Optimal Power Flow Using Graph Neural Networks

Damian Owerko, Fernando Gama, and Alejandro Ribeiro
Dept. of Electrical and Systems Engineering
University of Pennsylvania

Supported by NSF CCF 1717120, ARO W911NF1710438,
ARL DCIST CRA W911NF-17-2-0181, ISTC-WAS and Intel DevCloud

May 8, 2020
45th Int. Conf. Acoustics, Speech and Signal Processing (ICASSP 2020)

- Electrical grids $\Rightarrow$ of critical importance to our lives

- Electrical grids $\Rightarrow$ of critical importance to our lives

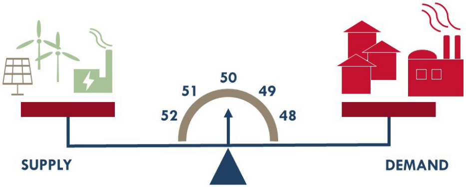
  $\Rightarrow$ We often do not realize how dependent on electricity we are

- Electrical grids $\Rightarrow$ of critical importance to our lives
  - $\Rightarrow$ We often do not realize how dependent on electricity we are

- Electrical grids $\Rightarrow$ of critical importance to our lives
  - $\Rightarrow$ We often do not realize how dependent on electricity we are


- Electrical grids $\Rightarrow$ of critical importance to our lives
  - $\Rightarrow$ We often do not realize how dependent on electricity we are

- Electrical grids $\Rightarrow$ of critical importance to our lives
  - $\Rightarrow$ We often do not realize how dependent on electricity we are



- Requires a constant balance between supply and demand $\Rightarrow$ supply = demand

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow

  $\Rightarrow$ given the demanded power, what is the optimal amount power each generator should produce?

Optimal Power Flow

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow

  $\Rightarrow$ given the demanded power, what is the optimal amount power each generator should produce?

  $\Rightarrow$ very difficult due to sinusoidal nature of electrical power

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow

  $\Rightarrow$ given the demanded power, what is the optimal amount power each generator should produce?

  $\Rightarrow$ very difficult due to sinusoidal nature of electrical power

  $\Rightarrow$ Approximate solutions are either **not-robust**, **costly** or do not **scale**

Optimal Power Flow

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow

  $\Rightarrow$ given the demanded power, what is the optimal amount power each generator should produce?

  $\Rightarrow$ very difficult due to sinusoidal nature of electrical power

  $\Rightarrow$ Approximate solutions are either **not-robust**, **costly** or do not **scale**

Objective

- Use **graph neural networks** to learn the optimal power allocation in a network.

Optimal Power Flow

- The grid is a network $\Rightarrow$ power is generated/demanded at each node by generators/consumers

Two key problems in grid management:

- Power flow $\Rightarrow$ finding the **state** = voltages/currents given net power = generated - demanded at each node
- Optimal power flow
  - $\Rightarrow$ given the demanded power, what is the optimal amount power each generator should produce?
  - $\Rightarrow$ very difficult due to sinusoidal nature of electrical power
  - $\Rightarrow$ Approximate solutions are either **not-robust**, **costly** or do not **scale**

Objective

- Use **graph neural networks** to learn the optimal power allocation in a network.
  - $\Rightarrow$ Local computations, distributed implementations, scalability

Optimal Power Flow

Electrical Grids as Graphs

Graph Neural Networks

Imitating Optimal Power Flow

Conclusions

- Each node in an electrical grid can generate/consume electricity

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

  $v_n \Rightarrow$ The voltage at the node

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$\delta_n \;\Rightarrow$ The voltage angle at the node

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$p_n \Rightarrow$ The net active power flowing into the node

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$q_n \Rightarrow$ The net reactive power flowing into the node

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

  $v_n \Rightarrow$ The voltage at the node

  $\delta_n \Rightarrow$ The voltage angle at the node

  $p_n \Rightarrow$ The net active power flowing into the node

  $q_n \Rightarrow$ The net reactive power flowing into the node

$$\mathbf{x}_n = \begin{bmatrix} v_n & \delta_n & p_n & q_n \end{bmatrix}$$

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$$\mathbf{x}_n = \begin{bmatrix} v_n & \delta_n & p_n & q_n \end{bmatrix}$$

- The state of the whole grid is

$$\mathbf{X} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\delta} & \mathbf{p} & \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T & \dots & \mathbf{x}_n^T & \dots & \mathbf{x}_N^T \end{bmatrix}^T$$

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$$\mathbf{x}_n = \begin{bmatrix} v_n & \delta_n & p_n & q_n \end{bmatrix}$$

- The state of the whole grid is

$$\mathbf{X} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\delta} & \mathbf{p} & \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T & \dots & \mathbf{x}_n^T & \dots & \mathbf{x}_N^T \end{bmatrix}^T$$

- The physical characteristics of the grid are described by the power flow equations

$$\mathbf{p} = \mathcal{P}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

$$\mathbf{q} = \mathcal{Q}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$$\mathbf{x}_n = \begin{bmatrix} v_n & \delta_n & p_n & q_n \end{bmatrix}$$

- The state of the whole grid is

$$\mathbf{X} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\delta} & \mathbf{p} & \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T & \dots & \mathbf{x}_n^T & \dots & \mathbf{x}_N^T \end{bmatrix}^T$$

- The physical characteristics of the grid are described by the power flow equations

$$\mathbf{p} = \mathcal{P}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

$$\mathbf{q} = \mathcal{Q}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

$\Rightarrow$ Relate local net power generation with the global state

- Each node in an electrical grid can generate/consume electricity
- The state of the nth node is expressed by 4 real scalars:

$$\mathbf{x}_n = \begin{bmatrix} v_n & \delta_n & p_n & q_n \end{bmatrix}$$

- The state of the whole grid is

$$\mathbf{X} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\delta} & \mathbf{p} & \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T & \dots & \mathbf{x}_n^T & \dots & \mathbf{x}_N^T \end{bmatrix}^T$$

- The physical characteristics of the grid are described by the power flow equations

$$\mathbf{p} = \mathcal{P}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

$$\mathbf{q} = \mathcal{Q}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W})$$

$\Rightarrow$ Relate local net power generation with the global state

$\Rightarrow$ Depends on the topology $\mathbf{W}$ of the grid $\mathbf{W}$ is the set of electrical components in the grid

- Thus optimal power flow is a minimization problem

$$\underset{\{p_n\}}{\text{minimize}} \sum_{n=1}^{N} c_n(p_n, q_n) \tag{1}$$

(5)

- Where,

  $\Rightarrow$ $c_n(p_n, q_n)$ is the cost to generate power at the nth node

- Thus optimal power flow is a minimization problem

$$\underset{\{p_n\}}{\text{minimize}} \sum_{n=1}^{N} c_n(p_n, q_n) \tag{1}$$

$$\text{subject to } \mathbf{p} = \mathcal{P}(\mathbf{v}, \delta; \mathbf{W}), \tag{2}$$

$$\mathbf{q} = \mathcal{Q}(\mathbf{v}, \delta; \mathbf{W}), \tag{3}$$

(5)

- Where,
  - $\Rightarrow$ $c_n(p_n, q_n)$ is the cost to generate power at the nth node
  - $\Rightarrow$ (2) and (3) are the powerflow equations $\Rightarrow$ sinusoidal constraints

- Thus optimal power flow is a minimization problem

$$\begin{aligned} \underset{\{p_n\}}{\text{minimize}} \sum_{n=1}^{N} c_n(p_n, q_n) && (1)\\ \text{subject to } \mathbf{p} = \mathcal{P}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W}), && (2)\\ \mathbf{q} = \mathcal{Q}(\mathbf{v}, \boldsymbol{\delta}; \mathbf{W}), && (3)\\ \mathbf{X}^{\text{min}} \preceq \mathbf{X} \preceq \mathbf{X}^{\text{max}}. && (4)\\ && (5) \end{aligned}$$

- Where,
  - $\Rightarrow$ $c_n(p_n, q_n)$ is the cost to generate power at the nth node
  - $\Rightarrow$ (2) and (3) are the powerflow equations $\Rightarrow$ sinusoidal constraints
  - $\Rightarrow$ $\mathbf{X}^{min}$ and $\mathbf{X}^{max}$ collect the minimum and maximum values each state entry can take
  - $\Rightarrow$ $\mathbf{X} \preceq \mathbf{Y} \iff [\mathbf{X}]_{ij} \leq [\mathbf{Y}]_{ij}$

DCOPF

- Most commonly used in practice
- Uses small angle approximations to linearize

$$\boldsymbol{\delta} = \mathbf{0}$$

- Assumption invalid for moderately/heavily loaded grids

DCOPF

- Most commonly used in practice
- Uses small angle approximations to linearize

$$\boldsymbol{\delta} = \mathbf{0}$$

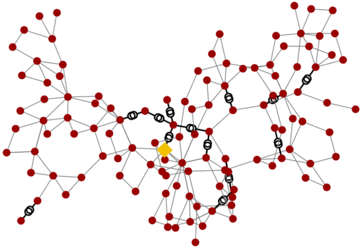
- Assumption invalid for moderately/heavily loaded grids

ACOPF

- Provides **exact** solution to OPF
- Solved using interior point methods (IPOPT)
- Very slow for large networks

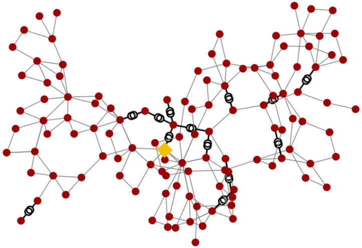
  $\Rightarrow$ Impractical for real-time optimization

- Electrical grid $\Rightarrow$ weighted graph



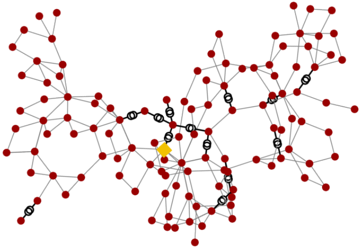
IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power



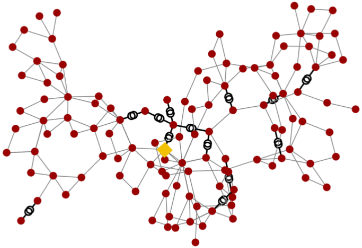
IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power
- Edges represent electrical connections between nodes



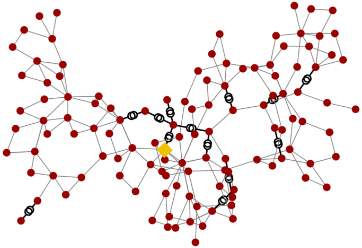
IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power
- Edges represent electrical connections between nodes
- State matrix $\mathbf{X} \in \mathbb{R}^{N \times 4}$ = graph signal with 4 features
  - $\Rightarrow$ Each row is the state of the corresponding node



IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power
- Edges represent electrical connections between nodes
- State matrix $\mathbf{X} \in \mathbb{R}^{N \times 4}$ = graph signal with 4 features
  - $\Rightarrow$ Each row is the state of the corresponding node
- Adjacency matrix $\mathbf{A}$ = Gaussian kernel of impedance

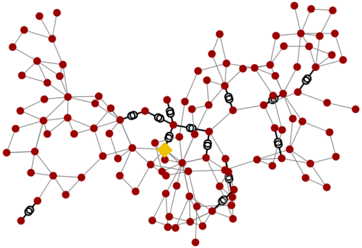


IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power
- Edges represent electrical connections between nodes
- State matrix $\mathbf{X} \in \mathbb{R}^{N \times 4}$ = graph signal with 4 features
  - $\Rightarrow$ Each row is the state of the corresponding node
- Adjacency matrix $\mathbf{A}$ = Gaussian kernel of impedance
  - $\Rightarrow$ How close the two nodes are two each other

$$\mathbf{A}_{ij} = exp(-k|z_{ij}|^2)$$

- Where z_{ij} is the line impedance



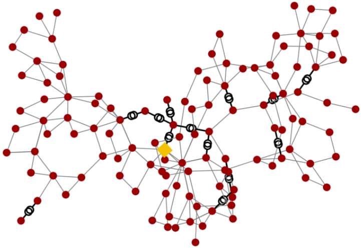
IEEE118 Power System Network

- Electrical grid $\Rightarrow$ weighted graph
- Nodes produce/consume power
- Edges represent electrical connections between nodes
- State matrix $\mathbf{X} \in \mathbb{R}^{N \times 4}$ = graph signal with 4 features
  - $\Rightarrow$ Each row is the state of the corresponding node
- Adjacency matrix $\mathbf{A}$ = Gaussian kernel of impedance
  - $\Rightarrow$ How close the two nodes are two each other

$$w_{ij} = exp(-k|z_{ij}|^2)$$

$$\mathbf{A}_{ij} = \begin{cases} w_{ij}, & \text{if } w_{ij} > w \\ 0, & \text{otherwise} \end{cases}$$

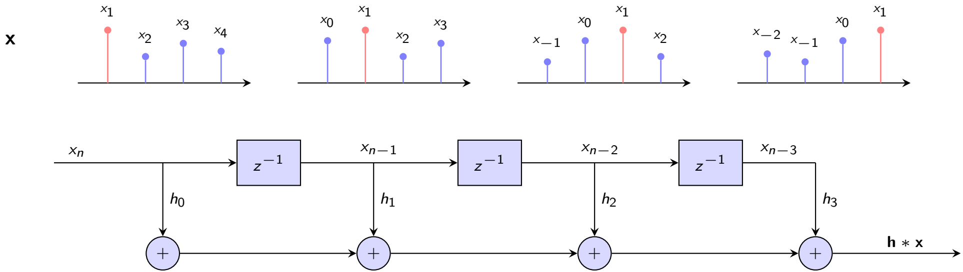
- Where z_{ij} is the line impedance



IEEE118 Power System Network

- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} * \mathbf{h} = \sum_{k=0}^{K-1} h_k x_{n-k}$$



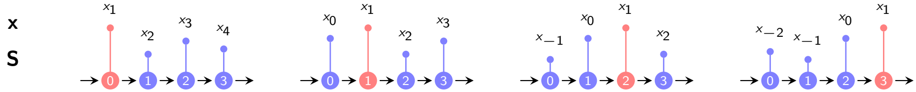
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

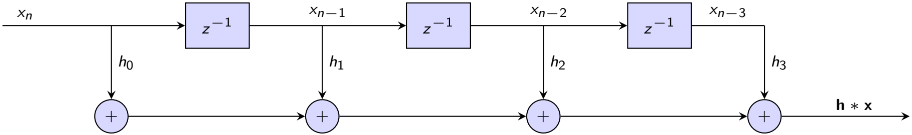
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} * \mathbf{h} = \sum_{k=0}^{K-1} h_k \, x_{n-k}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$





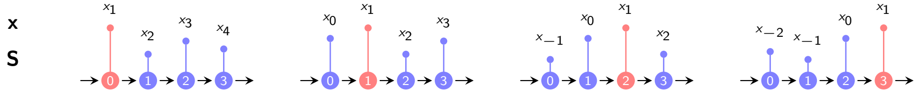
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

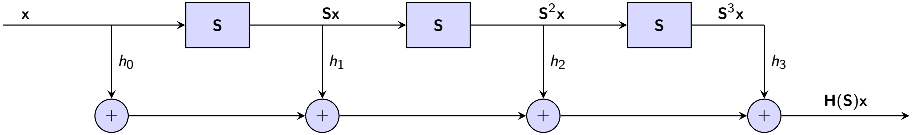
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & 0 & 0 & 0 & \cdots \\ \cdots & 1 & 0 & 0 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ x_1 \\ x_2 \\ x_3 \\ \vdots \end{bmatrix}$$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

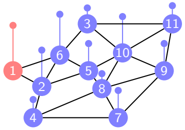
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

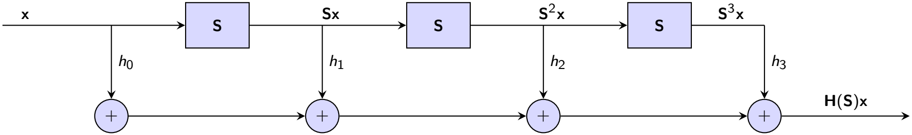
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

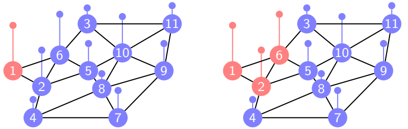
- Graph **convolution** $\Rightarrow$ **Linear combination** of shifted versions of the signal $\mathbf{x}$

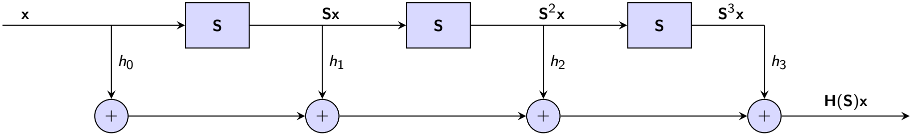
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ **shifts** the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0\\ 1&0&0&1&1&1&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&1&1\\ 0&1&0&0&0&0&1&1&0&0&0\\ 0&1&0&0&1&0&0&1&0&1&0\\ 1&1&1&0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&1&1&0&0\\ 0&0&0&1&1&0&1&0&1&1&0\\ 0&0&0&0&0&0&1&1&0&1&1\\ 0&0&0&0&1&0&0&1&1&0&1\\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

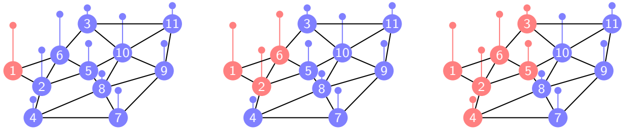
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

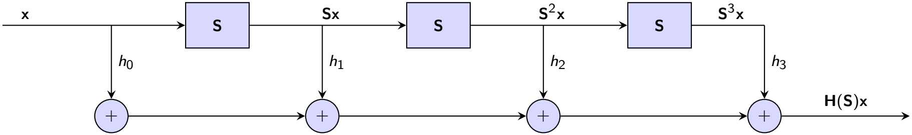
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

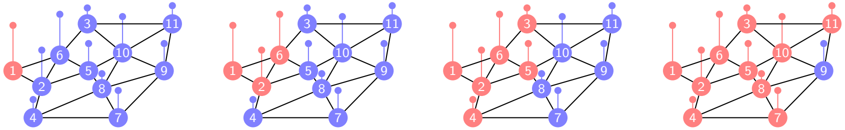
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal $\mathbf{x}$

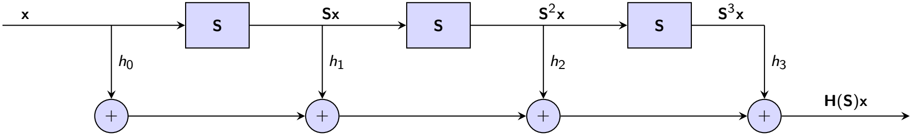
$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x}$$

$$\begin{bmatrix} 0&1&0&0&0&1&0&0&0&0&0 \\ 1&0&0&1&1&1&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&1&1 \\ 0&1&0&0&0&0&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&0&1&0 \\ 1&1&1&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&1&0&1&1&0 \\ 0&0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&0&1&0&0&1&1&0&1 \\ 0&0&1&0&0&0&0&0&1&1&0 \end{bmatrix}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph $\Rightarrow$ $\mathbf{Sx}$ shifts the signal $\mathbf{x}$

$(\mathbf{x}; \mathbf{S})$





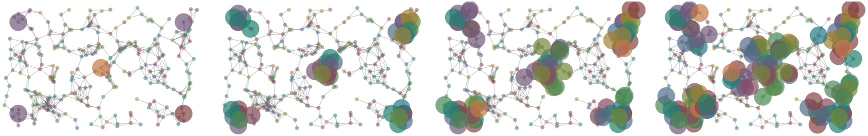
Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

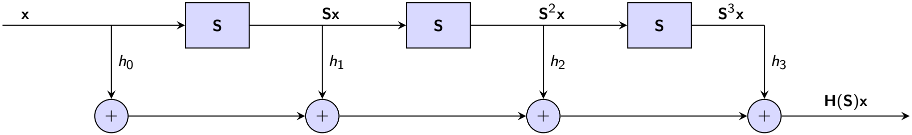
- Graph convolution $\Rightarrow$ Linear combination of shifted versions of the signal

$$\mathbf{x} *_{\mathbf{S}} \mathbf{h} = \sum_{k=0}^{K-1} h_k \, \mathbf{S}^k \mathbf{x} = \mathbf{H}(\mathbf{S})\mathbf{x}$$

- Notion of shift $\mathbf{S}$ $\Rightarrow$ Matrix description of graph (adjacency, Laplacian)
- Linear combination of neighboring signal $\Rightarrow$ Local operation

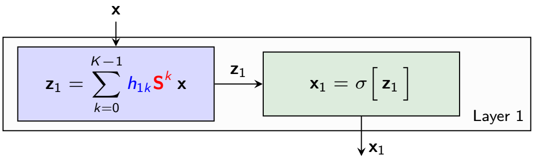
$(\mathbf{x}; \mathbf{S})$





Gama, Marques, Leus, Ribeiro, "Convolutional Graph Neural Networks", Asilomar, 2019.

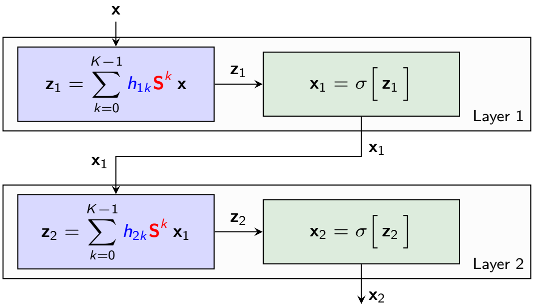
- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

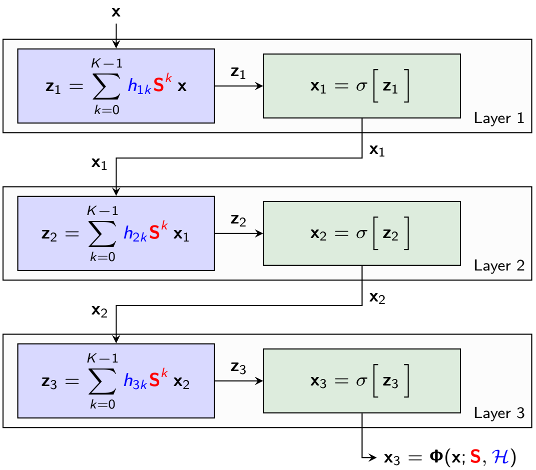
Graph Neural Networks

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

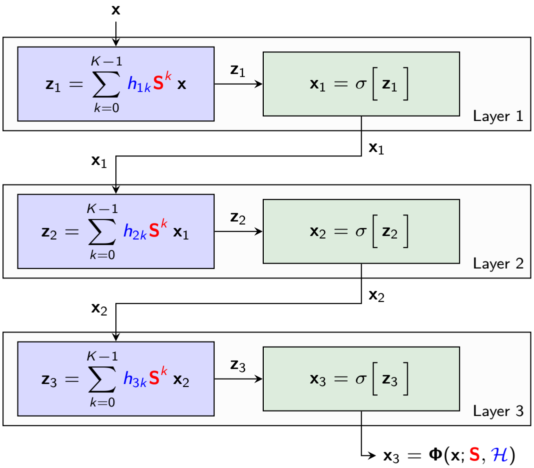
- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)
- The GNN $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ depends on the filters $\mathcal{H}$
  - $\Rightarrow$ Learn filter taps $\mathcal{H}$ from training data
  - $\Rightarrow$ Also depends on the graph $\mathbf{S}$



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

Graph Neural Networks

- Cascade of L layers
  - $\Rightarrow$ Graph convolutions with filters $\mathcal{H} = \{\mathbf{h}_\ell\}$
  - $\Rightarrow$ Pointwise nonlinearity (activation functions)
- The GNN $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$ depends on the filters $\mathcal{H}$
  - $\Rightarrow$ Learn filter taps $\mathcal{H}$ from training data
  - $\Rightarrow$ Also depends on the graph $\mathbf{S}$
- Nonlinear mapping $\mathbf{\Phi}(\mathbf{x}; \mathbf{S}, \mathcal{H})$
  - $\Rightarrow$ Exploit underlying graph structure $\mathbf{S}$
  - $\Rightarrow$ Local information
  - $\Rightarrow$ Distributed implementation



Gama, Marques, Leus, Ribeiro, "Convolutional Neural Network Architectures for Signals Supported on Graphs", IEEE TSP, 2019

- We use the GNN as a model $\mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})$

- We use the GNN as a model $\mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})$
- Want to imitate the OPF solution $\mathbf{p}^*$

- We use the GNN as a model $\mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})$
- Want to imitate the OPF solution $\mathbf{p}^*$

  $\Rightarrow$ Want to minimize a loss $\mathcal{L}$ over a dataset $\mathcal{T} = \{(\mathbf{X}, \mathbf{p}^*)\}$

$$\min_{\mathcal{H}} \sum_{\mathcal{T}} \mathcal{L}\Big(\mathbf{p}^*, \mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})\Big)$$

- We use the GNN as a model $\mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})$
- Want to imitate the OPF solution $\mathbf{p}^*$

$\Rightarrow$ Want to minimize a loss $\mathcal{L}$ over a dataset $\mathcal{T} = \{(\mathbf{X}, \mathbf{p}^*)\}$

$$\min_{\mathcal{H}} \sum_{\mathcal{T}} \mathcal{L}\Big(\mathbf{p}^*, \mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})\Big)$$

$\Rightarrow$ We use $\mathcal{L} = \text{MSE}$

- We use the GNN as a model $\mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})$
- Want to imitate the OPF solution $\mathbf{p}^*$

  $\Rightarrow$ Want to minimize a loss $\mathcal{L}$ over a dataset $\mathcal{T} = \{(\mathbf{X}, \mathbf{p}^*)\}$

$$\min_{\mathcal{H}} \sum_{\mathcal{T}} \mathcal{L}\Big(\mathbf{p}^*, \mathbf{\Phi}(\mathbf{X}; \mathbf{A}, \mathcal{H})\Big)$$

  $\Rightarrow$ We use $\mathcal{L} = \text{MSE}$
- Once $\mathbf{\Phi}$ is trained we do not need the costly $\mathbf{p}^*$ to make predictions

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118
  - $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118
  - $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$
  - $\Rightarrow$ We want to test our model under varying conditions $\Rightarrow$ sample a uniform distribution around the reference loads

$$\mathbf{p}^L \sim \text{Uniform}(0.9\ \mathbf{p}^L_{\text{ref}}, 1.1\ \mathbf{p}^L_{\text{ref}})$$
$$\mathbf{q}^L \sim \text{Uniform}(0.9\ \mathbf{q}^L_{\text{ref}}, 1.1\ \mathbf{q}^L_{\text{ref}})$$

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118
  - $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$
  - $\Rightarrow$ We want to test our model under varying conditions $\Rightarrow$ sample a uniform distribution around the reference loads

$$\mathbf{p}^L \sim \text{Uniform}(0.9\ \mathbf{p}^L_{\text{ref}}, 1.1\ \mathbf{p}^L_{\text{ref}})$$
$$\mathbf{q}^L \sim \text{Uniform}(0.9\ \mathbf{q}^L_{\text{ref}}, 1.1\ \mathbf{q}^L_{\text{ref}})$$

- For each load sample $\mathbf{p}^L, \mathbf{q}^L$

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118

  $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$

  $\Rightarrow$ We want to test our model under varying conditions $\Rightarrow$ sample a uniform distribution around the reference loads

$$\mathbf{p}^L \sim \text{Uniform}(0.9\ \mathbf{p}^L_{\text{ref}}, 1.1\ \mathbf{p}^L_{\text{ref}})$$
$$\mathbf{q}^L \sim \text{Uniform}(0.9\ \mathbf{q}^L_{\text{ref}}, 1.1\ \mathbf{q}^L_{\text{ref}})$$

- For each load sample $\mathbf{p}^L, \mathbf{q}^L$

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118

  $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$

  $\Rightarrow$ We want to test our model under varying conditions $\Rightarrow$ sample a uniform distribution around the reference loads

$$\mathbf{p}^L \sim \text{Uniform}(0.9\ \mathbf{p}^L_{\text{ref}}, 1.1\ \mathbf{p}^L_{\text{ref}})$$

$$\mathbf{q}^L \sim \text{Uniform}(0.9\ \mathbf{q}^L_{\text{ref}}, 1.1\ \mathbf{q}^L_{\text{ref}})$$

- For each load sample $\mathbf{p}^L, \mathbf{q}^L$

  $\mathbf{X} =$ the **sub-optimal** state $\Rightarrow$ the DC-OPF solution to the case

- We construct $\mathcal{T}$ using the IEEE power system test cases – IEEE30 and IEEE118

  $\Rightarrow$ Each test case describes the grid structure, constraints, and provides reference loads $\mathbf{P}^L_{ref}, \mathbf{Q}^L_{ref}$

  $\Rightarrow$ We want to test our model under varying conditions $\Rightarrow$ sample a uniform distribution around the reference loads

$$\mathbf{p}^L \sim \text{Uniform}(0.9\ \mathbf{p}^L_{\text{ref}}, 1.1\ \mathbf{p}^L_{\text{ref}})$$
$$\mathbf{q}^L \sim \text{Uniform}(0.9\ \mathbf{q}^L_{\text{ref}}, 1.1\ \mathbf{q}^L_{\text{ref}})$$

- For each load sample $\mathbf{p}^L, \mathbf{q}^L$

  $\mathbf{X}$ = the **sub-optimal** state $\Rightarrow$ the DC-OPF solution to the case

  $\mathbf{p}^*$ = the optimal AC-OPF solution obtained using **IPOPT** $\Rightarrow$ costly, **only** needed during training

Imitating Optimal Power Flow

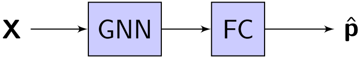
- We compare GNNs against Multi-Layer Perceptrons

Global vs Local MLP

Note the difference between the Global and Local MLP. In the latter architecture there are N independent neural networks, one for each node.

- We compare GNNs against Multi-Layer Perceptrons

Global GNN



Local GNN

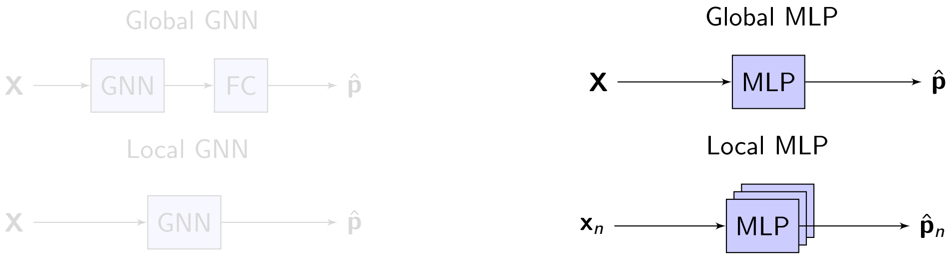
Global MLP

Local MLP

Global vs Local MLP

Note the difference between the Global and Local MLP. In the latter architecture there are N independent neural networks, one for each node.

- We compare GNNs against Multi-Layer Perceptrons



Global vs Local MLP

Note the difference between the Global and Local MLP. In the latter architecture there are N independent neural networks, one for each node.

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - $\Rightarrow$ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  ⇒ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories ⇒ This is more prominent on the IEEE118 dataset

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - $\Rightarrow$ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories $\Rightarrow$ This is more prominent on the IEEE118 dataset
  - $\Rightarrow$ 213% improvement of Global GNN over Global MLP

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - ⇒ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories ⇒ This is more prominent on the IEEE118 dataset
  - ⇒ 213% improvement of Global GNN over Global MLP
  - ⇒ 1082% improvement of Local GNN over Local MLP

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - ⇒ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories ⇒ This is more prominent on the IEEE118 dataset
  - ⇒ 213% improvement of Global GNN over Global MLP
  - ⇒ 1082% improvement of Local GNN over Local MLP
- GNNs are also much faster than traditional methods

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - $\Rightarrow$ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories $\Rightarrow$ This is more prominent on the IEEE118 dataset
  - $\Rightarrow$ 213% improvement of Global GNN over Global MLP
  - $\Rightarrow$ 1082% improvement of Local GNN over Local MLP
- GNNs are also much faster than traditional methods
  - $\Rightarrow$ Finding p^* using IPOPT takes 2s for IEEE30 18s for IEEE118

- We train the models under MSE loss on data synthesized from IEEE30 and IEEE118 test cases
  - $\Rightarrow$ We compare the RMSE for all four models

IEEE30	GNN	MLP
Global	0.061	0.090
Local	0.139	0.161

IEEE118	GNN	MLP
Global	0.00306	0.00958
Local	0.03038	0.35932

- GNNs outperform MLPs in all experimental categories $\Rightarrow$ This is more prominent on the IEEE118 dataset
  - $\Rightarrow$ 213% improvement of Global GNN over Global MLP
  - $\Rightarrow$ 1082% improvement of Local GNN over Local MLP
- GNNs are also much faster than traditional methods
  - $\Rightarrow$ Finding p^* using IPOPT takes 2s for IEEE30 18s for IEEE118
  - $\Rightarrow$ The GNN takes $\approx 50\mu s$ to make predictions $\Rightarrow$ GNNs are 10^5 times faster

Conclusions

- Solving OPF is central to electrical grid operation
- OPF $\Rightarrow$ How to satisfy demand while **minimizing** operational costs?

  $\Rightarrow$ Non-linear constrains $\Rightarrow$ computationally expensive (NP hard)
- GNNs are well suited to applications on the electrical grid

  $\Rightarrow$ Scalable $\Rightarrow$ number of taps independent on network size

  $\Rightarrow$ Exploit the network structure of the data
- GNNs are up to 10^5 times faster than IPOPT

Thank You!