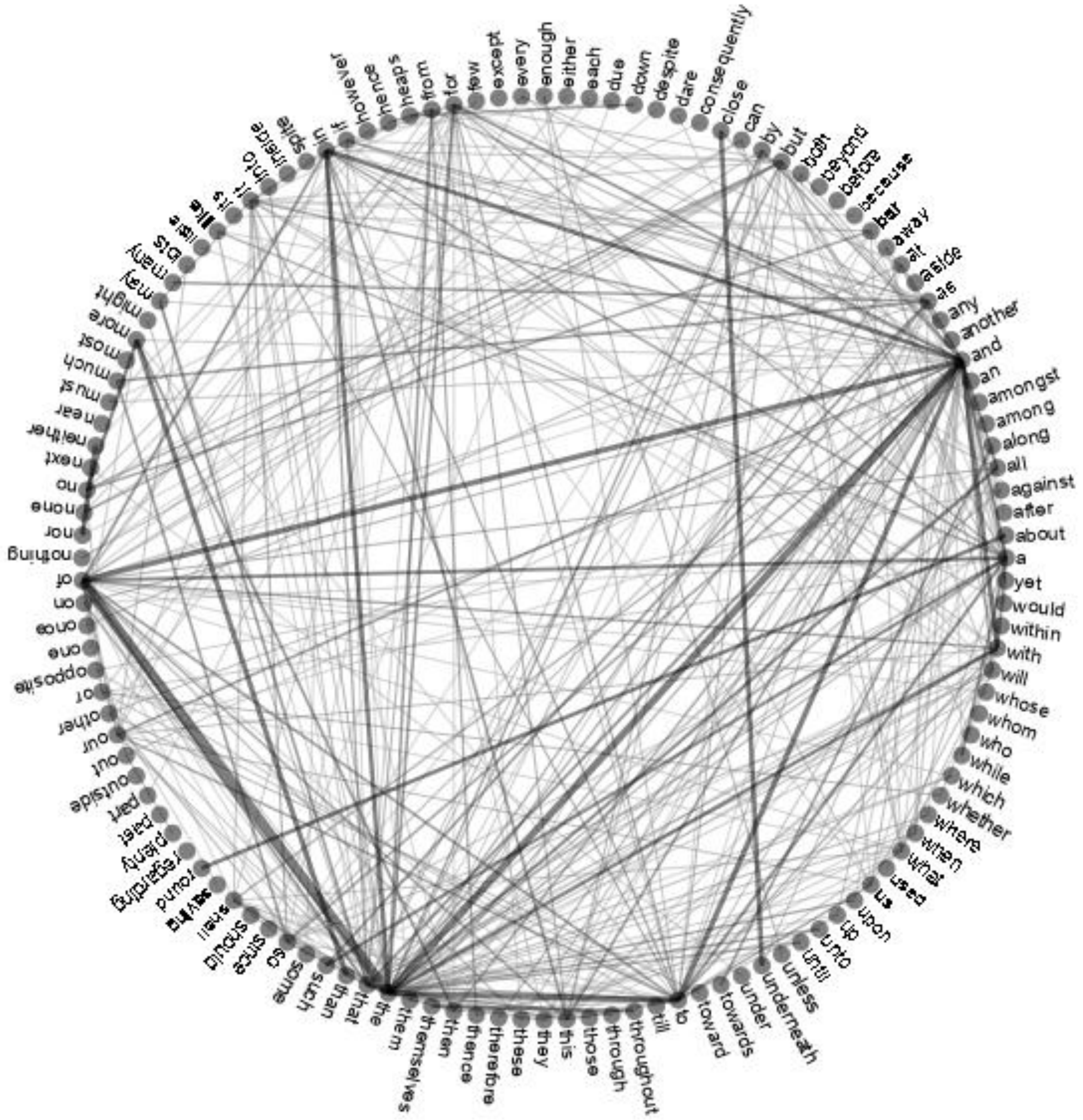
- In both cases there exists a graph that contains meaningful information about the problem to solve

- Nodes represent different function words and edges how often words appear close to each other
  - ⇒ A proxy for the different ways in which different authors use the English language grammar

William Shakespeare



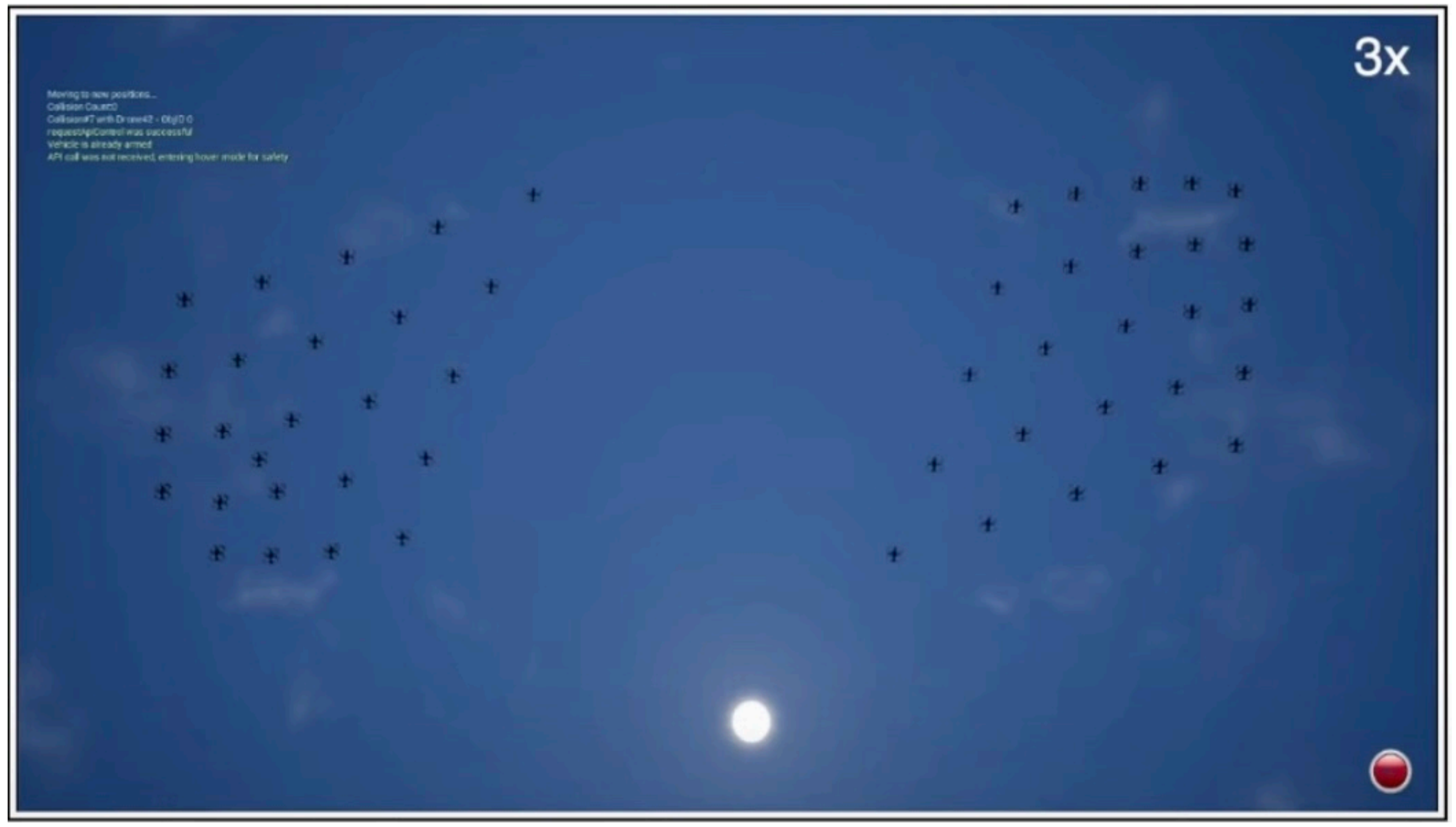
Christopher Marlowe



- WAN differences differentiate the writing styles of Marlowe and Shakespeare in, e.g., Henry VI

- Graphs are more than data structures $\Rightarrow$ They are models of physical systems with multiple agents

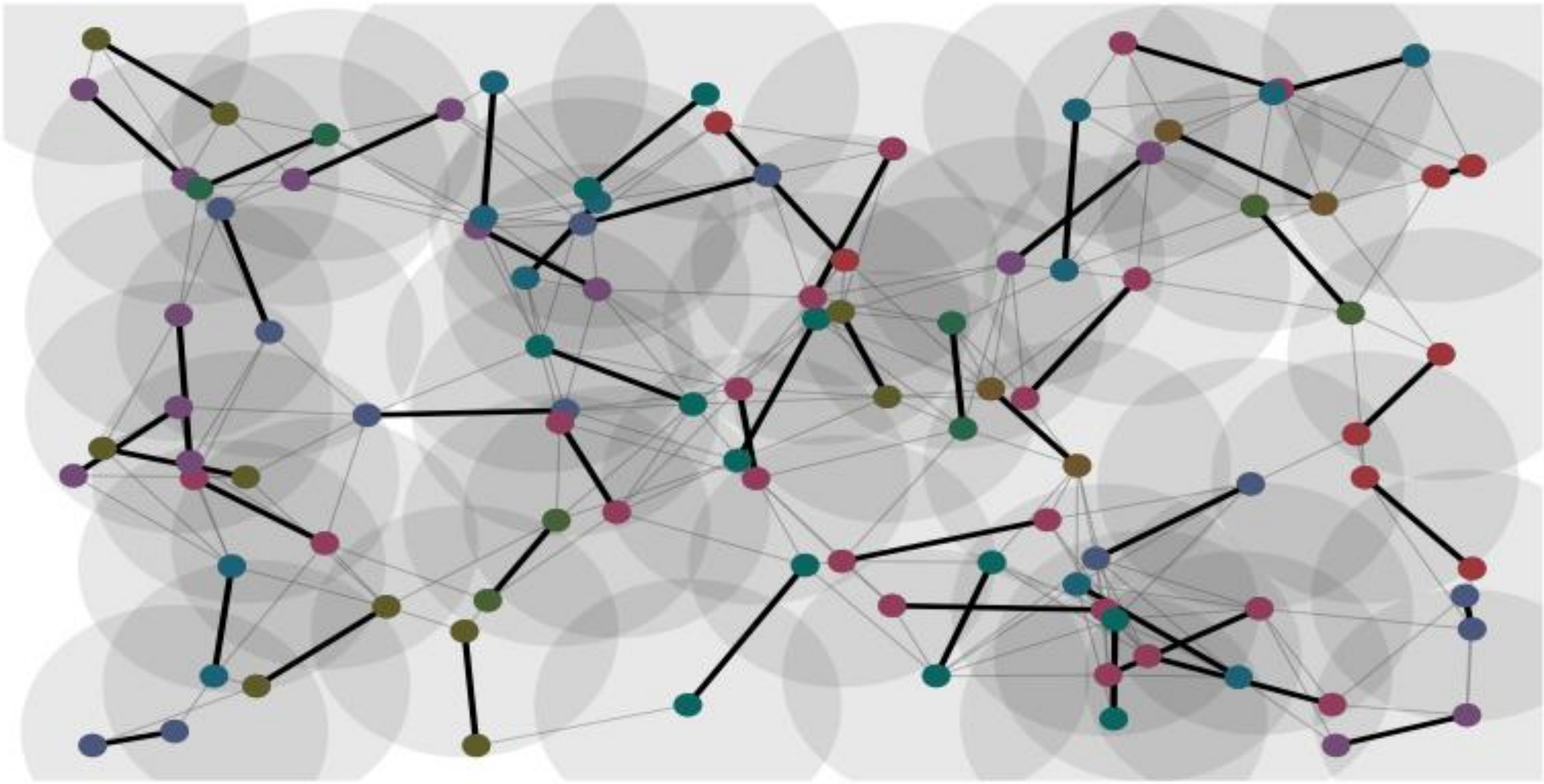
Decentralized Control of Autonomous Systems



Coordinate a team of agents without central coordination

Tolstaya et al '19, arxiv.org/abs/1903.10527

Wireless Communications Networks



Manage interference when allocating bandwidth and power

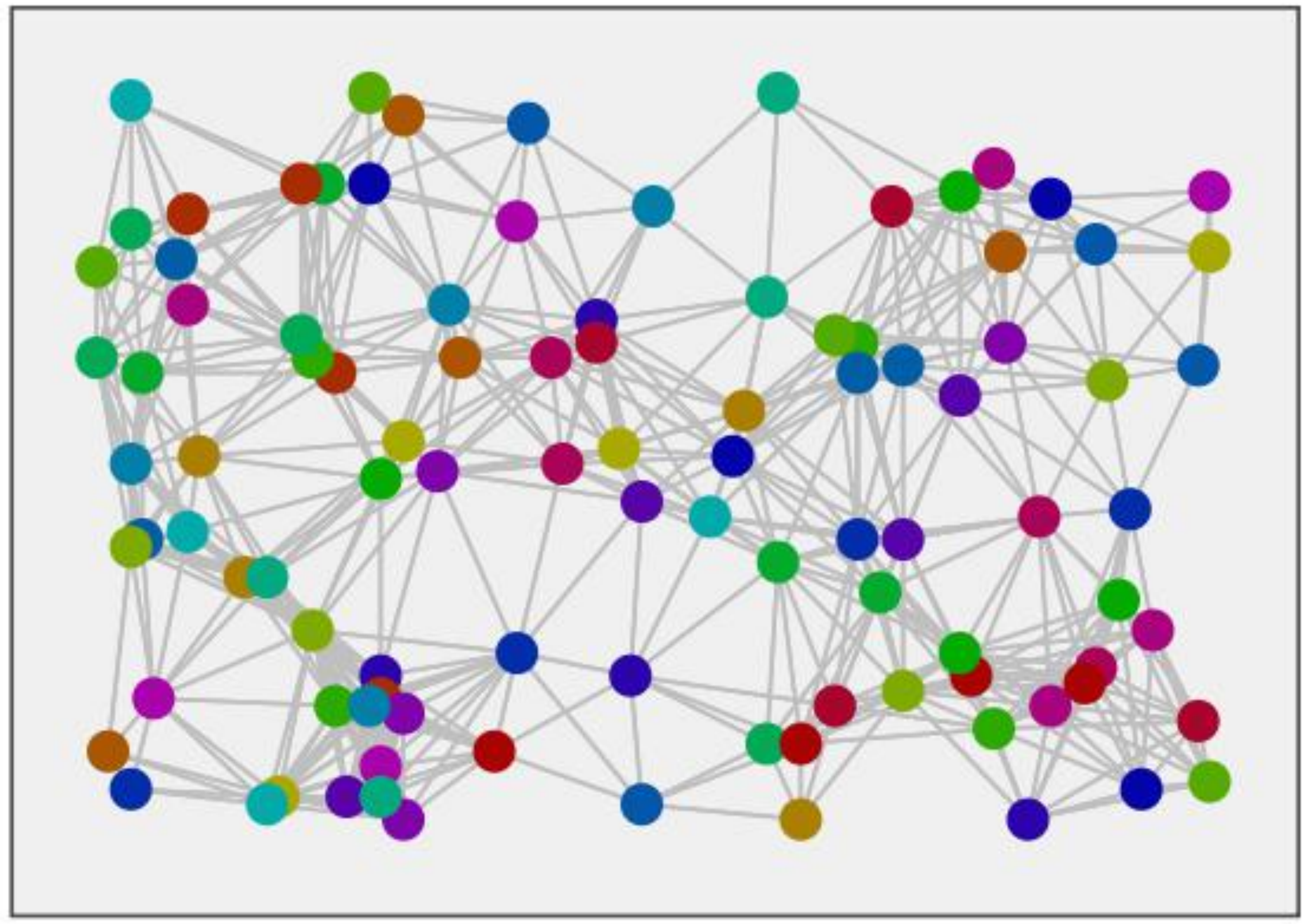
Eisen-Ribeiro '19, arxiv.org/abs/1909.01865

- The graph is the source of the problem $\Rightarrow$ Challenge is that goals are global but information is local

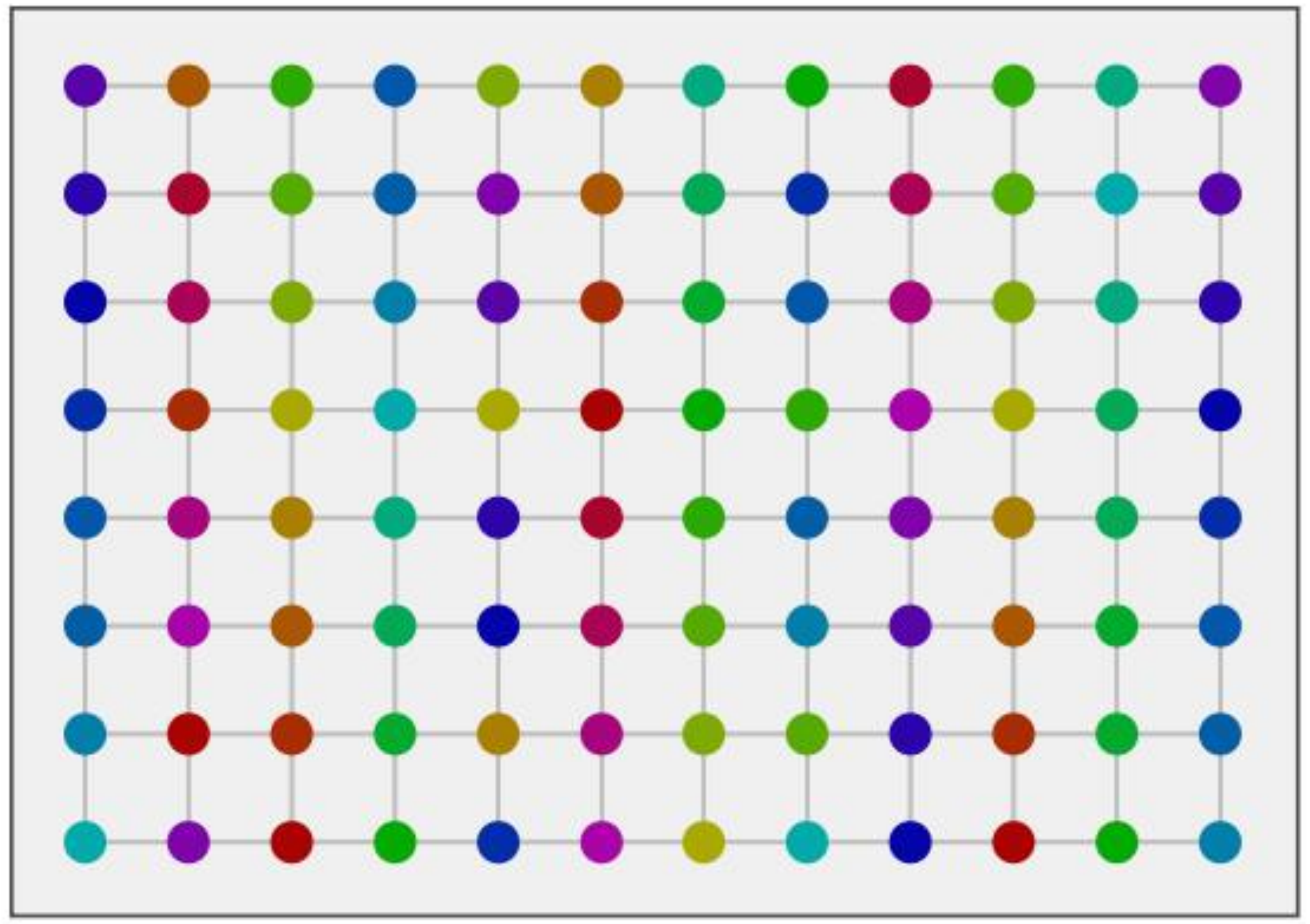
Machine Learning on Graphs: The How

- CNNs are made up of layers composing convolutional filter banks with pointwise nonlinearities

Process graphs with graph convolutional NNs



Process images with convolutional NNs



- Generalize convolutions to graphs $\Rightarrow$ Compose graph filter banks with pointwise nonlinearities
- Stack in layers to create a graph (convolutional) Neural Network (GNN)

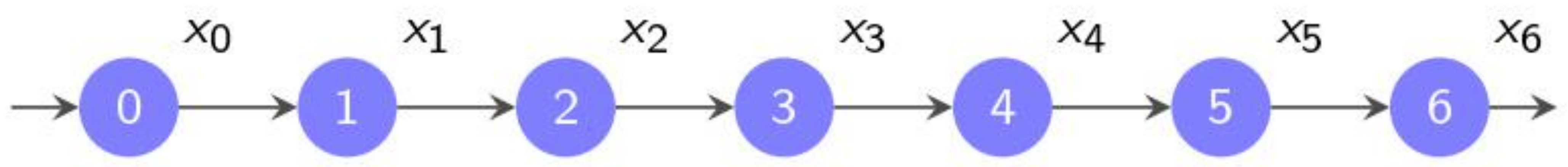
Convolutions in Time, in Space, and on Graphs

- How do we generalize convolutions in time and space to operate on graphs?

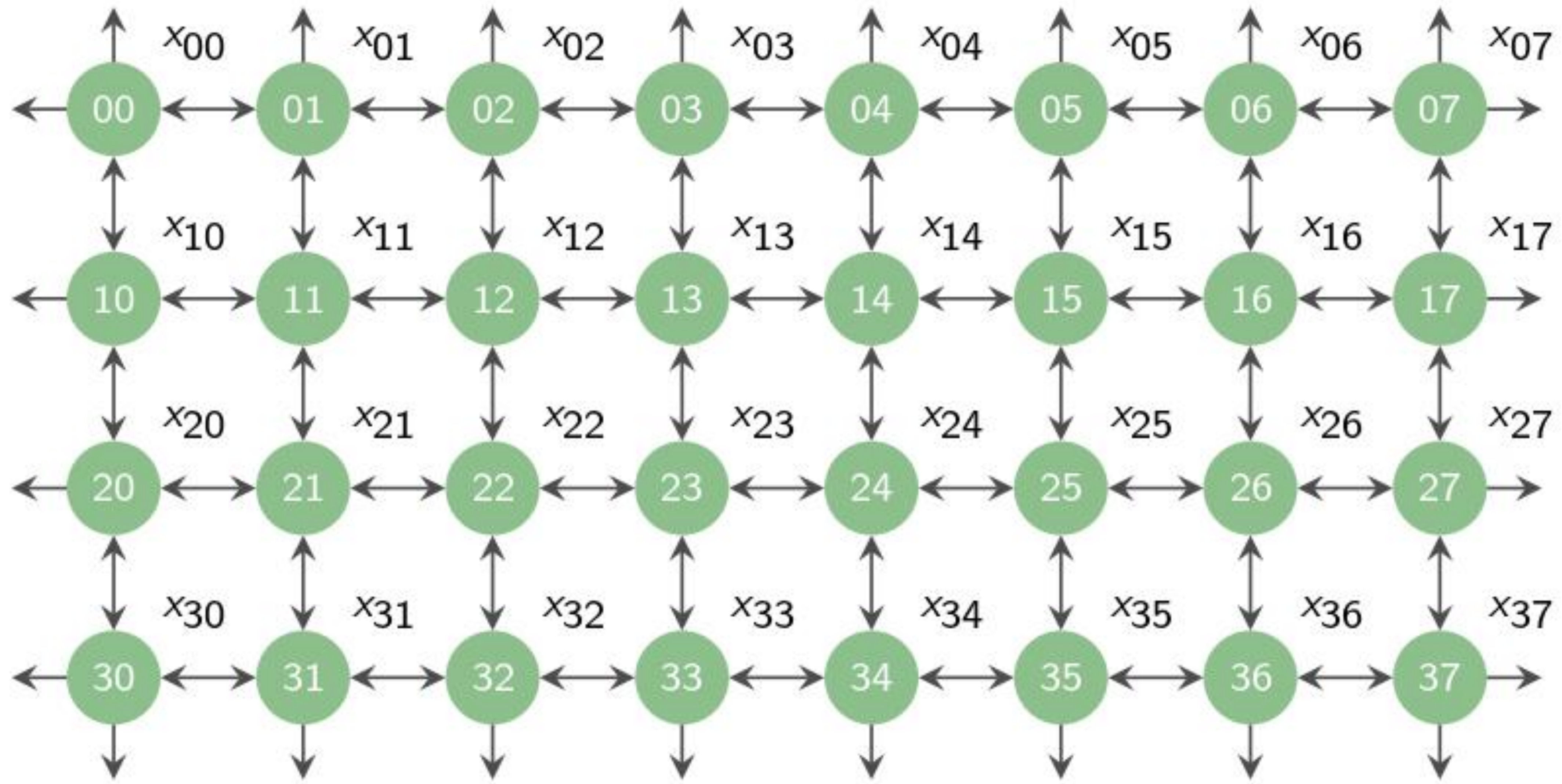
$\Rightarrow$ Even though we do not often think of them as such, convolutions are operations on graphs

- We can describe discrete time and space using graphs that support time or space signals

Description of time with a directed line graph



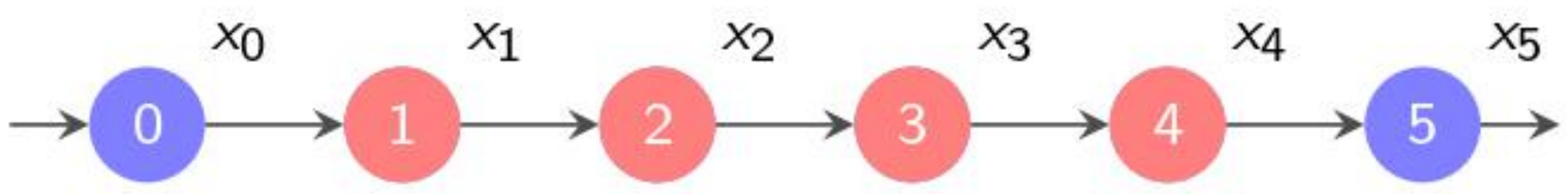
Description of images (space) with a grid graph



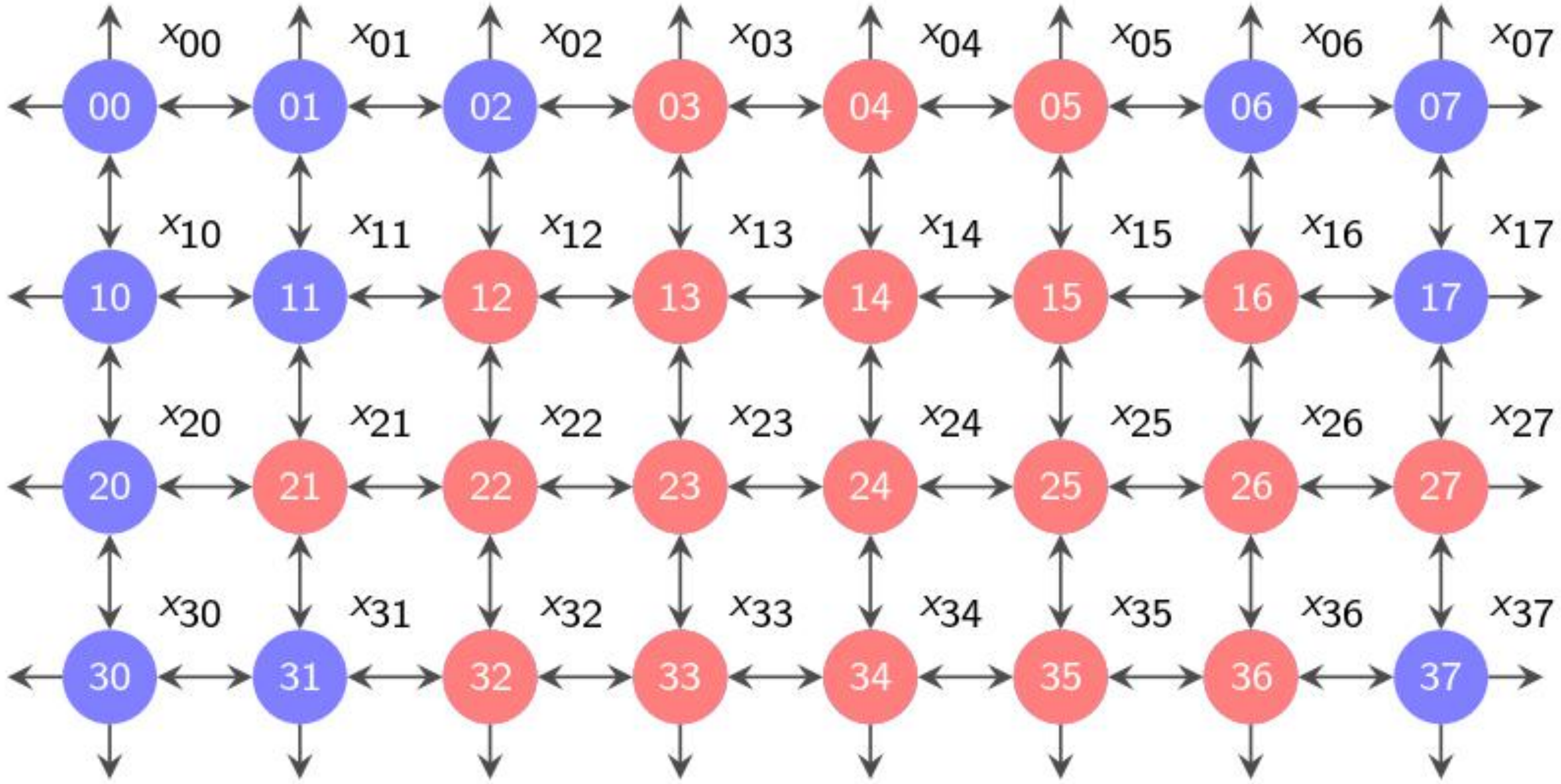
- Line graph represents adjacency of points in time. Grid graph represents adjacency of points in space

- Use line and grid graphs to write convolutions as polynomials on respective adjacency matrices $\mathbf{S}$

Description of time with a directed line graph



Description of images (space) with a grid graph

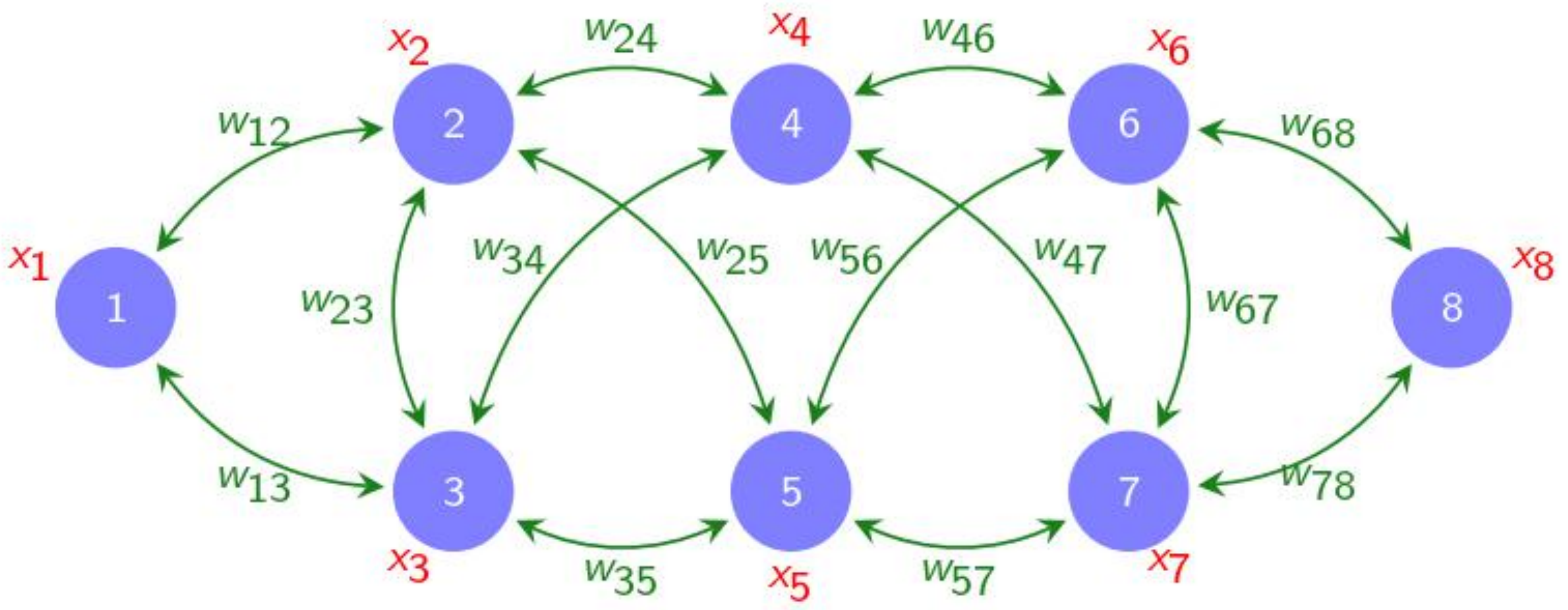


- Filter with coefficients h_k $\Rightarrow$ Output $\mathbf{z} = h_0 \mathbf{S}^0\mathbf{x} + h_1 \mathbf{S}^1\mathbf{x} + h_2 \mathbf{S}^2\mathbf{x} + h_3 \mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k \mathbf{S}^k\mathbf{x}$

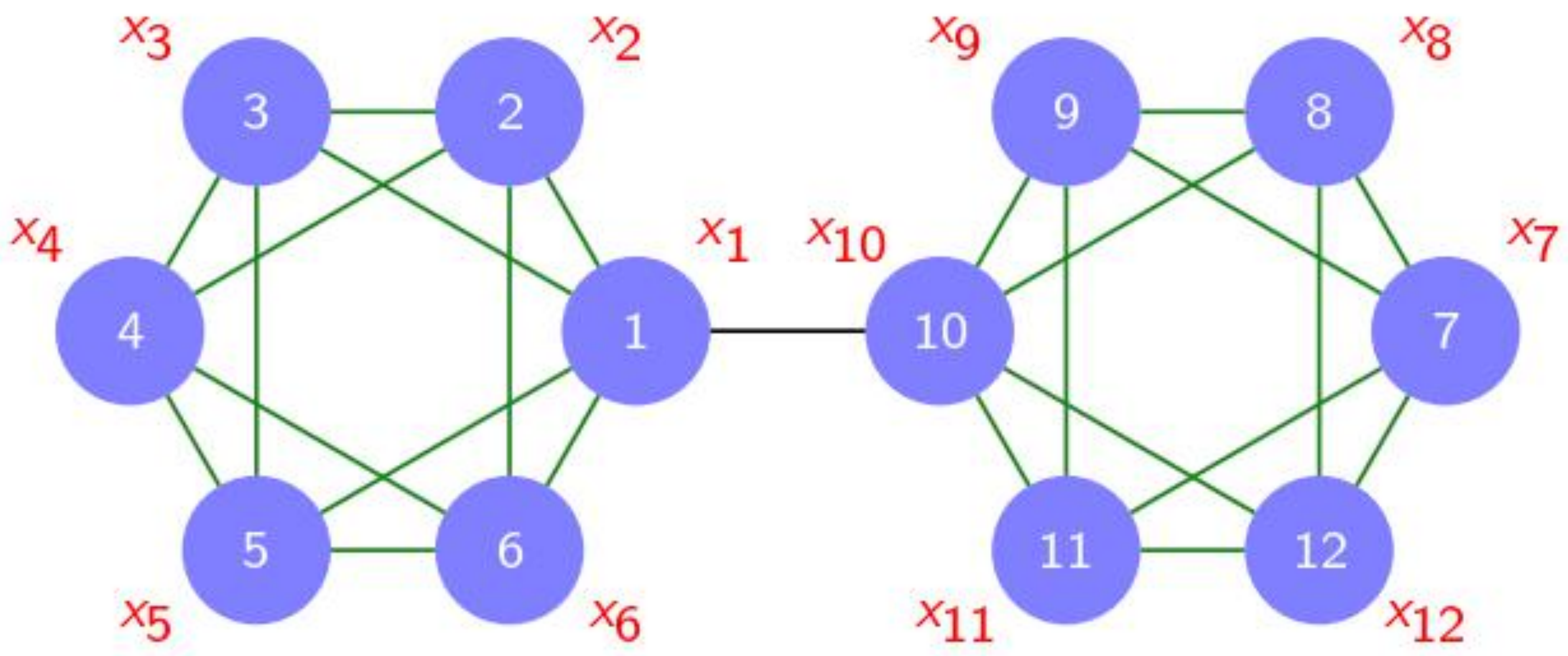
- Time and Space are pervasive and important, but still a (very) limited class of signals

- Use graphs as generic descriptors of signal structure with signal values associated to nodes and edges expressing expected similarity between signal components

A signal supported on a graph



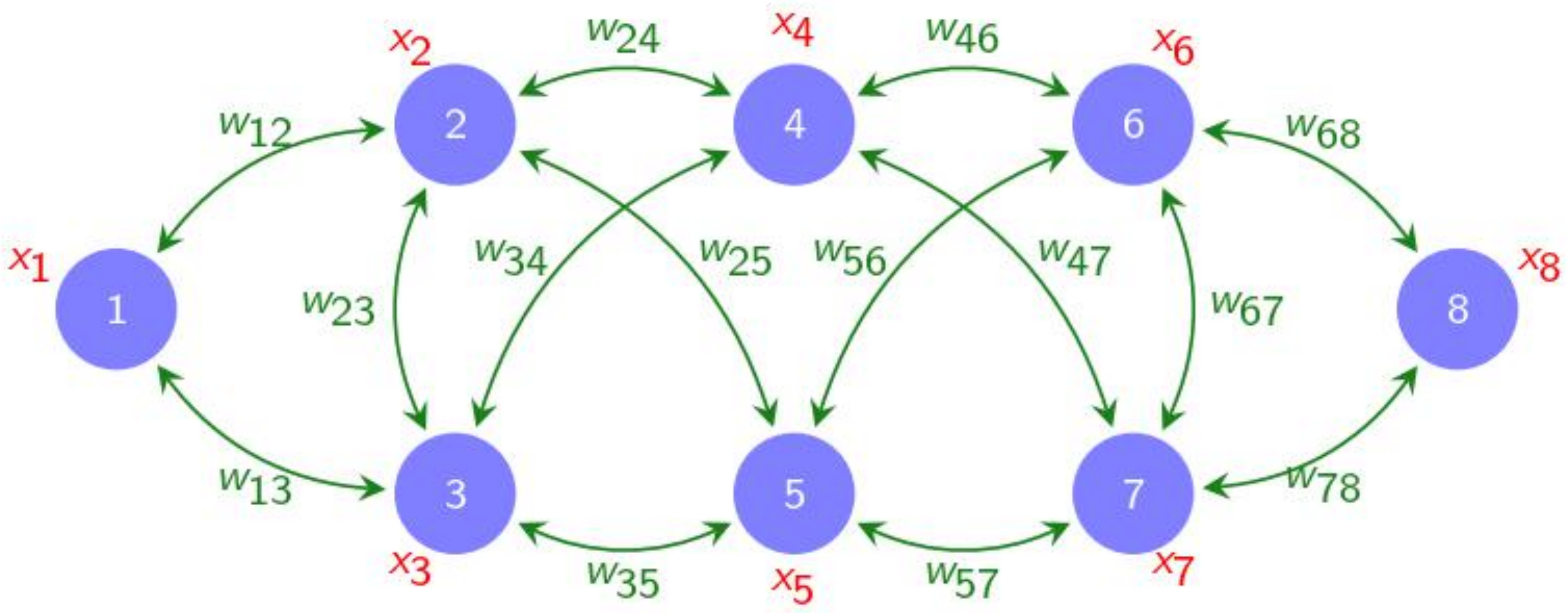
Another signal supported on another graph



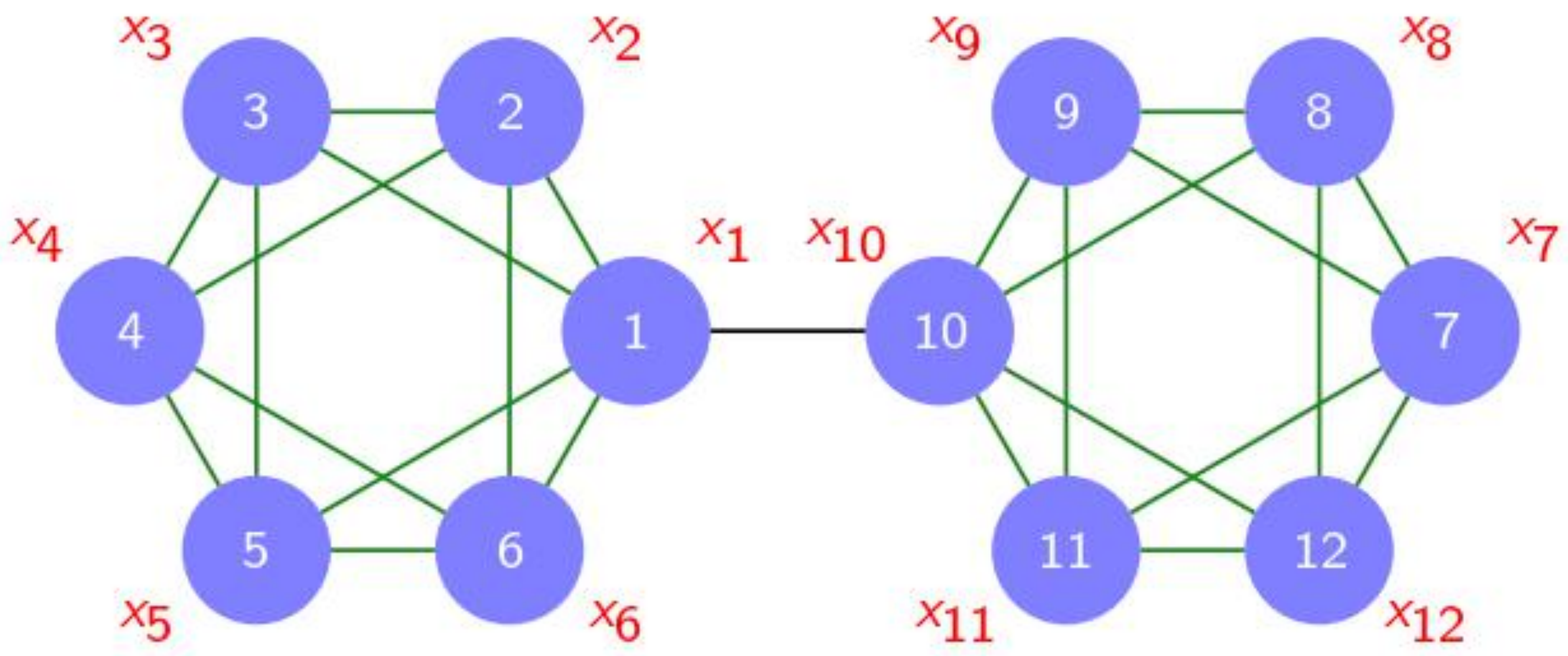
- Nodes are customers. Signal values are product ratings. Edges are cosine similarities of past scores

- Time and Space are pervasive and important, but still a (very) limited class of signals
- Use graphs as generic descriptors of signal structure with signal values associated to nodes and edges expressing expected similarity between signal components

A signal supported on a graph



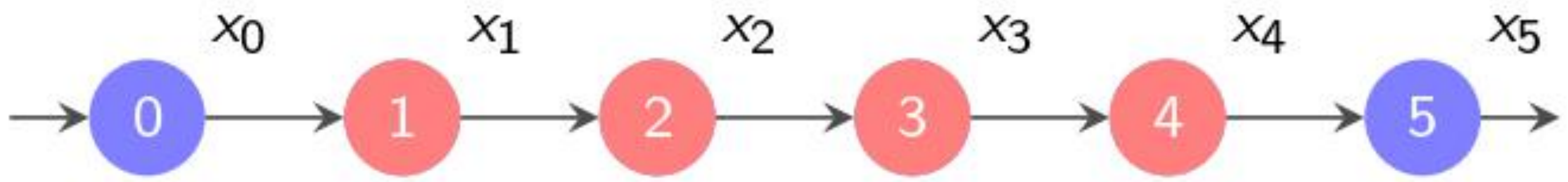
Another signal supported on another graph



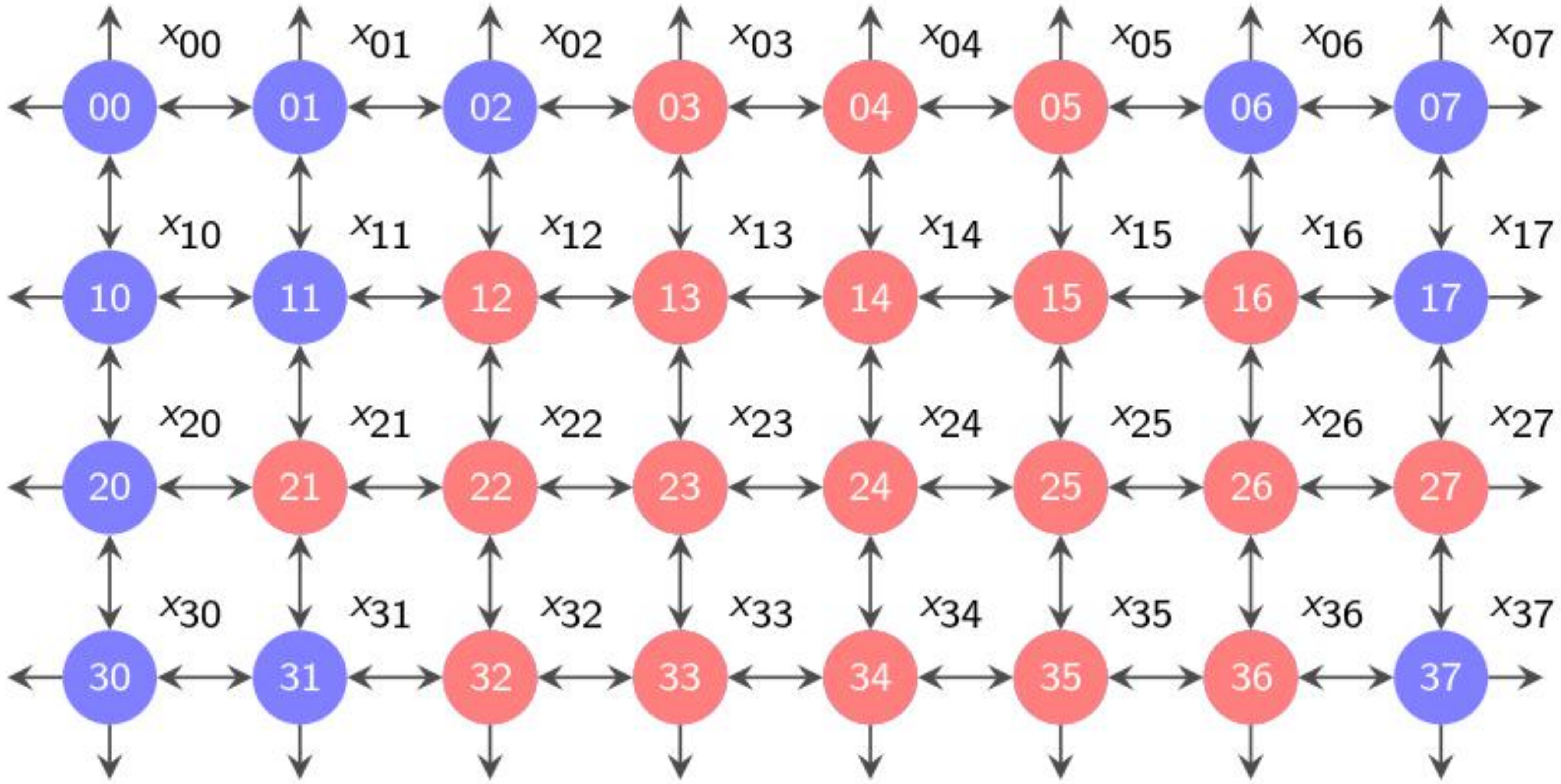
- Nodes are drones. Signal values are velocities. Edges are sensing and communication ranges

- We've already seen that convolutions in time and space are polynomials on adjacency matrices

Description of time with a directed line graph



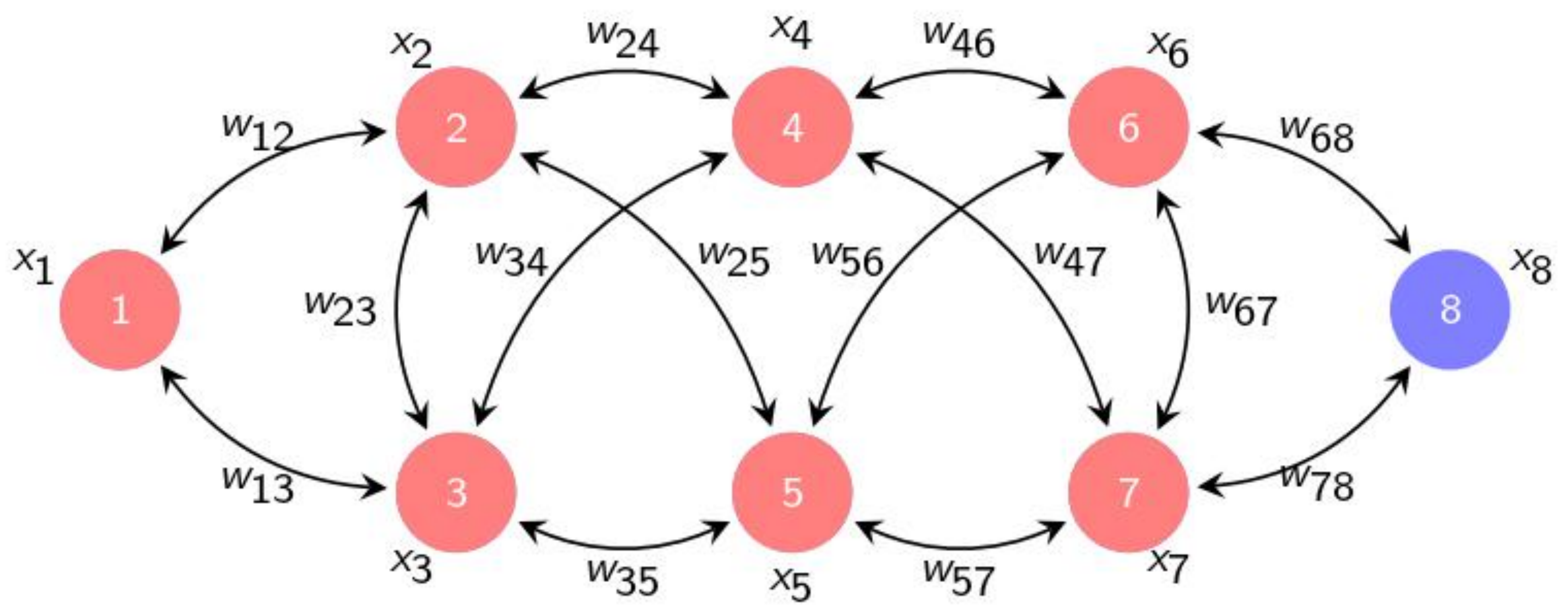
Description of images (space) with a grid graph



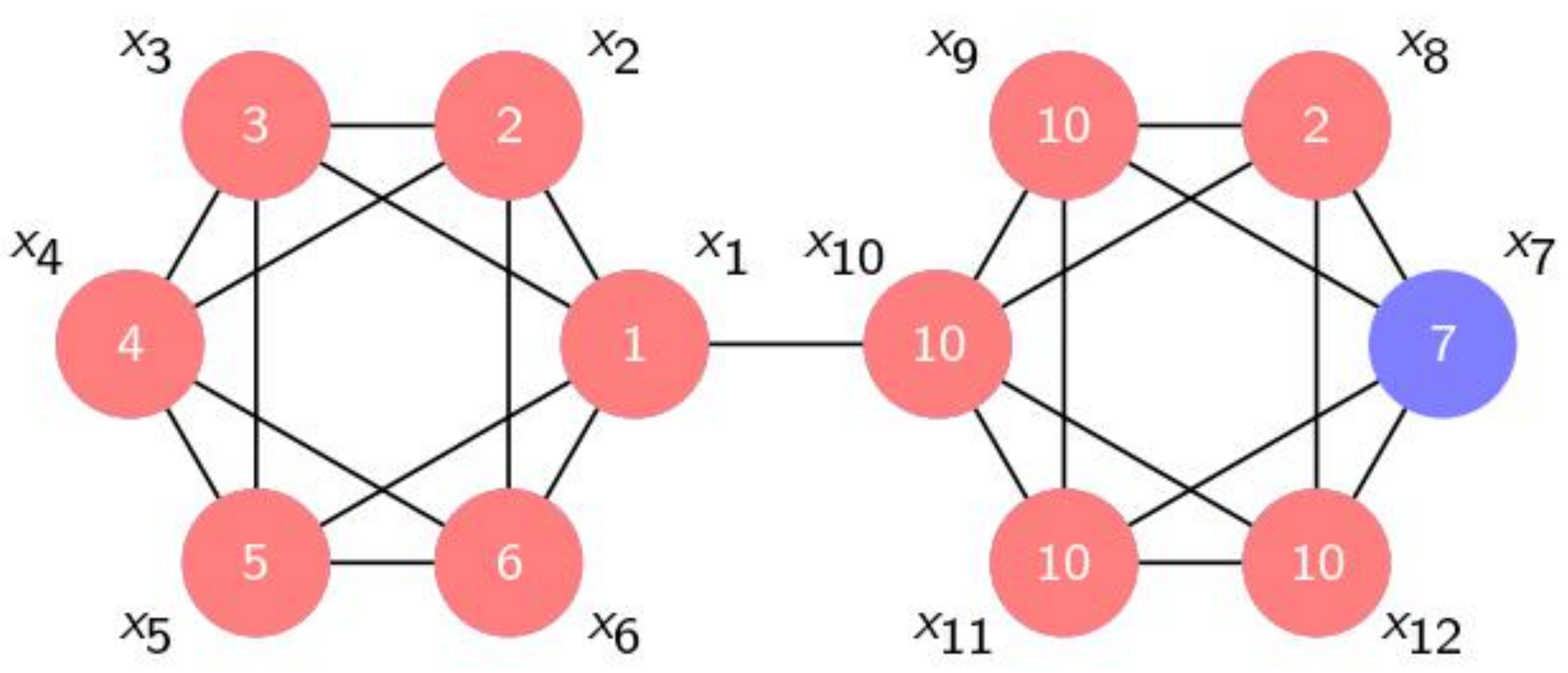
- Filter with coefficients h_k $\Rightarrow$ Output $\mathbf{z} = \; h_0\,\mathbf{S}^0\mathbf{x} \; + h_1\,\mathbf{S}^1\mathbf{x} \; + h_2\,\mathbf{S}^2\mathbf{x} \; + h_3\,\mathbf{S}^3\mathbf{x} \; + \ldots = \sum_{k=0}^{\infty} h_k\,\mathbf{S}^k\mathbf{x}$

- For graph signals we define graph convolutions as polynomials on matrix representations of graphs

A signal supported on a graph



Another signal supported on another graph



- Filter with coefficients $h_k \Rightarrow$ Output $\mathbf{z} = h_0\,\mathbf{S}^0\mathbf{x} + h_1\,\mathbf{S}^1\mathbf{x} + h_2\,\mathbf{S}^2\mathbf{x} + h_3\,\mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k\,\mathbf{S}^k\mathbf{x}$

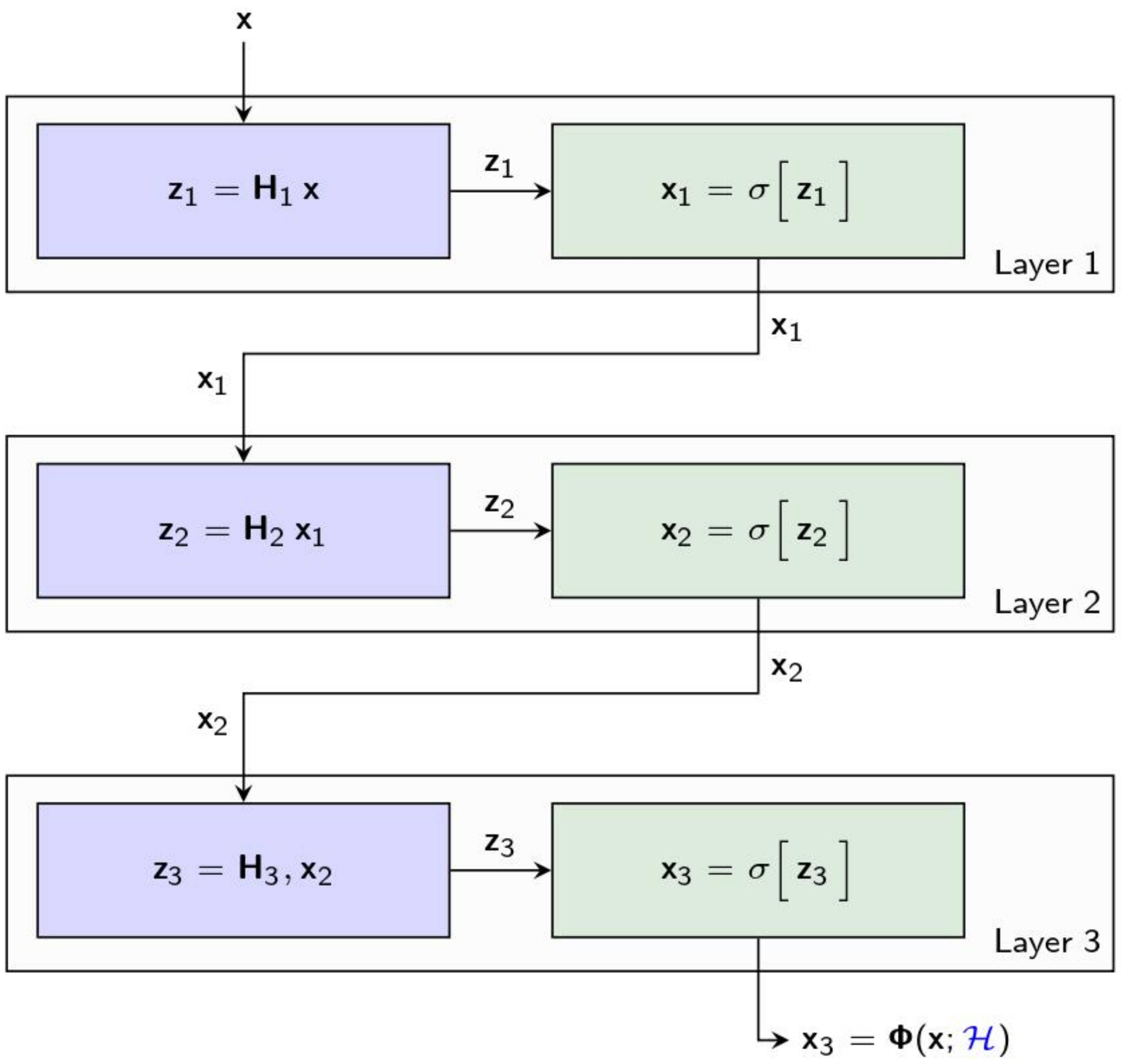
- Graph convolutions share the locality of conventional convolutions. Recovered as particular case

Convolutional Neural Networks and Graph Neural Networks

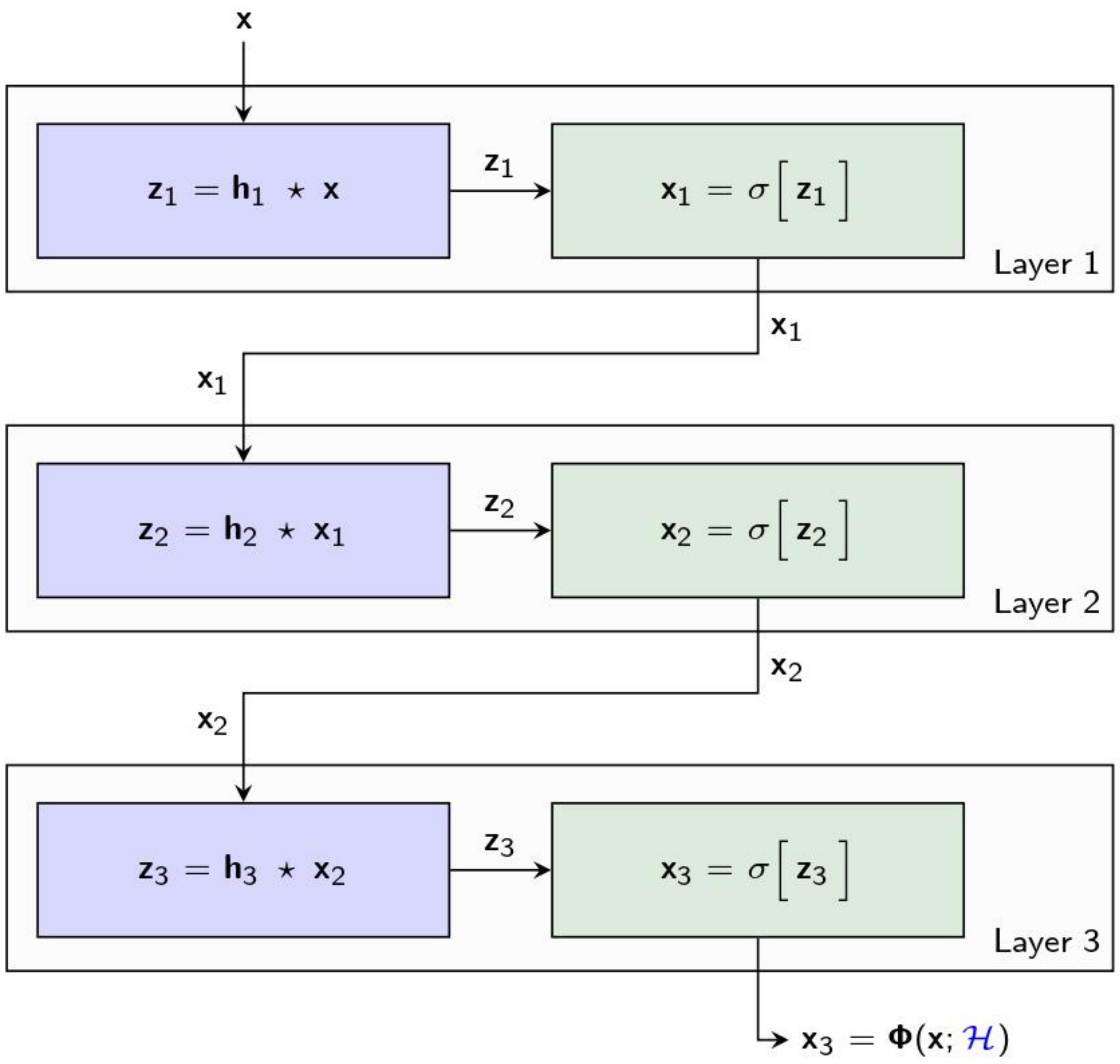
- CNNs and GNNe are minor variations of linear convolutional filters

  $\Rightarrow$ Compose filters with pointwise nonlinearities and compose these compositions into several layers

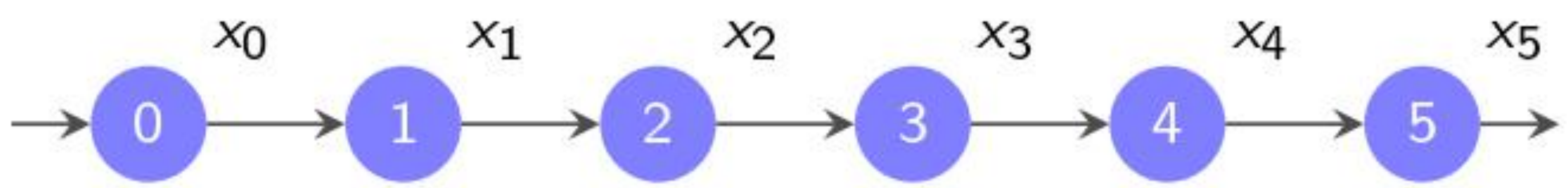
- A neural network composes a cascade of layers
- Each of which are themselves compositions of linear maps with pointwise nonlinearities
- Does not scale to large dimensional signals $\mathbf{x}$



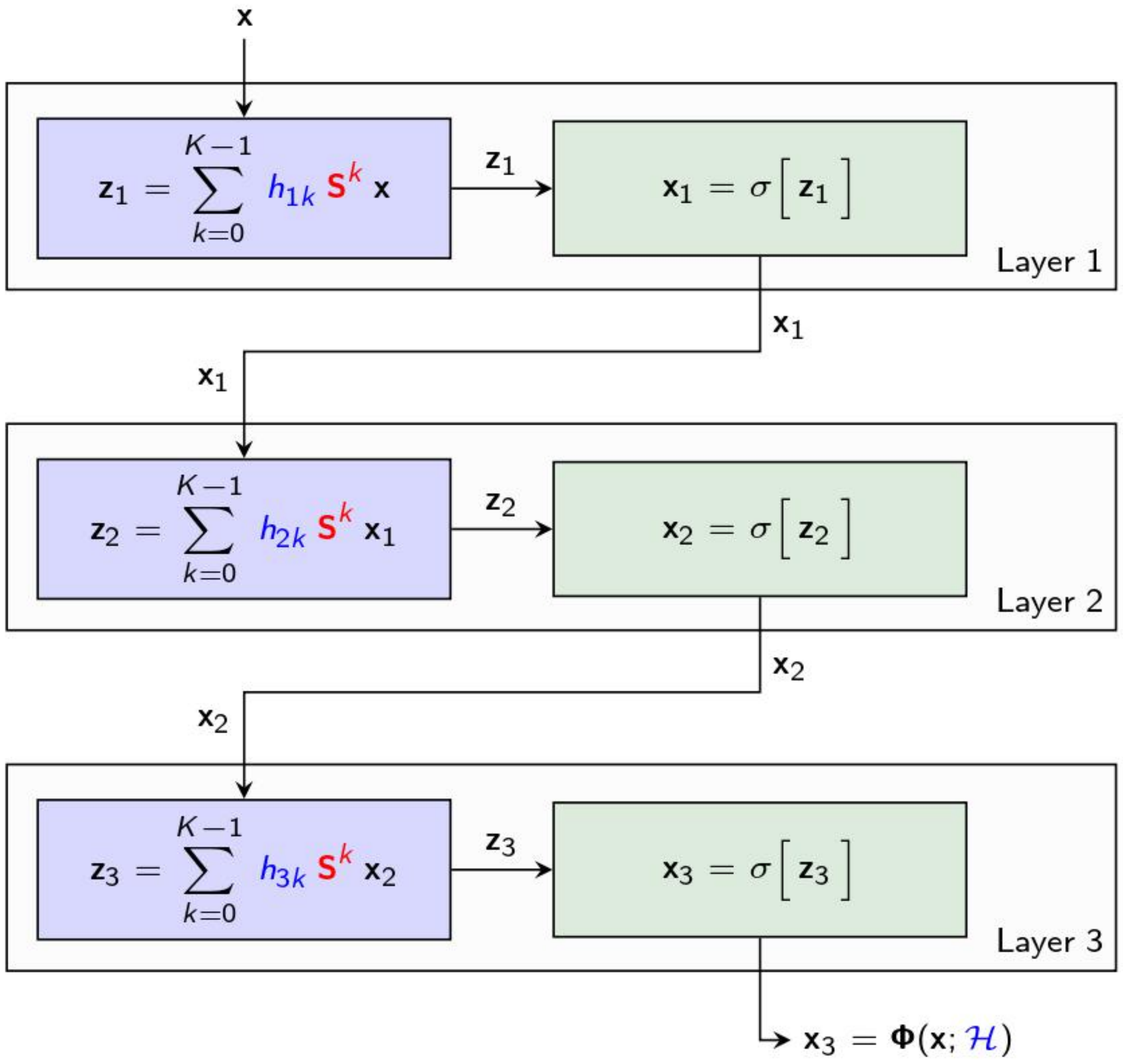
- A convolutional NN composes a cascade of layers
- Each of which are themselves compositions of convolutions with pointwise nonlinearities
- Scales well. The Deep Learning workhorse
- A CNNs are minor variation of convolutional filters
  - $\Rightarrow$ Just add nonlinearity and compose
  - $\Rightarrow$ They scale because convolutions scale



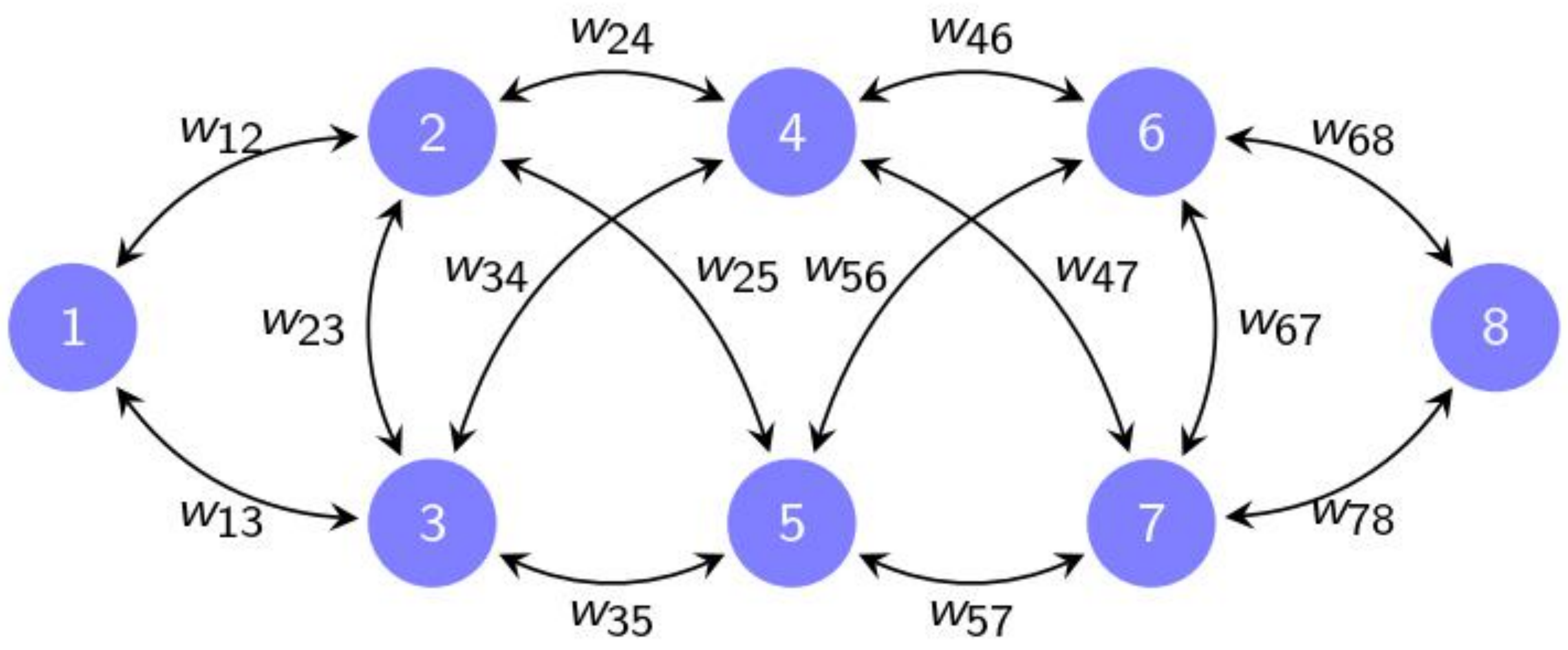
- Those convolutions are polynomials on the adjacency matrix of a line graph

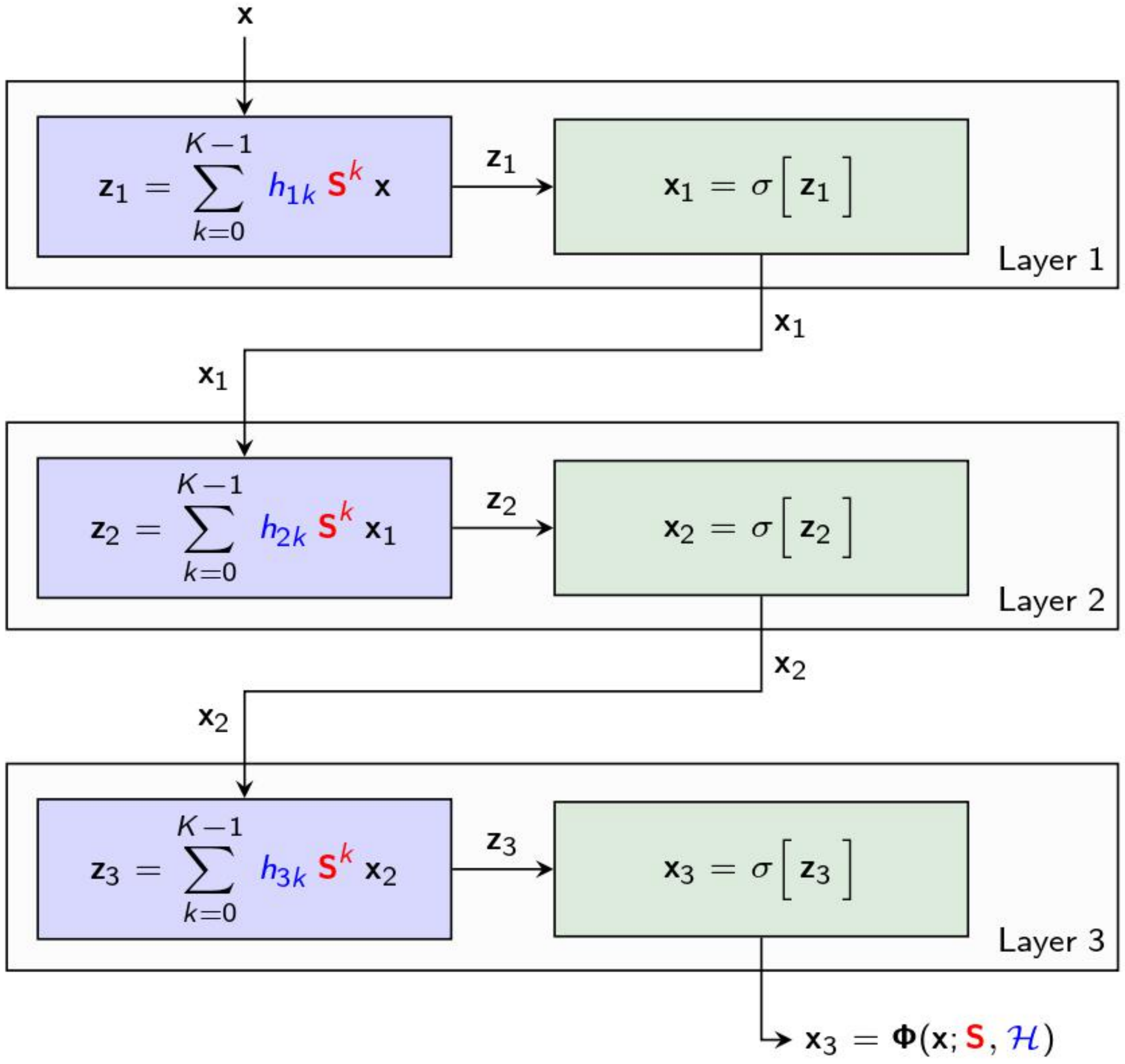


- Just another way of writing convolutions and Just another way of writing CNNs
- But one that lends itself to generalization



- The graph can be any arbitrary graph
- The polynomial on the matrix representation $\mathbf{S}$ becomes a graph convolutional filter



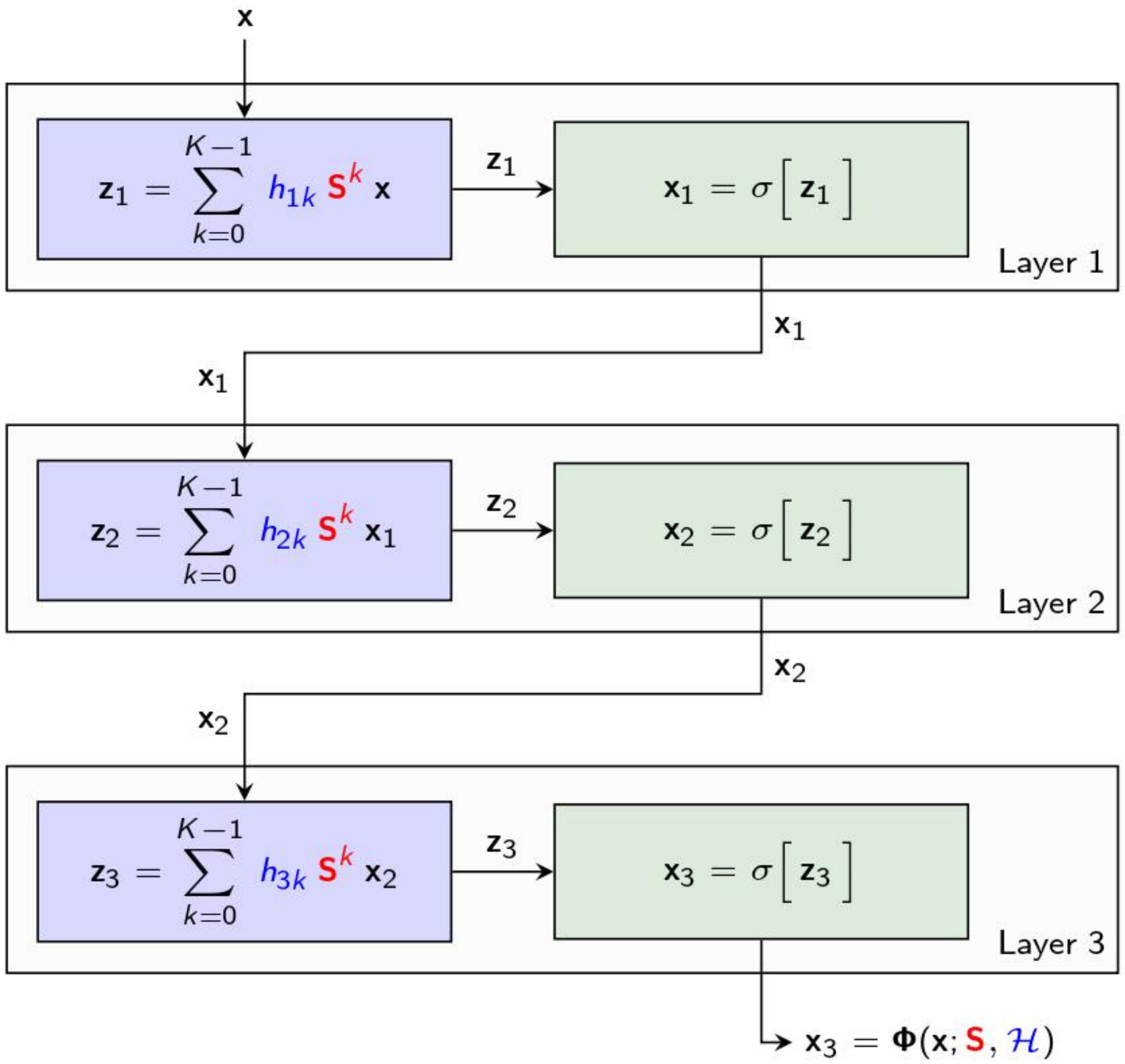


- A graph NN composes a cascade of layers

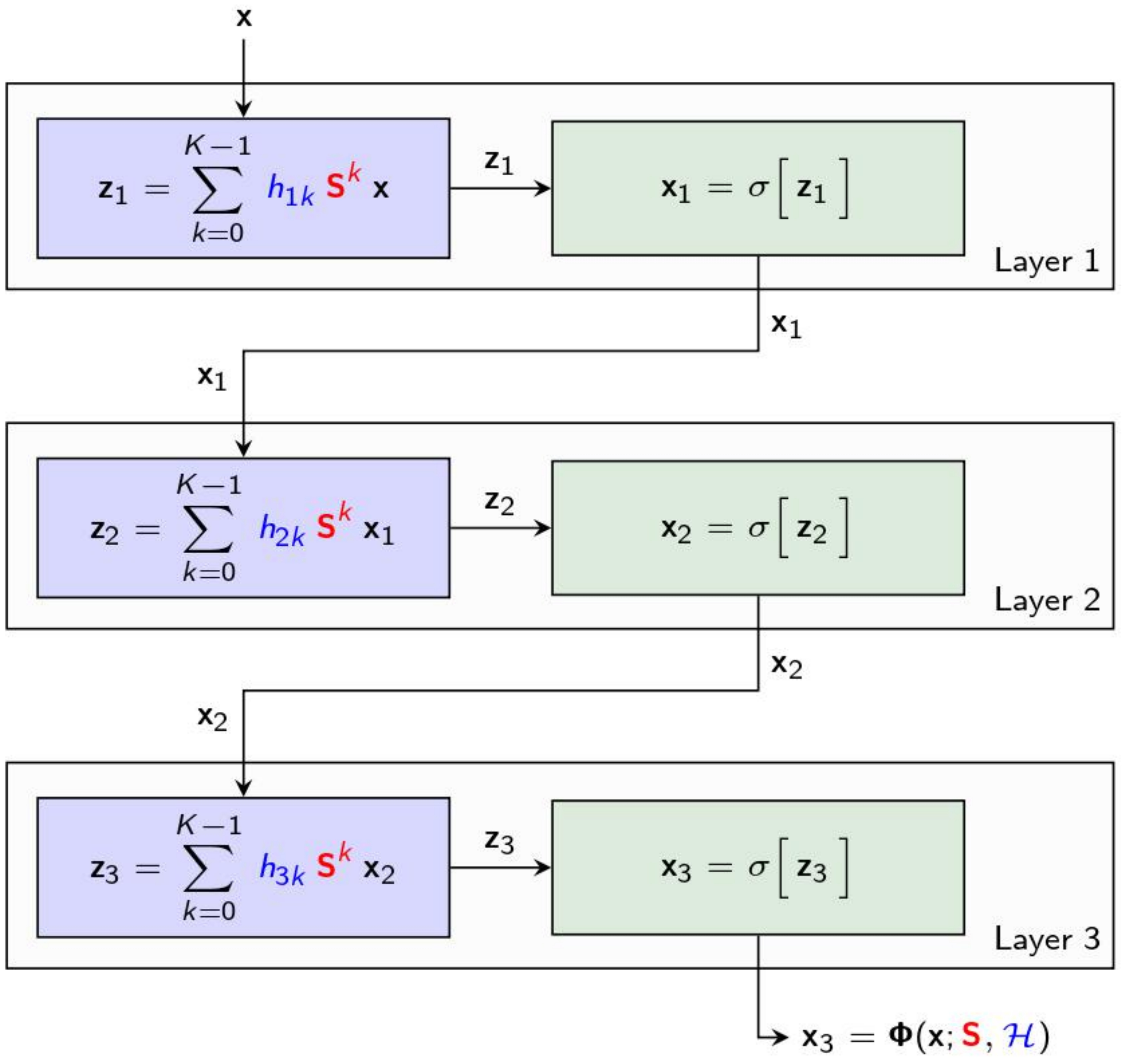
- Each of which are themselves compositions of graph convolutions with pointwise nonlinearities

- A NN with linear maps restricted to convolutions

- Recovers a CNN if **S** describes a line graph



- There is growing evidence of scalability.
- A GNN is a minor variation of a graph filter
  - $\Rightarrow$ Just add nonlinearity and compose
- Both are scalable because they leverage the signal structure codified by the graph



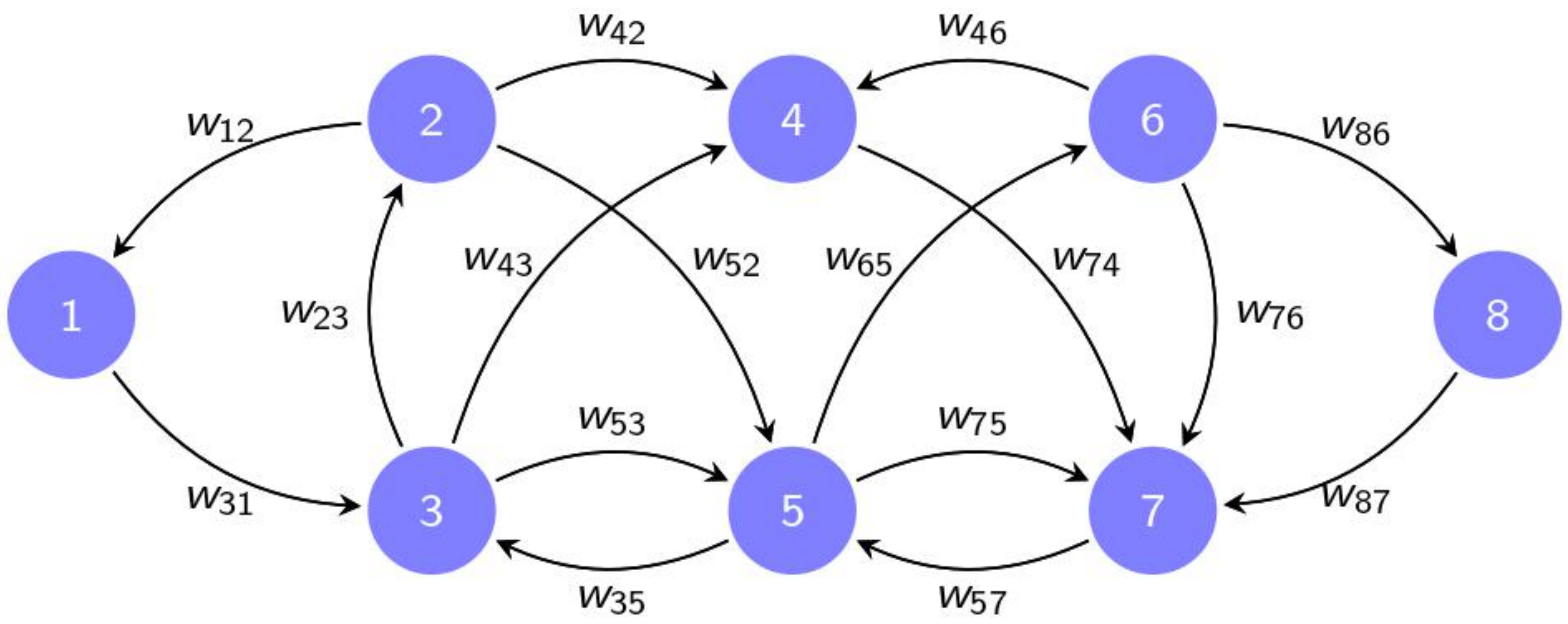
Graphs

- A graph is a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, which includes vertices $\mathcal{V}$, edges $\mathcal{E}$, and weights $\mathcal{W}$

  $\Rightarrow$ Vertices or nodes are a set of n labels. Typical labels are $\mathcal{V} = \{1, \ldots, n\}$

  $\Rightarrow$ Edges are ordered pairs of labels (i, j). We interpret $(i, j) \in \mathcal{E}$ as "i can be influenced by j."

  $\Rightarrow$ Weights $w_{ij} \in \mathbb{R}$ are numbers associated to edges (i, j). "Strength of the influence of j on i."

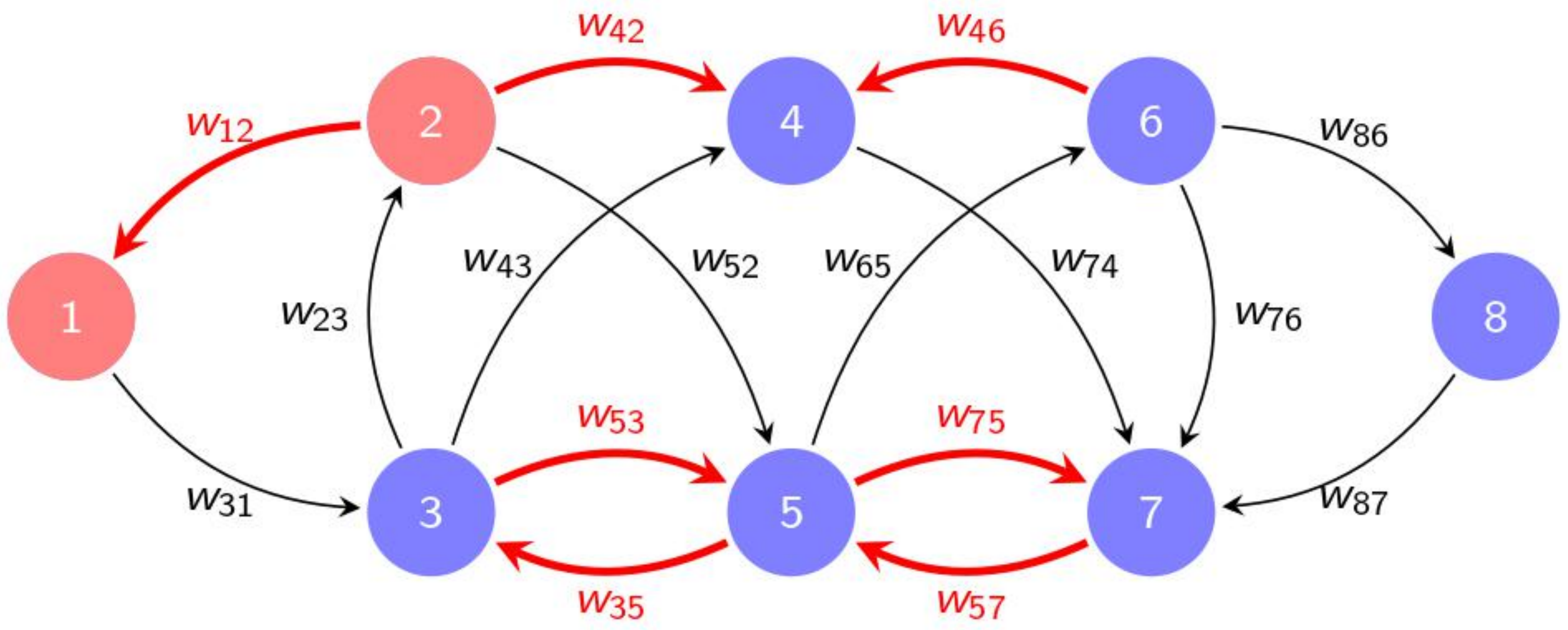


- Edge (i, j) is represented by an arrow pointing from j into i. Influence of node j on node i

  $\Rightarrow$ This is the opposite of the standard notation used in graph theory

- Edge (i, j) is different from edge (j, i) $\Rightarrow$ It is possible to have $(i, j) \in .\mathcal{E}$ and $(j, i) \notin \mathcal{E}$

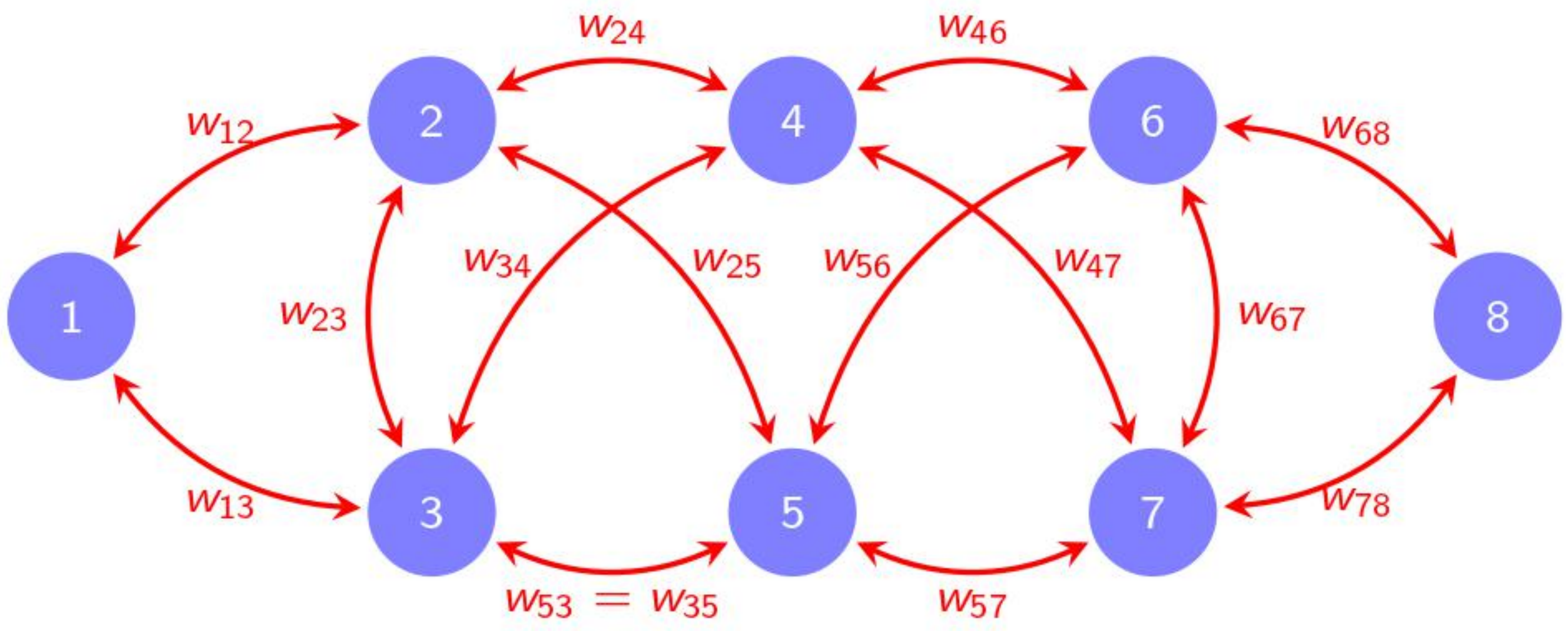
- If both edges are in the edge set, the weights can be different $\Rightarrow$ It is possible to have $w_{ij} \neq w_{ji}$



- A graph is symmetric or undirected if both, the edge set and the weight are symmetric

$\Rightarrow$ Edges come in pairs $\Rightarrow$ We have $(i, j) \in \mathcal{E}$ if and only if $(j, i) \in \mathcal{E}$

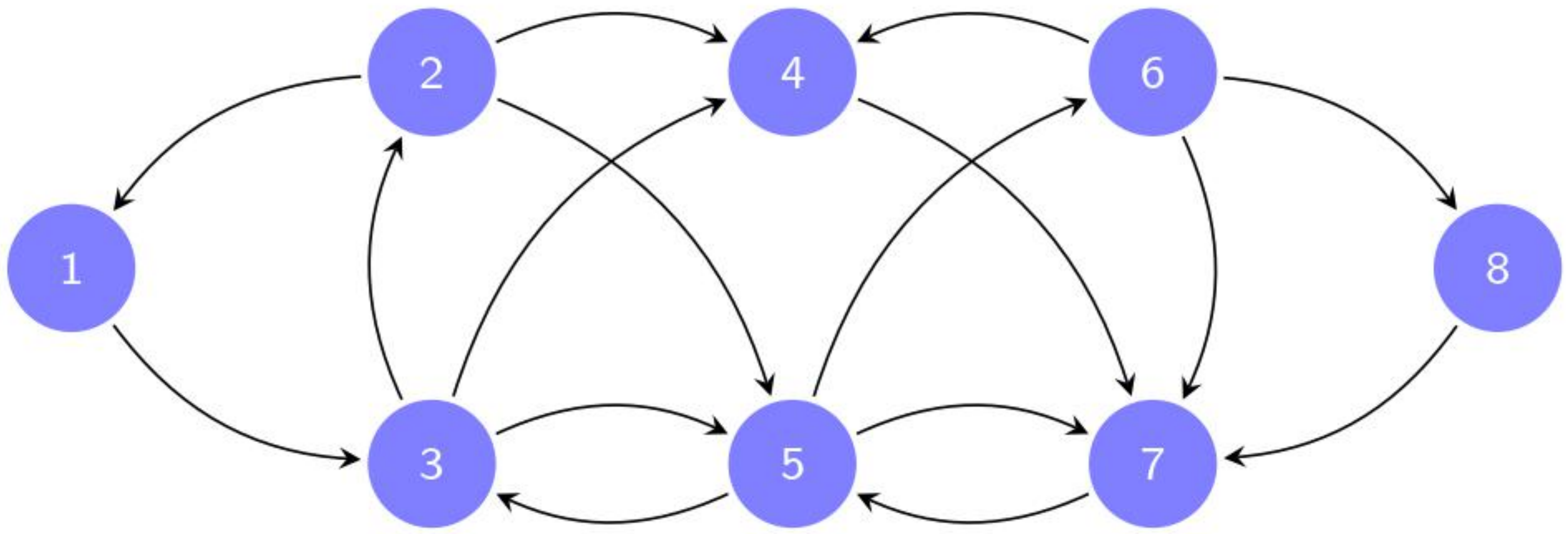
$\Rightarrow$ Weights are symmetric $\Rightarrow$ We must have $w_{ij} = w_{ji}$ for all $(i, j) \in \mathcal{E}$



- A graph is unweighted if it doesn't have weights

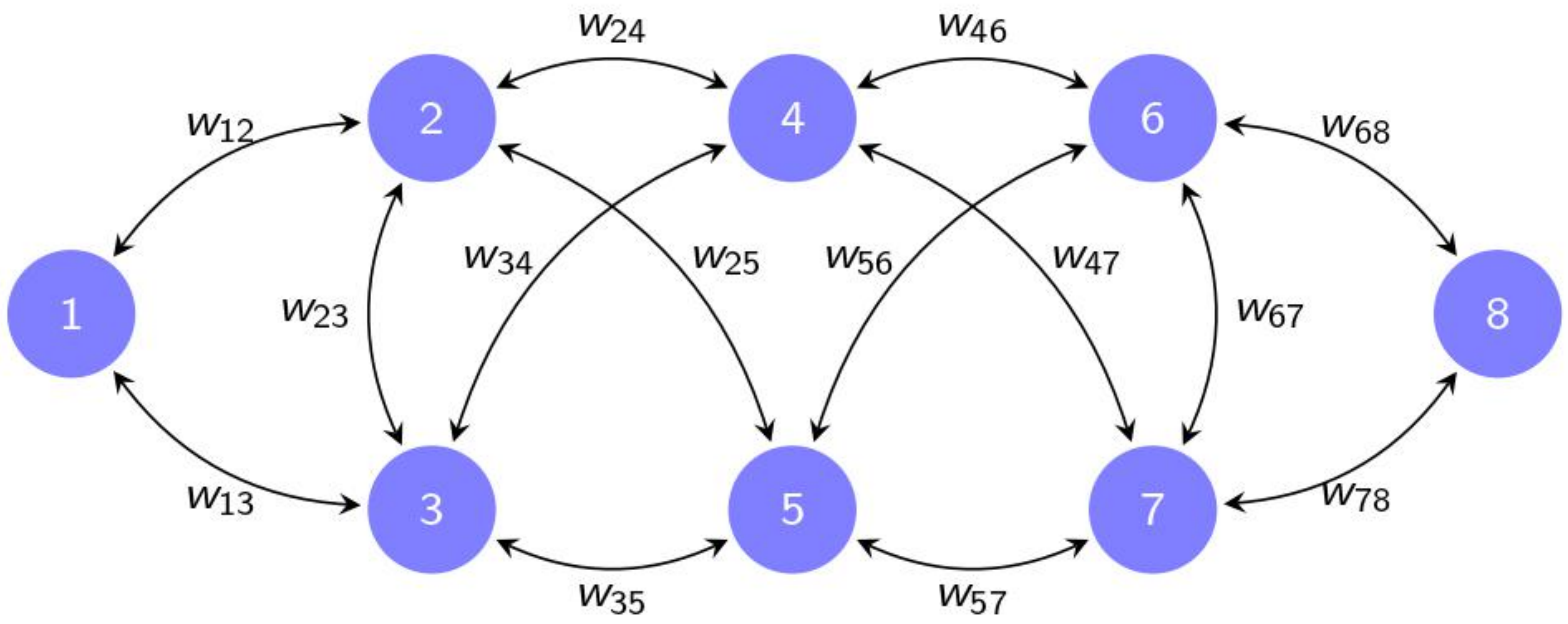
  $\Rightarrow$ Equivalently, we can say that all weights are units $\Rightarrow w_{ij} = 1$ for all $(i, j) \in \mathcal{E}$

- Unweighted graphs could be directed or symmetric



- Graphs can be directed or symmetric. Separately, they can be weighted or unweighted.

- Most of the graphs we encounter in practical situations are symmetric and weighted



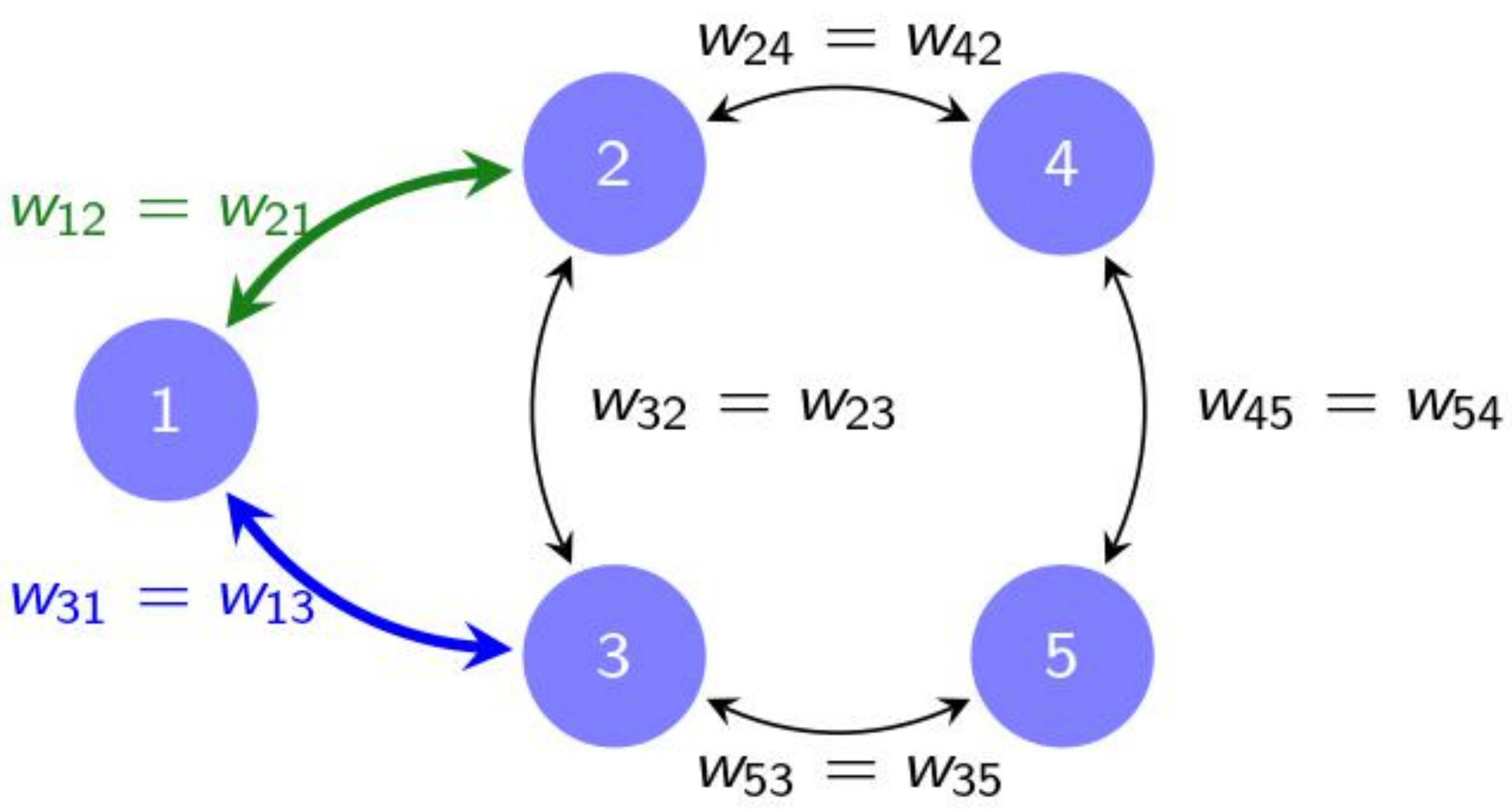
Graph Shift Operators

- Graphs have matrix representations. Which in this course, we call graph shift operators (GSOs)

- The adjacency matrix of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is the sparse matrix $\mathbf{A}$ with nonzero entries

$$A_{ij} = w_{ij}, \text{ for all } (i,j) \in \mathcal{E}$$

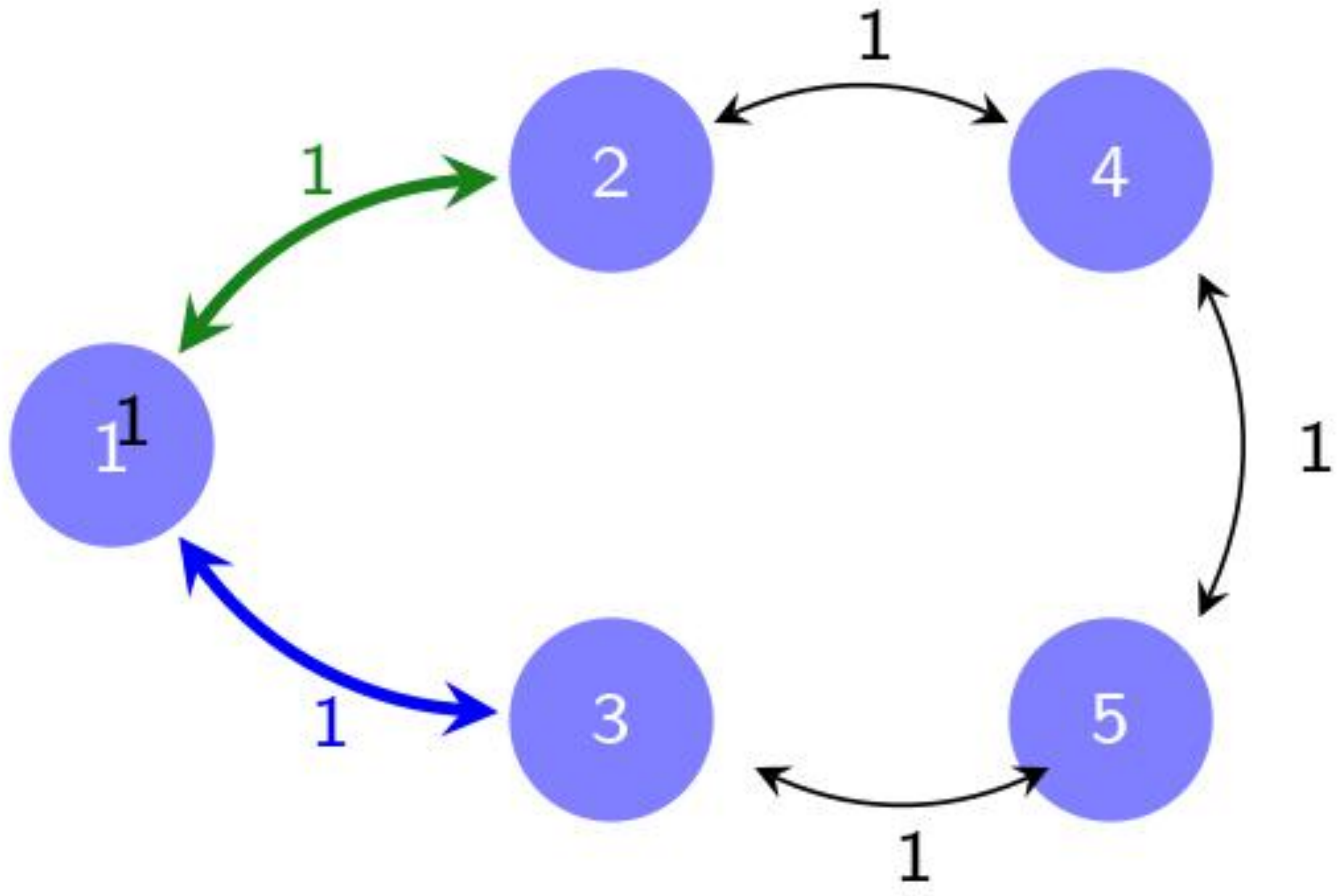
- If the graph is symmetric, the adjacency matrix is symmetric $\Rightarrow \mathbf{A} = \mathbf{A}^T$. As in the example



$$\mathbf{A} = \begin{bmatrix} 0 & w_{12} & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{23} & w_{24} & 0 \\ w_{31} & w_{32} & 0 & 0 & w_{35} \\ 0 & w_{42} & 0 & 0 & w_{45} \\ 0 & 0 & w_{53} & w_{54} & 0 \end{bmatrix}.$$

- For the particular case in which the graph is unweighted. Weights interpreted as units

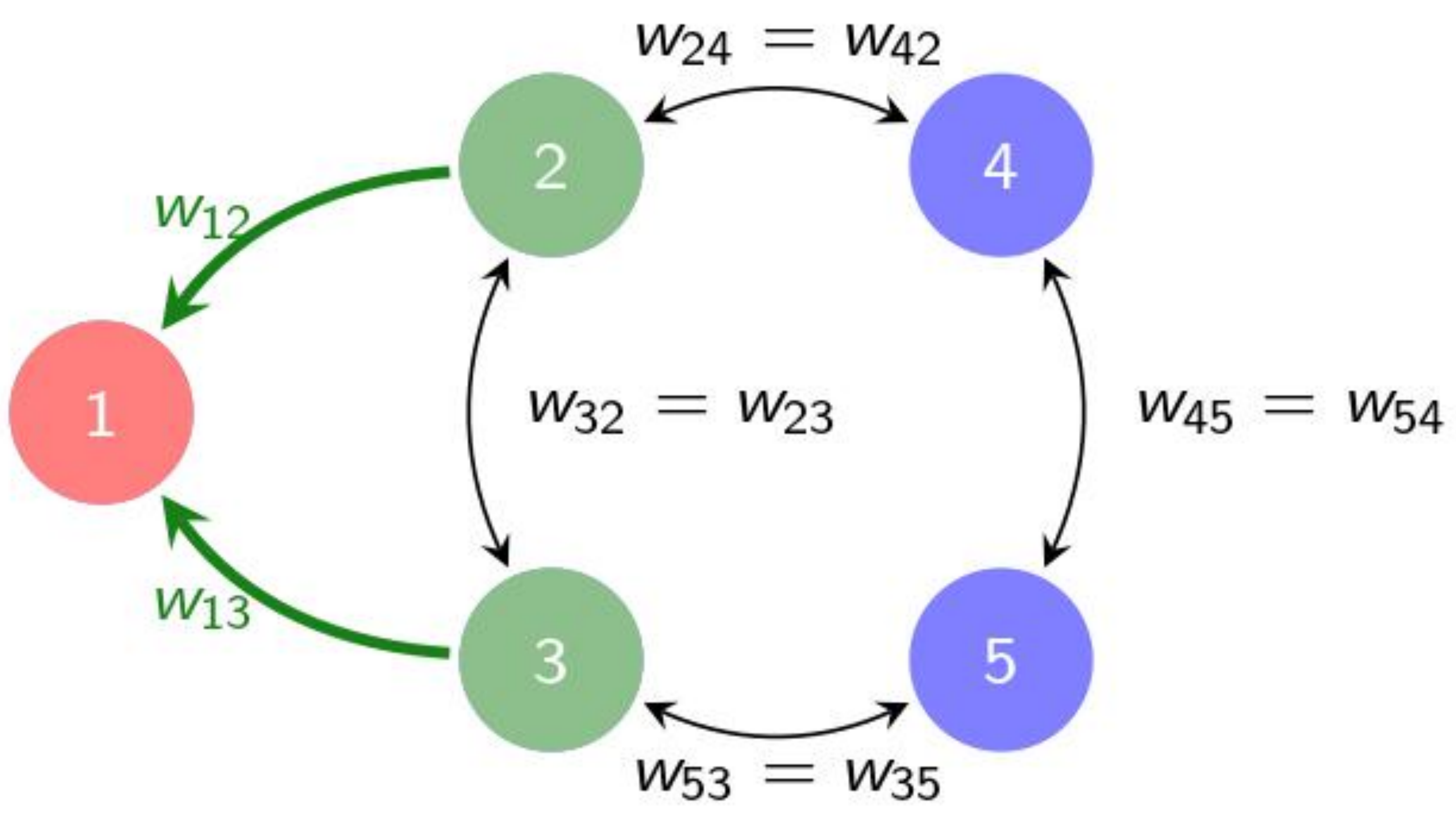
$$A_{ij} = 1, \qquad \text{for all} \quad (i,j) \in \mathcal{E}$$



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

- The neighborhood of node i is the set of nodes that influence i $\Rightarrow n(i) := \{j : (i, j) \in \mathcal{E}\}$

- Degree d_i of node i is the sum of the weights of its incident edges $\Rightarrow d_i = \sum_{j \in n(i)} w_{ij} = \sum_{j:(i,j) \in \mathcal{E}\}} w_{ij}$

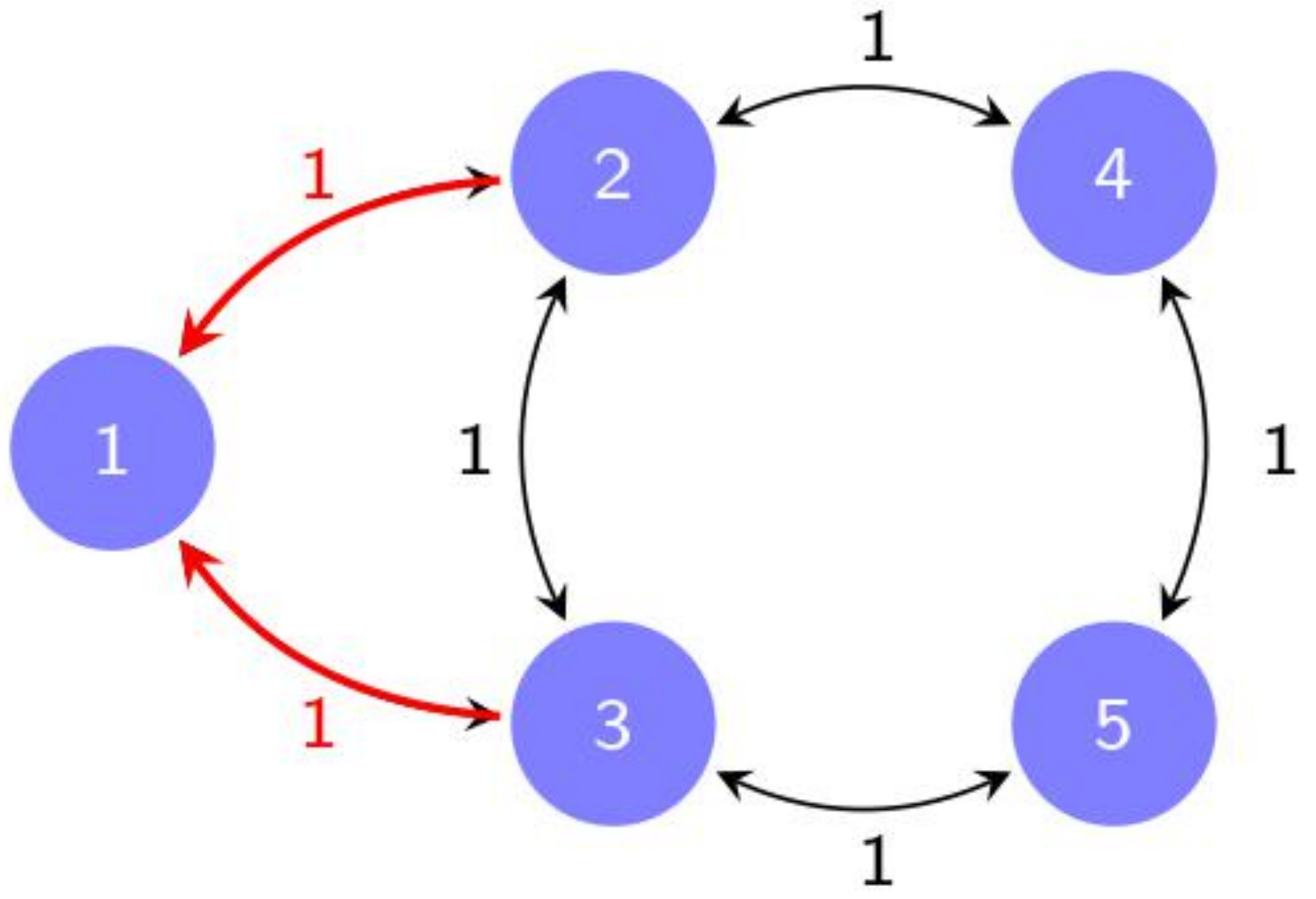


- Node 1 neighborhood $\Rightarrow n(1) = \{2, 3\}$

- Node 1 degree $\Rightarrow n(1) = w_{12} + w_{13}$

- The degree matrix is a diagonal matrix $\mathbf{D}$ with degrees as diagonal entries $\Rightarrow D_{ii} = d_i$

- Write in terms of adjacency matrix as $\mathbf{D} = \mathrm{diag}(\mathbf{A1})$. Because $(\mathbf{A1})_i = \sum_j w_{ij} = d_i$



$$\mathbf{D} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

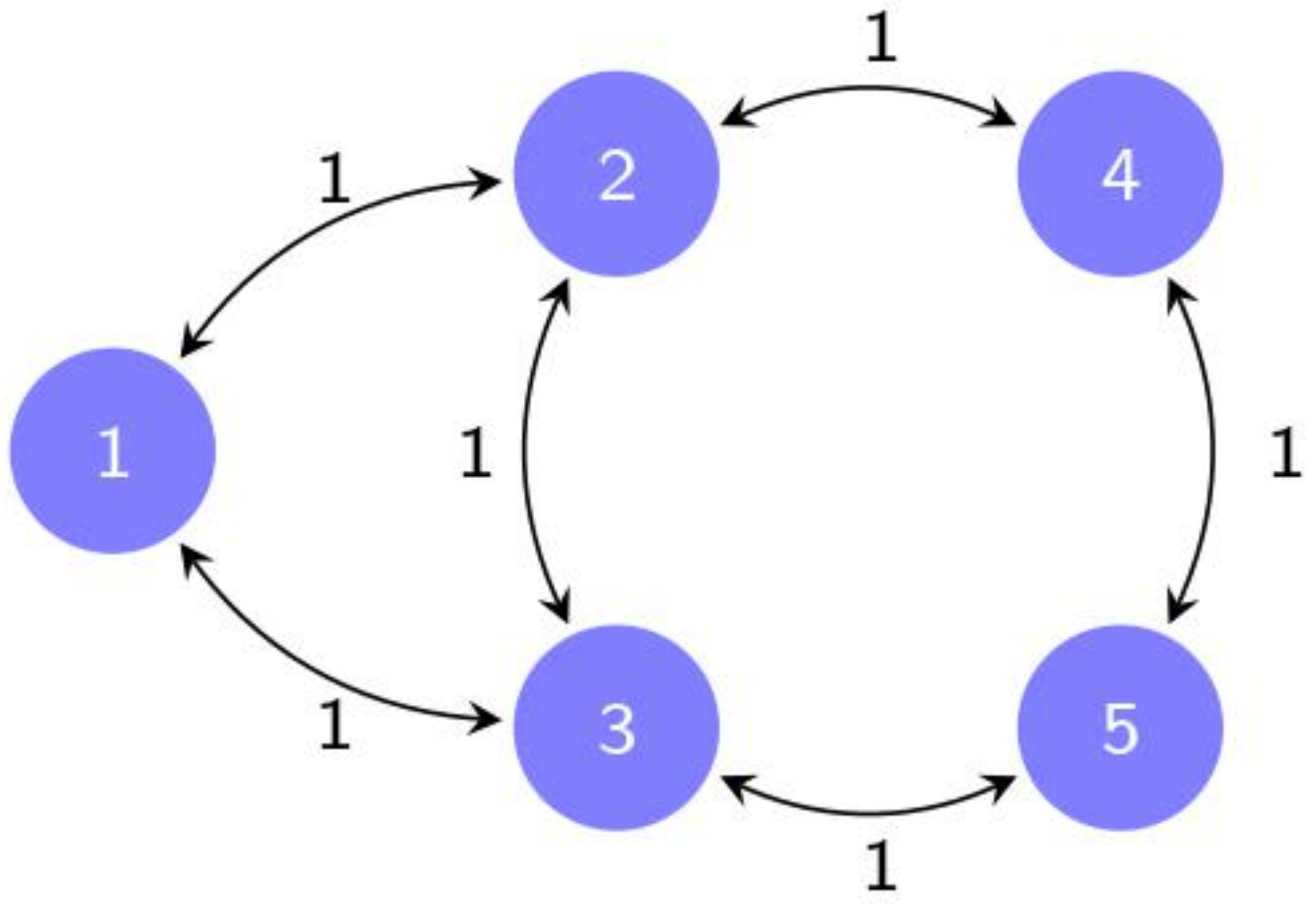
- The Laplacian matrix of a graph with adjacency matrix $\mathbf{A}$ is $\Rightarrow \mathbf{L} = \mathbf{D} - \mathbf{A} = \text{diag}(\mathbf{A1}) - \mathbf{A}$

- Can also be written explicitly in terms of graph weights $A_{ij} = w_{ij}$

$\Rightarrow$ Off diagonal entries $\Rightarrow L_{ij} = -A_{ij} = -w_{ij}$

$\Rightarrow$ Diagonal entries $\Rightarrow L_{ii} = d_i = \sum_{j \in n(i)} w_{ij}$

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$



- Normalized adjacency and Laplacian matrices express weights relative to the nodes' degrees

- Normalized adjacency matrix $\Rightarrow \bar{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \Rightarrow$ Results in entries $(\bar{\mathbf{A}})_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}$

- The normalized adjacency is symmetric if the graph is symmetric $\Rightarrow \bar{\mathbf{A}}^T = \bar{\mathbf{A}}$.

- Normalized Laplacian matrix $\Rightarrow \bar{\mathbf{L}} := \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. Same normalization of adjacency matrix

- Given definitions normalized representations $\Rightarrow \bar{\mathbf{L}} = \mathbf{D}^{-1/2}\Big(\mathbf{D} - \mathbf{A}\Big)\mathbf{D}^{-1/2} = \mathbf{I} - \bar{\mathbf{A}}$

  $\Rightarrow$ The normalized Laplacian and adjacency are essentially the same linear transformation.

- Normalized operators are more homogeneous. The entries in the vector $\mathbf{A1}$ tend to be similar.

- The Graph Shift Operator $\mathbf{S}$ is a stand in for any of the matrix representations of the graph

Adjacency Matrix	Laplacian Matrix	Normalized Adjacency	Normalized Laplacian
$\mathbf{S} = \mathbf{A}$	$\mathbf{S} = \mathbf{L}$	$\mathbf{S} = \bar{\mathbf{A}}$	$\mathbf{S} = \bar{\mathbf{L}}$

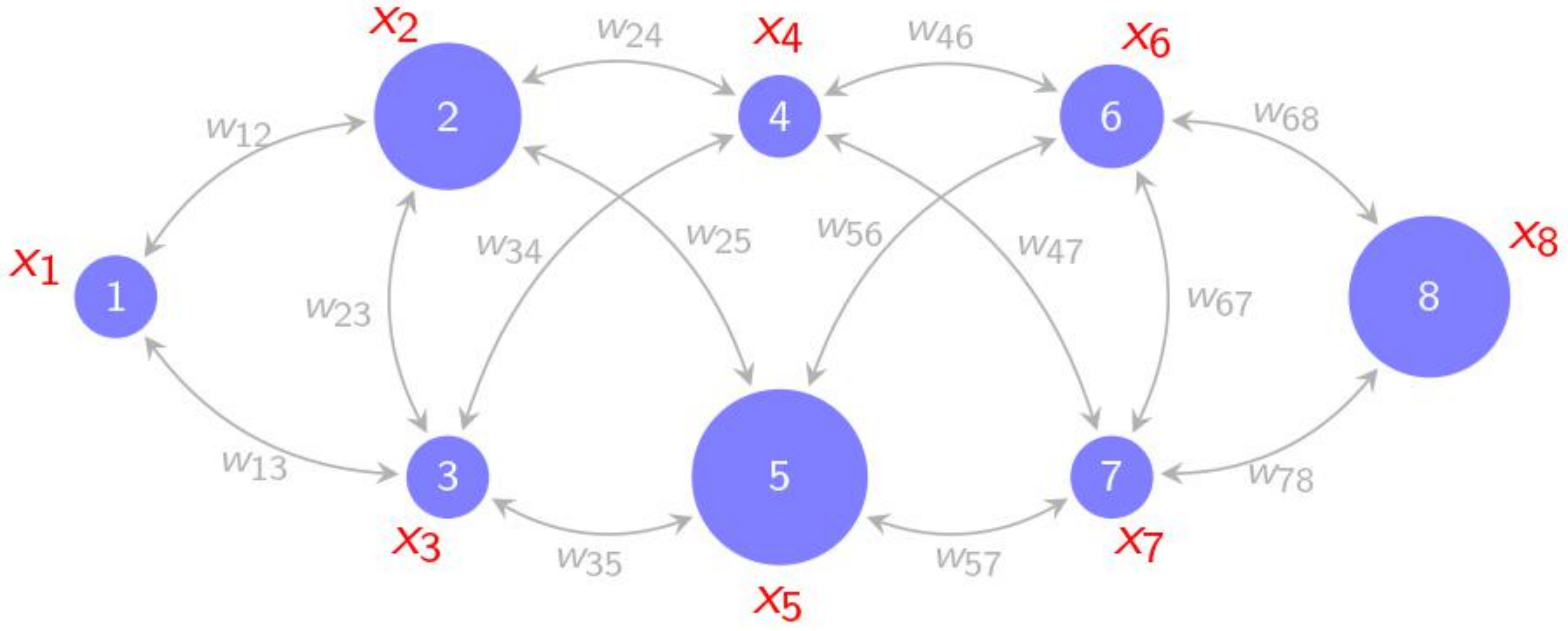
- If the graph is symmetric, the shift operator $\mathbf{S}$ is symmetric $\Rightarrow \mathbf{S} = \mathbf{S}^T$

- The specific choice matters in practice but most of results and analysis hold for any choice of $\mathbf{S}$

Graph Signals

- Graph Signals are supported on a graph. They are the objets we process in Graph Signal Processing

- Consider a given graph $\mathcal{G}$ with n nodes and shift operator $\mathbf{S}$
- A graph signal is a vector $\mathbf{x} \in \mathbb{R}^n$ in which component x_i is associated with node i
- To emphasize that the graph is intrinsic to the signal we may write the signal as a pair $\Rightarrow (\mathbf{S}, \mathbf{x})$



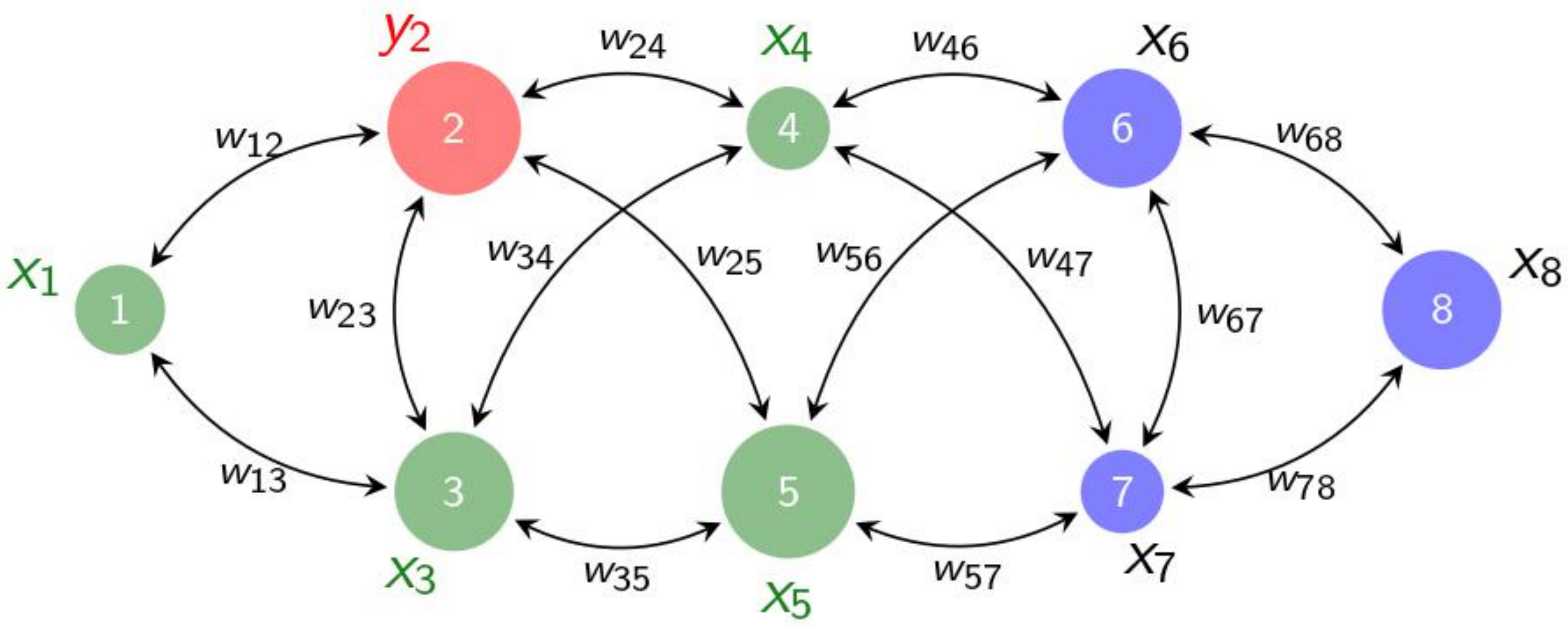
- The graph is an expectation of proximity or similarity between components of the signal $\mathbf{x}$

- Multiplication by the graph shift operator implements diffusion of the signal over the graph

- Define diffused signal $\mathbf{y} = \mathbf{S}\mathbf{x}$ $\Rightarrow$ Components are $y_i = \sum_{j \in n(i)} w_{ij} \, x_j = \sum_{j} w_{ij} \, x_j$

  $\Rightarrow$ Stronger weights contribute more to the diffusion output

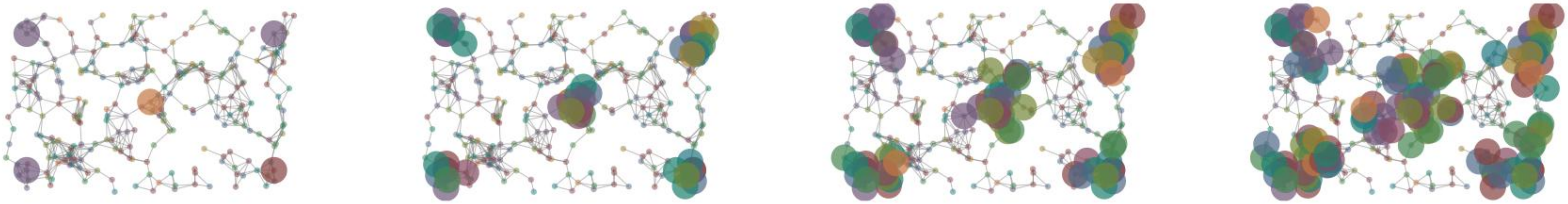
  $\Rightarrow$ Codifies a local operation where components are mixed with components of neighboring nodes.



- Compose the diffusion operator to produce diffusion sequence $\Rightarrow$ defined recursively as

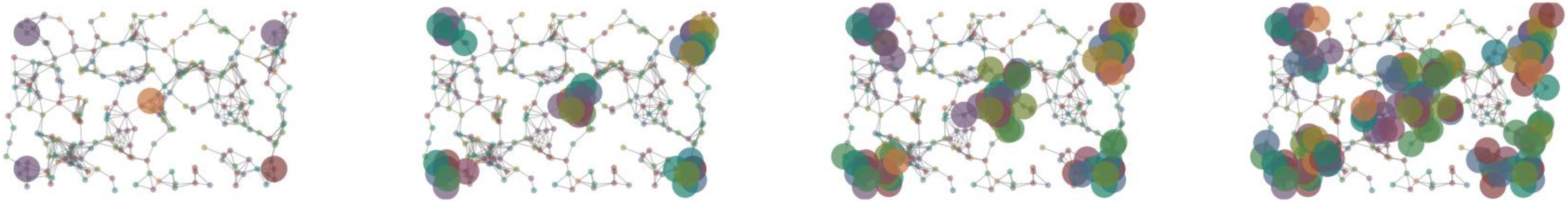
$$\mathbf{x}^{(k+1)} = \mathbf{S}\mathbf{x}^{(k)}, \qquad \text{with} \quad \mathbf{x}^{(0)} = \mathbf{x}$$

- Can unroll the recursion and write the diffusion sequence as the power sequence $\Rightarrow \mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x}$



$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0\mathbf{x}$ $\qquad$ $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}^{(0)} = \mathbf{S}^1\mathbf{x}$ $\qquad$ $\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)} = \mathbf{S}^2\mathbf{x}$ $\qquad$ $\mathbf{x}^{(3)} = \mathbf{S}\mathbf{x}^{(2)} = \mathbf{S}^3\mathbf{x}$

- The kth element of the diffusion sequence $x^{(k)}$ diffuses information to k-hop neighborhoods
  - $\Rightarrow$ One reason why we use the diffusion sequence to define graph convolutions
- We have two definitions. One recursive. The other one using powers of $\mathbf{S}$
  - $\Rightarrow$ Always implement the recursive version. The power version is good for analysis



$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0\mathbf{x}$ $\quad$ $\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}^{(0)} = \mathbf{S}^1\mathbf{x}$ $\quad$ $\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)} = \mathbf{S}^2\mathbf{x}$ $\quad$ $\mathbf{x}^{(3)} = \mathbf{S}\mathbf{x}^{(2)} = \mathbf{S}^3\mathbf{x}$

Graph Convolutional Filters

- Graph convolutional filters are the tool of choice for the linear processing of graph signals

- Given graph shift operator $\mathbf{S}$ and coefficients h_k, a graph filter is a polynomial (series) on $\mathbf{S}$

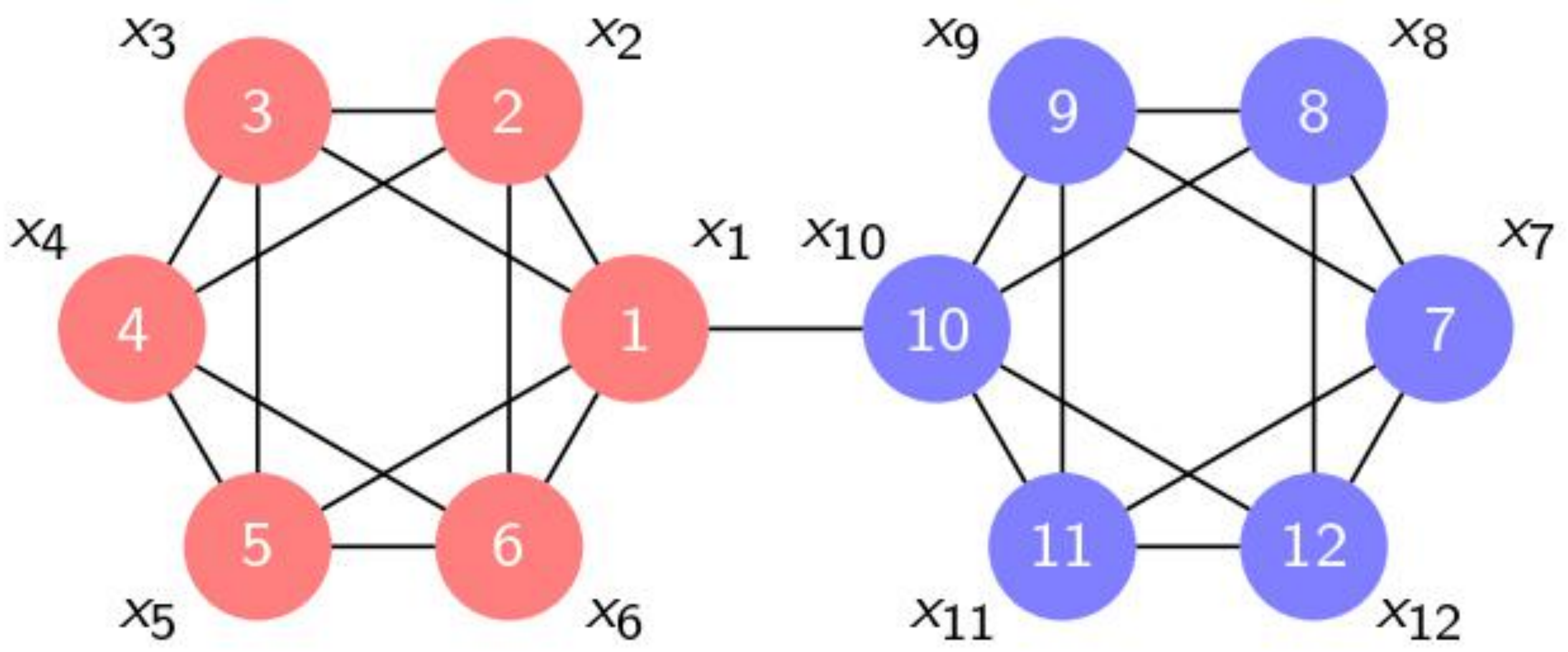
$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$$

- The result of applying the filter $\mathbf{H}(\mathbf{S})$ to the signal $\mathbf{x}$ is the signal

$$\mathbf{y} \; = \; \mathbf{H}(\mathbf{S})\,\mathbf{x} \; = \; \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$$

- We say that $\mathbf{y} = \mathbf{h} \star_{\mathbf{S}} \mathbf{x}$ is the graph convolution of the filter $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ with the signal $\mathbf{x}$

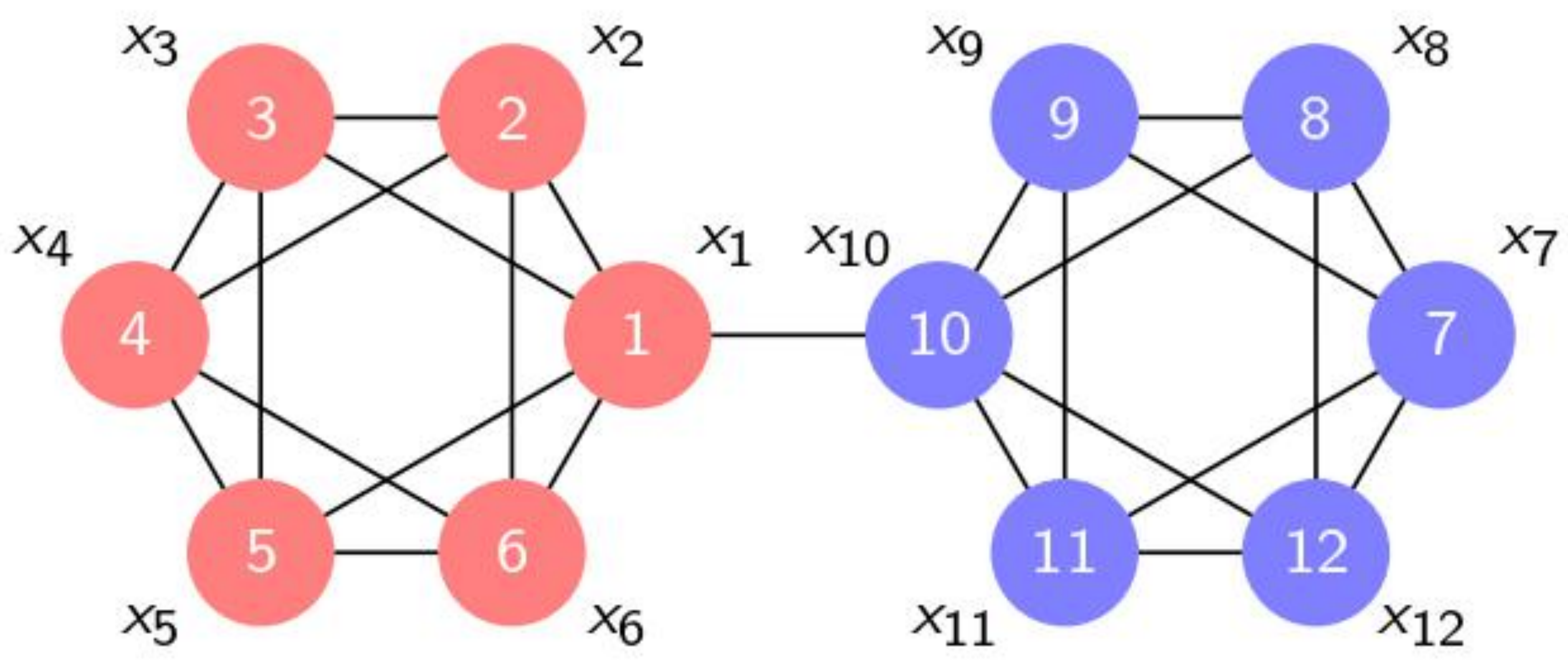
- Graph convolutions aggregate information growing from local to global neighborhoods
- Consider a signal $\mathbf{x}$ supported on a graph with shift operator $\mathbf{S}$. Along with filter $\mathbf{h} = \{h_k\}_{k=0}^{K-1}$



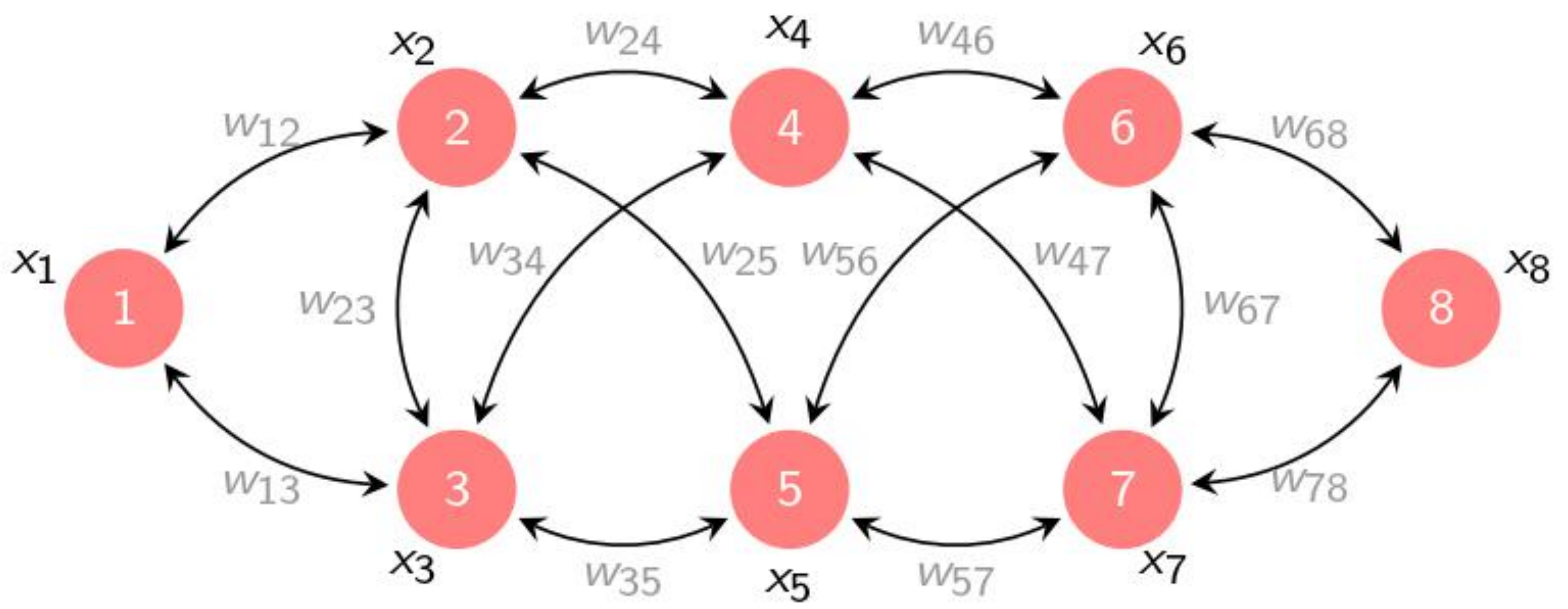
- Graph convolution output $\Rightarrow \mathbf{y} = \mathbf{h} \star_{\mathbf{S}} \mathbf{x} = h_0 \mathbf{S}^0 \mathbf{x} + h_1 \mathbf{S}^1 \mathbf{x} + h_2 \mathbf{S}^2 \mathbf{x} + h_3 \mathbf{S}^3 \mathbf{x} + \ldots = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$

- The same filter $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ can be executed in multiple graphs $\Rightarrow$ We can transfer the filter

Graph Filter on a Graph



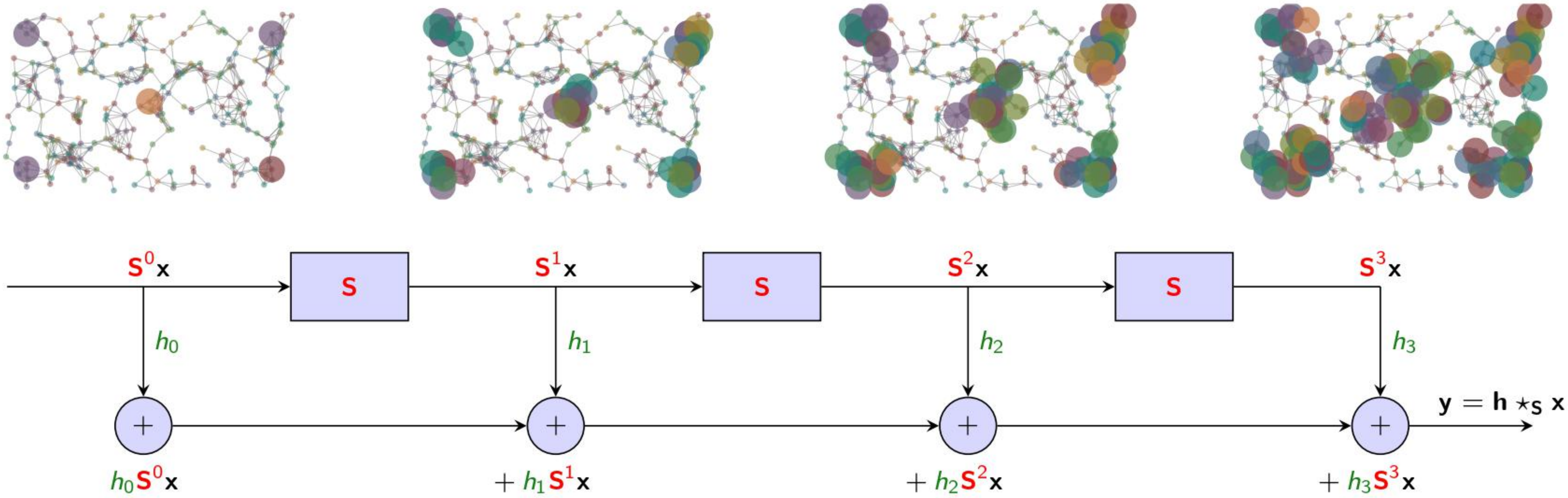
Same Graph Filter on Another Graph



- Graph convolution output $\Rightarrow \mathbf{y} = \mathbf{h} \star_{\mathbf{S}} \mathbf{x} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} + h_3\mathbf{S}^3\mathbf{x} + \ldots = \sum_{k=0}^{\infty} h_k\mathbf{S}^k\mathbf{x}$

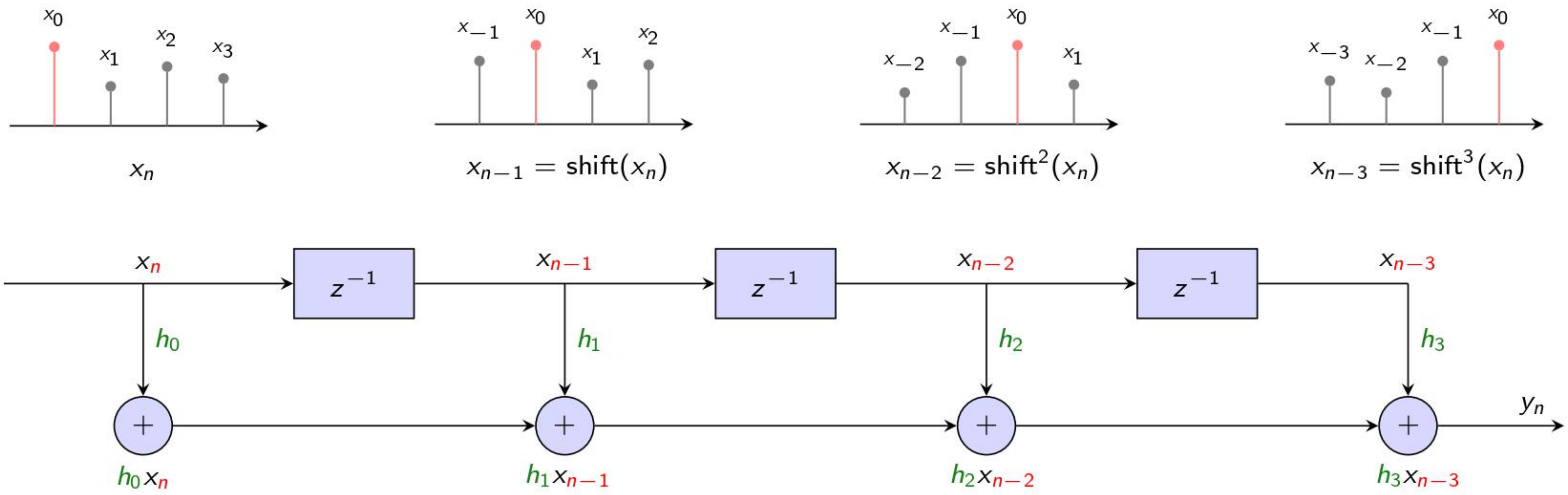
- Output depends on the filter coefficients $\mathbf{h}$, the graph shift operator $\mathbf{S}$ and the signal $\mathbf{x}$

- A graph convolution is a **weighted linear combination** of the elements of the **diffusion sequence**
- Can represent graph convolutions with a **shift register** $\Rightarrow$ Convolution $\equiv$ **Shift. Scale. Sum**



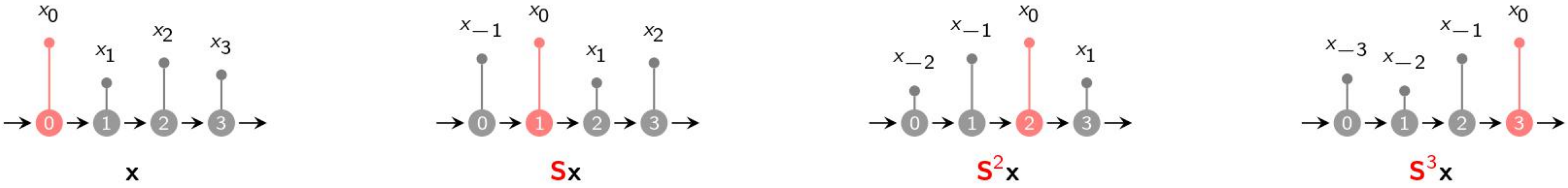
Time Convolutions as a Particular Case of Graph Convolutions

- Convolutional filters process signals in time by leveraging the time shift operator



- The time convolution is a linear combination of time shifted inputs $\Rightarrow y_n = \sum_{k=0}^{K-1} h_k x_{n-k}$

- Time signals are representable as graph signals supported on a line graph $\mathbf{S}$ $\Rightarrow$ The pair $(\mathbf{S}, \mathbf{x})$



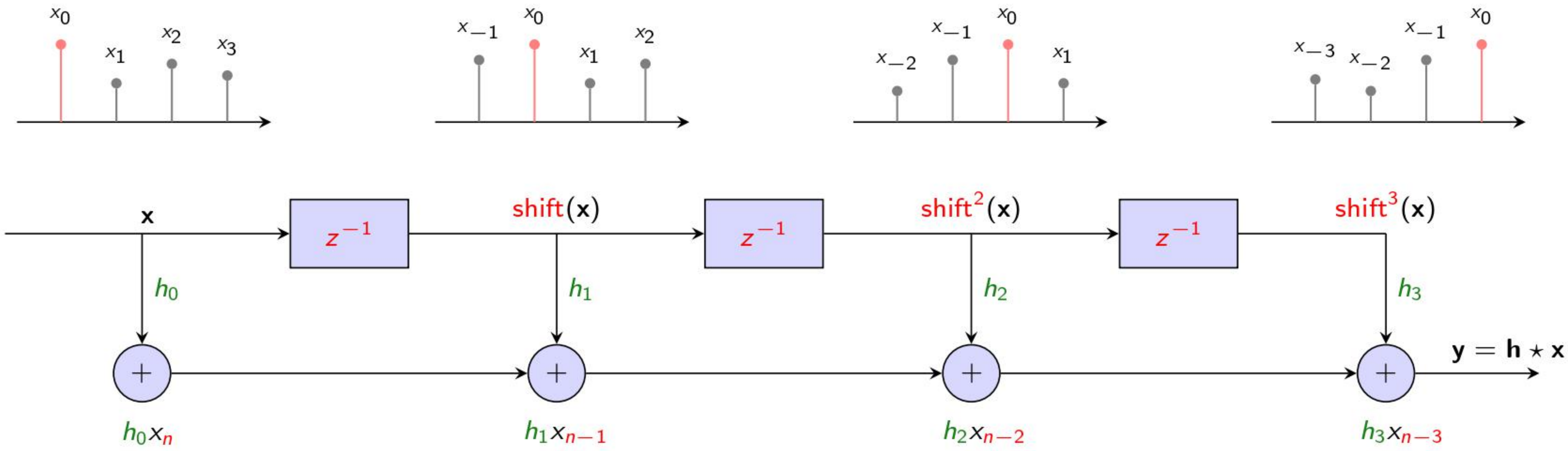
- Time shift is reinterpreted as multiplication by the adjacency matrix $\mathbf{S}$ of the line graph

$$\mathbf{S}^3\mathbf{x} = \mathbf{S}\left[\mathbf{S}^2\mathbf{x}\right] = \mathbf{S}\left[\mathbf{S}\left(\mathbf{S}\mathbf{x}\right)\right] = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ x_{-3} \\ x_{-2} \\ x_{-1} \\ x_0 \\ \vdots \end{bmatrix}$$

- Components of the shift sequence are powers of the adjacency matrix applied to the original signal

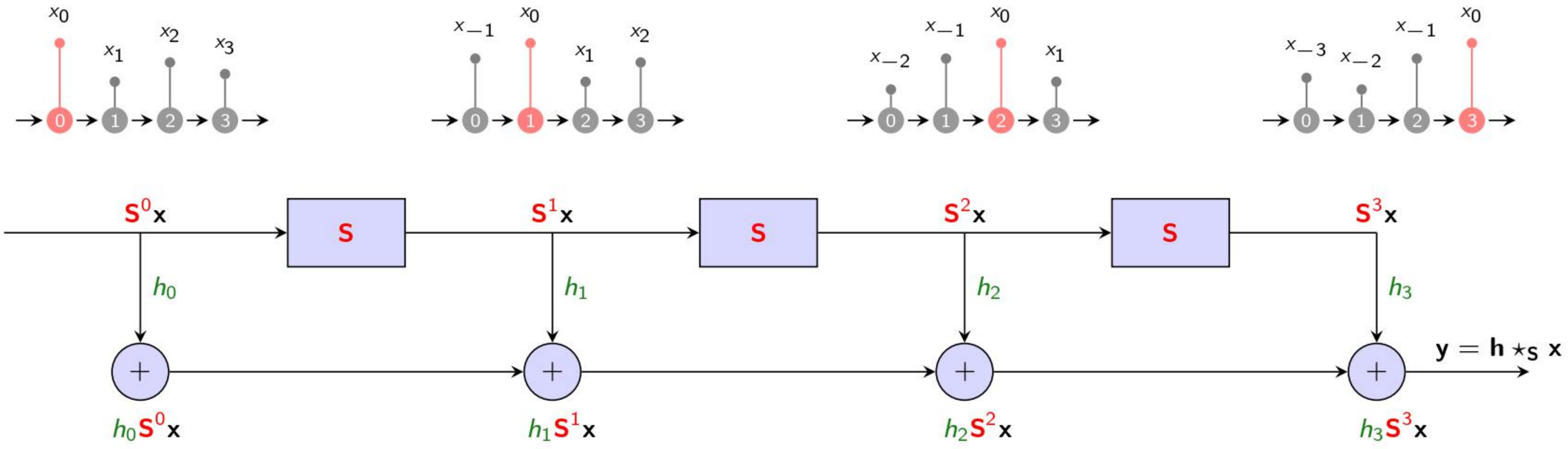
$\Rightarrow$ We can rewrite convolutional filters as polynomials on $\mathbf{S}$, the adjacency of the line graph

- The convolution operation is a linear combination of shifted versions of the input signal
- But we now know that time shifts are multiplications with the adjacency matrix $\mathbf{S}$ of line graph



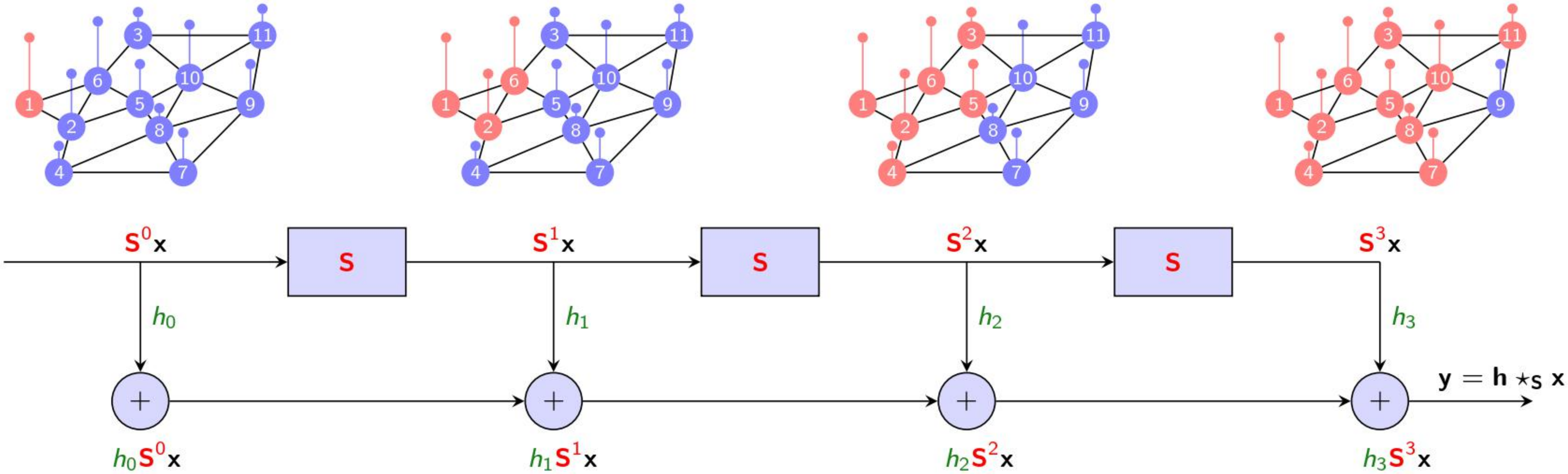
- Time convolution is a polynomial on adjacency matrix of line graph $\Rightarrow \mathbf{y} = \mathbf{h} \star \mathbf{x} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$

- The convolution operation is a linear combination of shifted versions of the input signal
- But we now know that time shifts are multiplications with the adjacency matrix $\mathbf{S}$ of line graph



- Time convolution is a polynomial on adjacency matrix of line graph $\Rightarrow \mathbf{y} = \mathbf{h} \star \mathbf{x} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$

- If we let $\mathbf{S}$ be the shift operator of an arbitrary graph we recover the graph convolution



- The slides are created based on a course named "Graph Neural Networks" in UPenn instructed by Prof. Alejandro Ribeiro