Node-Asynchronous Spectral Clustering on Directed Graphs

Oguzhan Teke P. P. Vaidyanathan

Department of Electrical Engineering
California Institute of Technology

45th International Conference on
Acoustics, Speech and Signal Processing
(ICASSP 2020)

Caltech

Outline

1. Graph Signal Processing and Spectral Clustering
2. Asynchronous Updates on Graphs
3. Autonomous Spectral Clustering
4. Conclusion

Outline

1 Graph Signal Processing and Spectral Clustering

2 Asynchronous Updates on Graphs

3 Autonomous Spectral Clustering

4 Conclusion

Outline

1. Graph Signal Processing and Spectral Clustering
2. Asynchronous Updates on Graphs
3. Autonomous Spectral Clustering
4. Conclusion

Outline

1. Graph Signal Processing and Spectral Clustering
2. Asynchronous Updates on Graphs
3. Autonomous Spectral Clustering
4. Conclusion

Outline

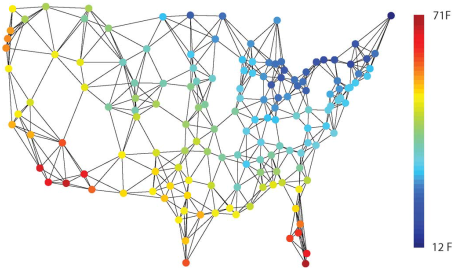
1 Graph Signal Processing and Spectral Clustering

2 Asynchronous Updates on Graphs

3 Autonomous Spectral Clustering

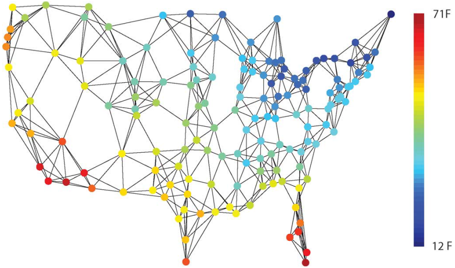
4 Conclusion

Preliminaries



$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{C}^N$$

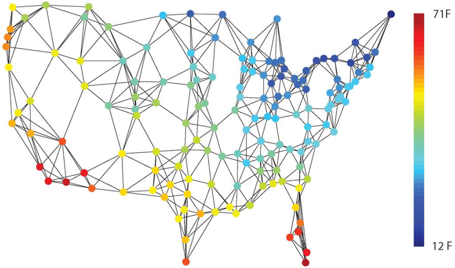
Preliminaries



$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{C}^N$$

$\mathbf{A}$ is the graph operator

Preliminaries



$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_N \end{bmatrix} \in \mathbb{C}^N$$

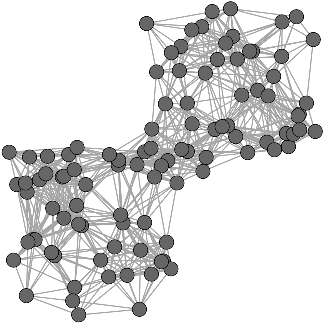
$\mathbf{A}$ is the graph operator

Adjacency matrix[1] : $\mathbf{A}$
Graph Laplacians[2] : $\mathbf{L}$, or $\mathcal{L}$
Other selections[3]

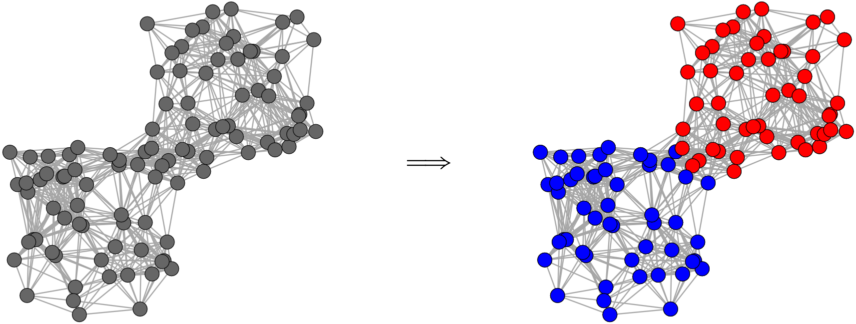
[1] Sandryhaila & Moura, "Discrete Signal Processing on Graphs," *IEEE Trans. S. P. vol. 61, no. 7, 2013*

[2] Shuman et al, "The emerging field of signal processing on graphs: ...," *IEEE S. P. Magazine, vol. 30, no. 3 2013*

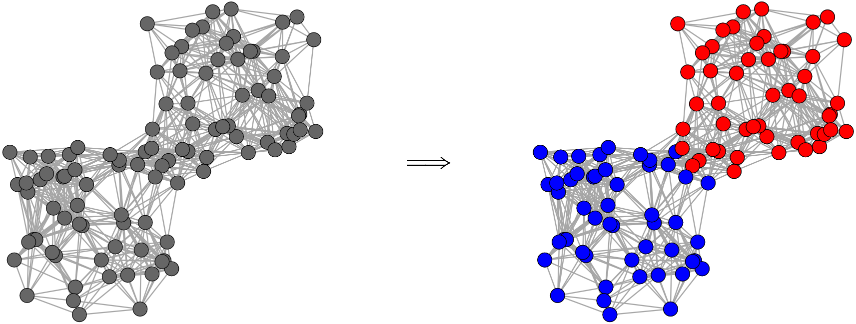
Spectral Clustering



Spectral Clustering

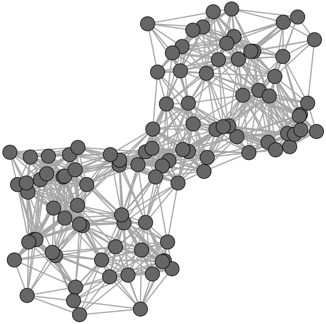


Spectral Clustering

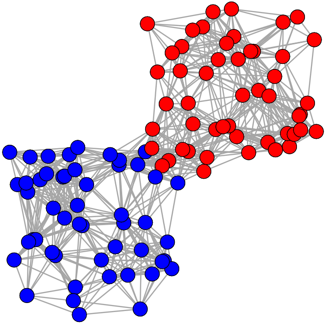


Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

Spectral Clustering



$\Longrightarrow$



Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

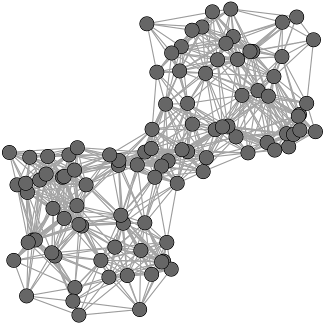
$0 = \lambda_1 < \lambda_2 \leqslant \cdots \leqslant \lambda_N$
(Fiedler Value)

$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$
(Fiedler Vector)

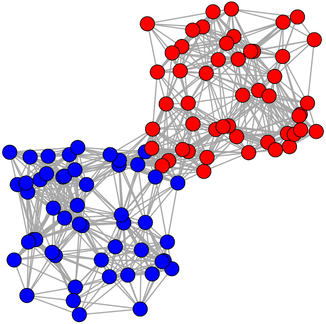
[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Ng, Jordan, and Weiss, "On spectral clustering: Analysis and an algorithm," *NIPS, vol. 67, no. 11, 2002*

Spectral Clustering



$\Longrightarrow$



Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

$$0 = \lambda_1 < \lambda_2 \leqslant \cdots \leqslant \lambda_N$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

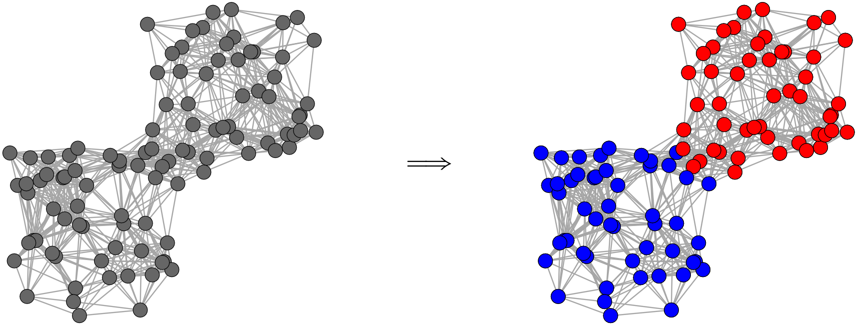
Compute $\mathbf{v}_2$ with random asynchronous computations?

[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Ng, Jordan, and Weiss, "On spectral clustering: Analysis and an algorithm," *NIPS, vol. 67, no. 11, 2002*

[3] Teke & Vaidyanathan, "Random Node-Asynchronous Updates on Graphs," *IEEE Trans. S.P., vol. 67, no. 11, 2019*

Spectral Clustering *Here: Directed Case*



Graph Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$

$$0 = \lambda_1 < \lambda_2 \leqslant \cdots \leqslant \lambda_N$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

Compute $\mathbf{v}_2$ with random asynchronous computations?

[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Ng, Jordan, and Weiss, "On spectral clustering: Analysis and an algorithm," *NIPS, vol. 67, no. 11, 2002*

[3] Teke & Vaidyanathan, "Random Node-Asynchronous Updates on Graphs," *IEEE Trans. S.P., vol. 67, no. 11, 2019*

Outline

1 Graph Signal Processing and Spectral Clustering

2 Asynchronous Updates on Graphs

3 Autonomous Spectral Clustering

4 Conclusion

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i$$

Asynchronous Fixed-Point Iterations

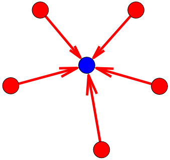
$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i = \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i$$



Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

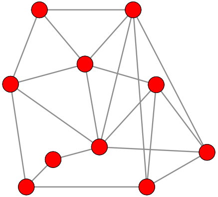
$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



Asynchronous Fixed-Point Iterations

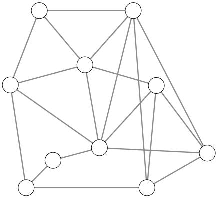
$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$\begin{aligned} x_k[i] &= \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i & \forall\ i \\ &= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i & \forall\ i \end{aligned}$$



Asynchronous Fixed-Point Iterations

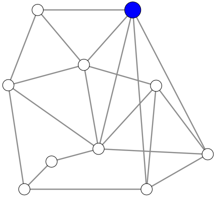
$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$\begin{aligned} x_k[i] &= \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i \\ &= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i \end{aligned}$$



Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

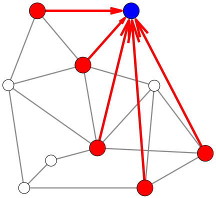
$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

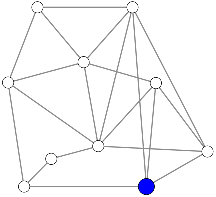
$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



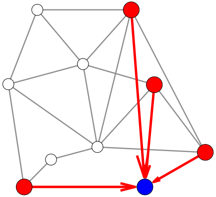
Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator $\qquad$ $\mathbf{x}_k$ = Signal on the Graph

$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$ $\qquad$ $\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

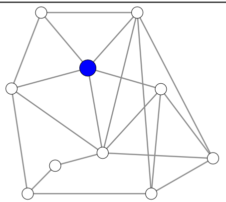
$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

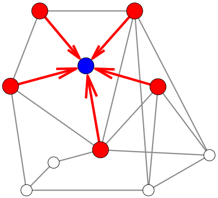
$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



Asynchronous Fixed-Point Iterations

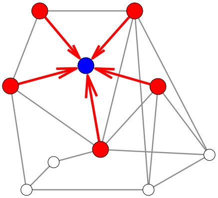
$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$
$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

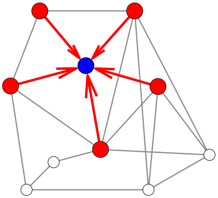
$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$

$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$



Recurrent NN
(Hopfield Model)[1]

$$x_k[i] = \theta\left(\mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i\right)$$

[1] Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS, 1982*

Asynchronous Fixed-Point Iterations

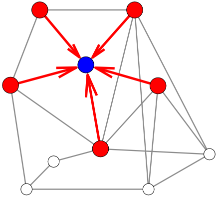
$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$
$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$



$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\boxed{\lim_{k \to \infty} \mathbf{x}_k = ?}$$

Recurrent NN (Hopfield Model)[1]

$$x_k[i] = \theta\left(\mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i\right)$$

[1] Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS, 1982*

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

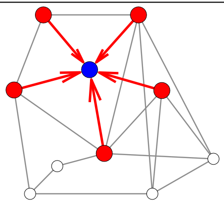
$\mathbf{u}$ = Input Signal ($\mathbf{u}=\mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$

$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k\to\infty} \mathbf{x}_k = ?$$



Recurrent NN
(Hopfield Model)

$$x_k[i] = \theta\left(\mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i\right)$$

Synchronous case: $\rho(\mathbf{A}) < 1 \implies \lim_{k\to\infty} \mathbf{x}_k = (\mathbf{I} - \mathbf{A})^{\text{-}1}\,\mathbf{u}$

[1] Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS*, *1982*

Asynchronous Fixed-Point Iterations

$\mathbf{A}$ = Graph Operator

$\mathbf{x}_k$ = Signal on the Graph

$$\mathbf{x}_k = \mathbf{A}\ \mathbf{x}_{k\text{-}1} + \mathbf{u}$$

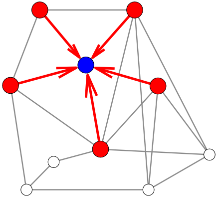
$\mathbf{u}$ = Input Signal ($\mathbf{u} = \mathbf{0}$ in clustering)

$$x_k[i] = \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i \qquad \forall\ i$$

$$= \sum_{j \in \mathcal{N}(i)} a_{i,j}\ x_{k\text{-}1}[j] + u_i \qquad \forall\ i$$

$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$



Recurrent NN (Hopfield Model)[1]

$$x_k[i] = \theta\left(\mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i\right)$$

Synchronous case: $\rho(\mathbf{A}) < 1 \implies \lim_{k \to \infty} \mathbf{x}_k = (\mathbf{I} - \mathbf{A})^{\text{-}1}\, \mathbf{u}$

Random Asynchronous case: To be discussed next ...

[1] Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *PNAS*, *1982*

Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases} \qquad \lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$

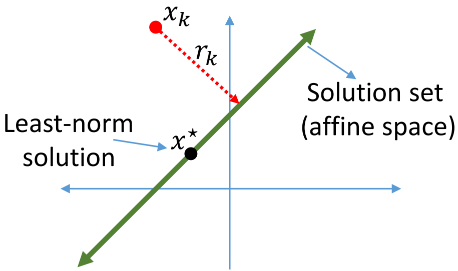
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$



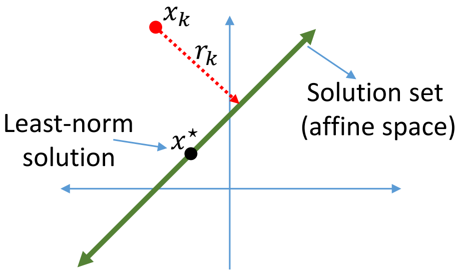
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$



$$\boxed{\mathbf{r}_k = \mathbf{Q}\left(\mathbf{x}_k - \mathbf{x}^\star\right)}$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

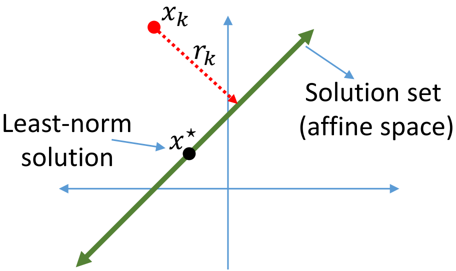
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \; \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$



$$\boxed{\mathbf{r}_k = \mathbf{Q}\,(\mathbf{x}_k - \mathbf{x}^\star)}$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

When does it converge?

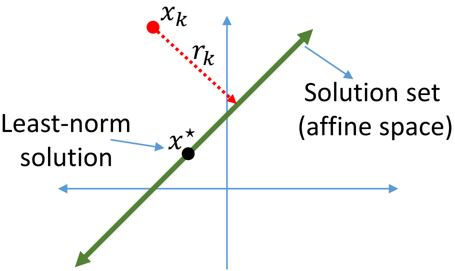
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k\to\infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$



$$\boxed{\mathbf{r}_k = \mathbf{Q}\,(\mathbf{x}_k - \mathbf{x}^\star)}$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

When does it converge?

When $\mathbf{A}$ is normal [1]

[1] Teke & Vaidyanathan, "Random Node-Asynchronous Updates on Graphs," *IEEE Trans. S.P., vol. 67, no. 11, 2019*

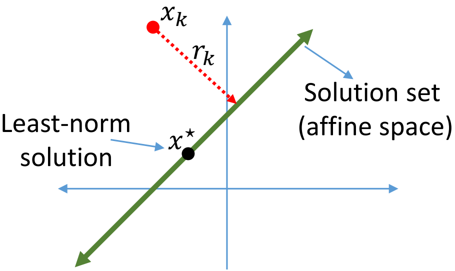
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k \to \infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})$$

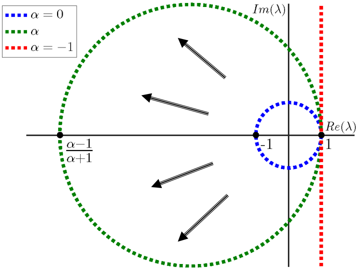


$$\mathbf{r}_k = \mathbf{Q}\,(\mathbf{x}_k - \mathbf{x}^\star)$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

When does it converge?

When $\mathbf{A}$ is normal [1]



[1] Teke & Vaidyanathan, "Random Node-Asynchronous Updates on Graphs," *IEEE Trans. S.P., vol. 67, no. 11, 2019*

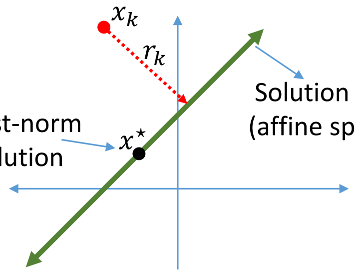
Mean-Squared Convergence of the Updates

$$x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1} + u_i, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

$$\lim_{k\to\infty} \mathbf{x}_k = ?$$

Where does it converge?

$$\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{u} \quad \Rightarrow \quad \boxed{\mathbf{u} \in \text{range}(\mathbf{I} - \mathbf{A})}$$

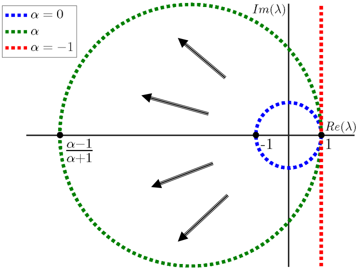


$$\boxed{\mathbf{r}_k = \mathbf{Q}\,(\mathbf{x}_k - \mathbf{x}^\star)}$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

When does it converge?

When $\mathbf{A}$ is normal [1]



$$\Downarrow$$

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0$$

[1] Teke & Vaidyanathan, "Random Node-Asynchronous Updates on Graphs," *IEEE Trans. S.P., vol. 67, no. 11, 2019*

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0 \qquad \Longleftrightarrow \qquad \rho(\boldsymbol{\Theta}\,\mathbf{S}) < 1$$

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] = 0 \qquad \Longleftrightarrow \qquad \rho(\mathbf{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P}) \otimes \mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^*-\mathbf{I}) \otimes (\mathbf{A}-\mathbf{I})\Big),$$

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0 \qquad \Longleftrightarrow \qquad \rho(\mathbf{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P}) \otimes \mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^* - \mathbf{I}) \otimes (\mathbf{A}-\mathbf{I})\Big),$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{P}\,(\mathbf{A}-\mathbf{I}), \qquad \mathbf{\Theta} = \mathbf{Q}^* \otimes \mathbf{Q}$$

[1] Teke & Vaidyanathan, "Node-Asynchronous Spectral Clustering on Directed Graphs," *ICASSP , 2020*

[2] Teke & Vaidyanathan http://systems.caltech.edu/dsp/students/oteke/files/icassp2020.pdf

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0 \qquad \Longleftrightarrow \qquad \rho(\mathbf{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P})\otimes\mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^*-\mathbf{I})\otimes(\mathbf{A}-\mathbf{I})\Big),$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{P}\,(\mathbf{A}-\mathbf{I}), \qquad \mathbf{\Theta} = \mathbf{Q}^*\otimes\mathbf{Q}$$

✓ Valid for *any* $\mathbf{A}$ (applicable to directed graphs)

[1] Teke & Vaidyanathan, "Node-Asynchronous Spectral Clustering on Directed Graphs," *ICASSP , 2020*

[2] Teke & Vaidyanathan http://systems.caltech.edu/dsp/students/oteke/files/icassp2020.pdf

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0 \qquad \Longleftrightarrow \qquad \rho(\boldsymbol{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P})\otimes\mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^*-\mathbf{I})\otimes(\mathbf{A}-\mathbf{I})\Big),$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{P}\,(\mathbf{A}-\mathbf{I}), \qquad \boldsymbol{\Theta} = \mathbf{Q}^* \otimes \mathbf{Q}$$

✓ Valid for *any* $\mathbf{A}$ (applicable to directed graphs)

✓ Ensures mean-squared convergence

[1] Teke & Vaidyanathan, "Node-Asynchronous Spectral Clustering on Directed Graphs," *ICASSP , 2020*

[2] Teke & Vaidyanathan http://systems.caltech.edu/dsp/students/oteke/files/icassp2020.pdf

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\big[\|\mathbf{r}_k\|_2^2\big] = 0 \qquad \Longleftrightarrow \qquad \rho(\mathbf{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P})\otimes\mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^*-\mathbf{I})\otimes(\mathbf{A}-\mathbf{I})\Big),$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{P}\,(\mathbf{A}-\mathbf{I}), \qquad \mathbf{\Theta} = \mathbf{Q}^* \otimes \mathbf{Q}$$

✓ Valid for *any* $\mathbf{A}$ (applicable to directed graphs)

✓ Ensures mean-squared convergence

✓ Robust to input noise [1, 2]

[1] Teke & Vaidyanathan, "Node-Asynchronous Spectral Clustering on Directed Graphs," *ICASSP , 2020*

[2] Teke & Vaidyanathan http://systems.caltech.edu/dsp/students/oteke/files/icassp2020.pdf

Mean-Squared Convergence of the Updates - Cont.

Theorem (The necessary and sufficient condition)

$$\lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] = 0 \qquad \Longleftrightarrow \qquad \rho(\mathbf{\Theta}\,\mathbf{S}) < 1$$

where

$$\mathbf{S} = \bar{\mathbf{A}}^* \otimes \bar{\mathbf{A}} + \Big((\mathbf{I}-\mathbf{P}) \otimes \mathbf{P}\Big)\,\mathbf{J}\,\Big((\mathbf{A}^* - \mathbf{I}) \otimes (\mathbf{A} - \mathbf{I})\Big),$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{P}\,(\mathbf{A} - \mathbf{I}), \qquad \mathbf{\Theta} = \mathbf{Q}^* \otimes \mathbf{Q}$$

✓ Valid for *any* $\mathbf{A}$ (applicable to directed graphs)

✓ Ensures mean-squared convergence

✓ Robust to input noise [1, 2]

✓ Can compute eigenvectors (to de discussed next)

[1] Teke & Vaidyanathan, "Node-Asynchronous Spectral Clustering on Directed Graphs," *ICASSP , 2020*

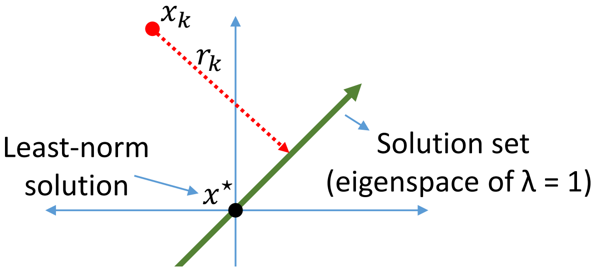
[2] Teke & Vaidyanathan http://systems.caltech.edu/dsp/students/oteke/files/icassp2020.pdf

Computing Eigenvectors

$$\mathbf{u} = \mathbf{0} \qquad \Longrightarrow \qquad x_k[i] = \begin{cases} \mathbf{a}_i\ \mathbf{x}_{k\text{-}1}, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

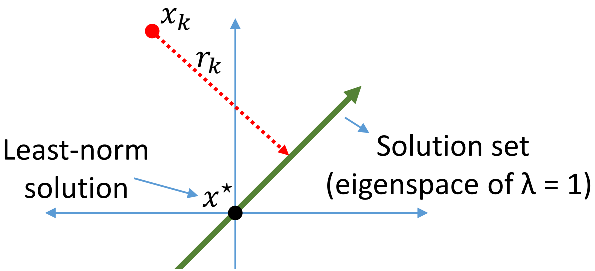
Computing Eigenvectors

$$\mathbf{u} = \mathbf{0} \qquad \Longrightarrow \qquad x_k[i] = \begin{cases} \mathbf{a}_i\,\mathbf{x}_{k\text{-}1}, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$



Computing Eigenvectors

$$\mathbf{u} = \mathbf{0} \quad \Longrightarrow \quad x_k[i] = \begin{cases} \mathbf{a}_i\, \mathbf{x}_{k\text{-}1}, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$

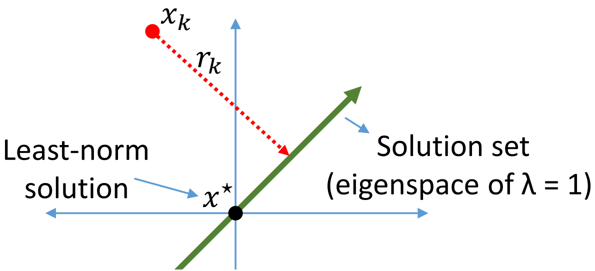


$$\boxed{\mathbf{r}_k = \mathbf{Q}\,\mathbf{x}_k}$$

$\mathbf{Q}$: Projection on $\text{null}^\perp(\mathbf{I} - \mathbf{A})$

Computing Eigenvectors

$$\mathbf{u} = \mathbf{0} \quad \Longrightarrow \quad x_k[i] = \begin{cases} \mathbf{a}_i \, \mathbf{x}_{k\text{-}1}, & \text{w.p.} \quad p_i, \\ x_{k\text{-}1}[i], & \text{w.p.} \quad 1\text{-}p_i. \end{cases}$$



$$\mathbf{r}_k = \mathbf{Q}\,\mathbf{x}_k$$

$\mathbf{Q}$: Projection on null$^\perp(\mathbf{I} - \mathbf{A})$

$$\lim_{k\to\infty} \mathbb{E}\left[\|\mathbf{r}_k\|_2^2\right] = \mathbf{0}$$

$$\Updownarrow$$

$\mathbf{x}_k$ converges to
an eigenvector of $\lambda = 1$

Outline

1 Graph Signal Processing and Spectral Clustering

2 Asynchronous Updates on Graphs

3 Autonomous Spectral Clustering

4 Conclusion

Use of Graph Polynomials

$$\mathbf{A}$$

Asynchronous

Use of Graph Polynomials

$\lambda = 1$

$\mathbf{A}$

Asynchronous

Use of Graph Polynomials

$$\lambda = 1 \quad \xrightarrow[\text{Asynchronous}]{\mathbf{A}} \quad \lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$$

Use of Graph Polynomials

$\lambda = 1$ $\xrightarrow[\text{Asynchronous}]{\mathbf{A}}$ $\lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$

$$H(\mathbf{A}) = \sum_{k=0}^{L} h_k \, \mathbf{A}^k$$

Asynchronous

Use of Graph Polynomials

$$\lambda = 1 \qquad \xrightarrow[\text{Asynchronous}]{\mathbf{A}} \qquad \lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$$

$$H(\lambda_2) = 1 \qquad H(\mathbf{A}) = \sum_{k=0}^{L} h_k \, \mathbf{A}^k$$

Asynchronous

Use of Graph Polynomials

$$\lambda = 1 \quad \xrightarrow[\text{Asynchronous}]{\mathbf{A}} \quad \lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$$

$$H(\lambda_2) = 1 \quad \xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \, \mathbf{A}^n} \quad \lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \, \mathbf{I})$$

(Fiedler Vector)

Use of Graph Polynomials

$\lambda = 1$ $\xrightarrow[\text{Asynchronous}]{\mathbf{A}}$ $\lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$

$H(\lambda_2) = 1$ $\xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \mathbf{A}^n}$ $\lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \mathbf{I})$

(Fiedler Vector)

$H(\mathbf{A}) \sim \mathbf{A}$

L^{th} order $\Rightarrow$ L-hop neighborhood

Use of Graph Polynomials

$$\lambda = 1 \quad \xrightarrow[\text{Asynchronous}]{\mathbf{A}} \quad \lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$$

$$\boxed{H(\lambda_2) = 1} \quad \xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \, \mathbf{A}^n} \quad \lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \, \mathbf{I})$$

(Fiedler Vector)

$H(\mathbf{A}) \sim \mathbf{A}$

L^{th} order $\Rightarrow$ L-hop neighborhood

Use of Graph Polynomials

$\lambda = 1$ $\xrightarrow[\text{Asynchronous}]{\mathbf{A}}$ $\lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$

$H(\lambda_2) = 1$ $\xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \mathbf{A}^n}$ $\lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \mathbf{I})$

(Fiedler Vector)

$H(\mathbf{A}) \sim \mathbf{A}$

L^{th} order $\Rightarrow$ L-hop neighborhood

$$\mathbf{\Phi} = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^L \\ 1 & \lambda_2 & \cdots & \lambda_2^L \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^L \end{bmatrix}$$

$$\mathbf{h} = [h_0 \;\; h_1 \;\; \cdots \;\; h_L]^T$$

Use of Graph Polynomials

$$\lambda = 1 \xrightarrow[\text{Asynchronous}]{\mathbf{A}} \lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$$

$$H(\lambda_2) = 1 \quad \xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \mathbf{A}^n} \quad \lim_{k\to\infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \mathbf{I})$$

(Fiedler Vector)

$H(\mathbf{A}) \sim \mathbf{A}$

L^{th} order $\Rightarrow$ L-hop neighborhood

$$\mathbf{\Phi} = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^L \\ 1 & \lambda_2 & \cdots & \lambda_2^L \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^L \end{bmatrix}$$

$$\mathbf{h} = [h_0 \;\; h_1 \;\; \cdots \;\; h_L]^T$$

Linear Programming:

$$\max_{\mathbf{h}} \quad c \quad \text{s.t.} \quad \begin{array}{c} \phi_2 \, \mathbf{h} = 1 \\ |\bar{\mathbf{\Phi}} \, \mathbf{h}| \leqslant (1 - c) \, \mathbb{1}_{N\text{-}1} \end{array}$$

$$c \geqslant 0$$

Use of Graph Polynomials

$\lambda = 1$ $\xrightarrow[\text{Asynchronous}]{\mathbf{A}}$ $\lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \mathbf{I})$

$H(\lambda_2) = 1$ $\xrightarrow[\text{Asynchronous}]{H(\mathbf{A}) = \sum_{n=0}^{L} h_n \mathbf{A}^n}$ $\lim_{k \to \infty} \mathbf{x}_k \in \text{null}(\mathbf{A} - \lambda_2 \mathbf{I})$

(Fiedler Vector)

$H(\mathbf{A}) \sim \mathbf{A}$

L^{th} order $\Rightarrow$ L-hop neighborhood

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^L \\ 1 & \lambda_2 & \cdots & \lambda_2^L \\ \vdots & \vdots & & \vdots \\ 1 & \lambda_N & \cdots & \lambda_N^L \end{bmatrix}$$

$$\mathbf{h} = [h_0 \;\; h_1 \;\; \cdots \;\; h_L]^T$$

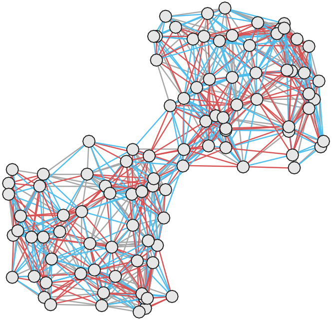
Linear Programming:

$$\max_{\mathbf{h}} \quad c \quad \text{s.t.} \quad \begin{aligned} \phi_2 \, \mathbf{h} &= 1 \\ |\bar{\boldsymbol{\Phi}} \, \mathbf{h}| &\leqslant (1 - c) \, \mathbb{1}_{N\text{-}1} \end{aligned}$$

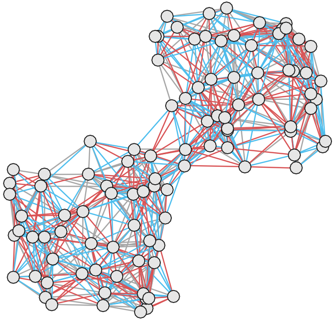
$$c \geqslant 0$$

$L = 2$ *works in practice*

A Numerical Application



A Numerical Application $(\mathbf{L})$



$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

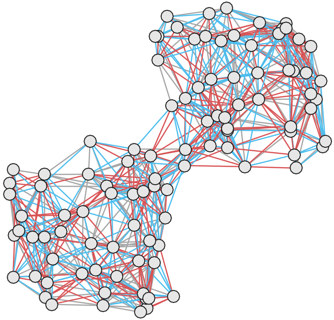
(Fiedler Value)

[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS, 2005*

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI, 2005*

A Numerical Application $(\mathbf{L})$



$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

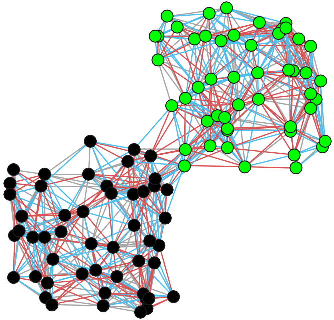
(Fiedler Vector)

[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, 1973

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS*, 2005

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI*, 2005

A Numerical Application $(\mathbf{L})$



$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

$$\mathbf{x} = \text{sign}(\mathbf{v}_2)$$

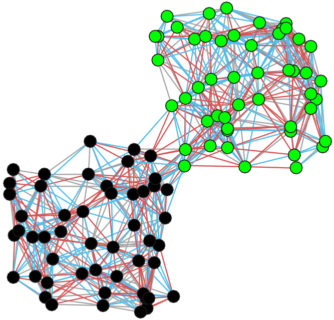
[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, 1973

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS*, 2005

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI*, 2005

A Numerical Application

(L)



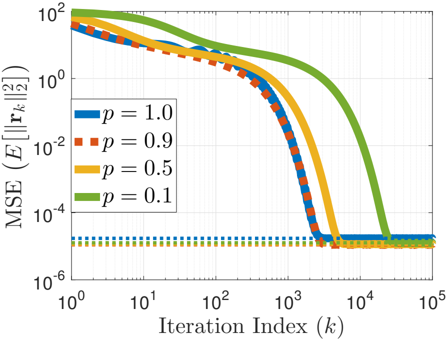
$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

$$\mathbf{x} = \text{sign}(\mathbf{v}_2)$$

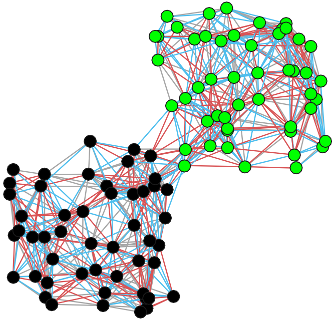


[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS, 2005*

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI, 2005*

A Numerical Application (L)



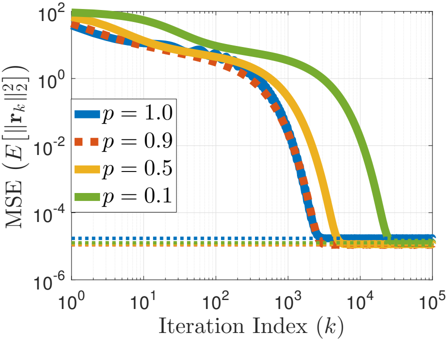
$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

$$\mathbf{x} = \text{sign}(\mathbf{v}_2)$$



p : Update probabilities

$p = 1$ is the synchronous case

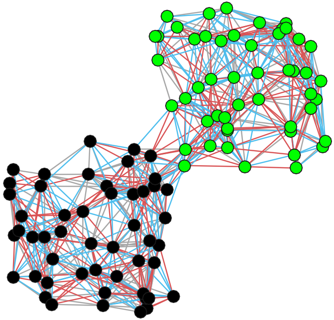
[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS, 2005*

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI, 2005*

A Numerical Application

(L)



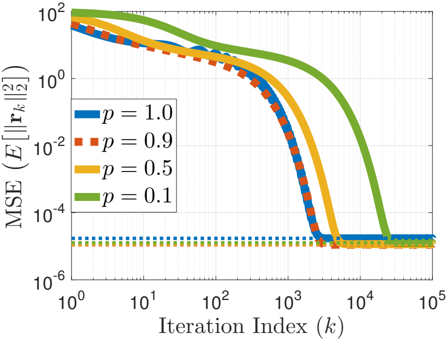
$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

$$\mathbf{x} = \text{sign}(\mathbf{v}_2)$$



p : Update probabilities

$p = 1$ is the synchronous case

Input noise

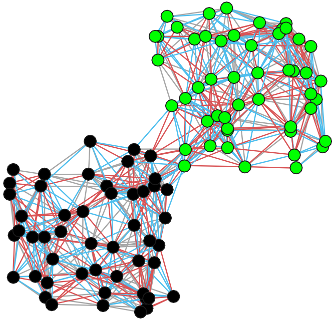
[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS, 2005*

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI, 2005*

A Numerical Application

(L)



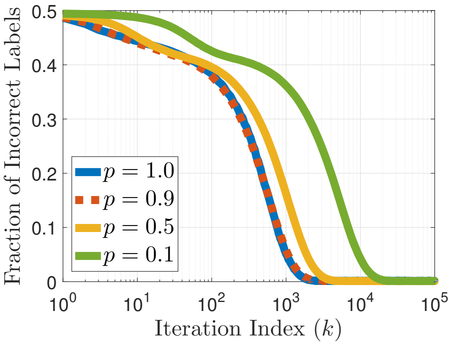
$$0 = \lambda_1 < |\lambda_2| \leqslant \cdots \leqslant |\lambda_N|$$

(Fiedler Value)

$$\mathbf{L}\,\mathbf{v}_2 = \lambda_2\,\mathbf{v}_2$$

(Fiedler Vector)

$$\mathbf{x} = \text{sign}(\mathbf{v}_2)$$



p : Update probabilities

$p = 1$ is the synchronous case

Input noise

[1] Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal, 1973*

[2] Zhou, Huang, and Scholkopf, "Learning from labeled and unlabeled data on a directed graph," *NIPS, 2005*

[3] Pentney and Meila, "Spectral clustering of biological sequence data," *National Conf. on AI, 2005*

Outline

1 Graph Signal Processing and Spectral Clustering

2 Asynchronous Updates on Graphs

3 Autonomous Spectral Clustering

4 Conclusion

Conclusions & Future Work

- Random asynchronous fixed-point iterations
- Mean-Squared convergence
- Can compute eigenvectors with polynomials
- Find the Fiedler vector for clustering

Conclusions & Future Work

- Random asynchronous fixed-point iterations
- Mean-Squared convergence
- Can compute eigenvectors with polynomials
- Find the Fiedler vector for clustering

- *Heuristic approach for the filter design*
- *Optimal filter design?*

Conclusions & Future Work

- Random asynchronous fixed-point iterations
- Mean-Squared convergence
- Can compute eigenvectors with polynomials
- Find the Fiedler vector for clustering

- *Heuristic approach for the filter design*
- *Optimal filter design?*

- *Optimal probabilities?*

Conclusions & Future Work

- Random asynchronous fixed-point iterations
- Mean-Squared convergence
- Can compute eigenvectors with polynomials
- Find the Fiedler vector for clustering

- *Heuristic approach for the filter design*
- *Optimal filter design?*
- *Optimal probabilities?*
- *Graph independent filter design?*

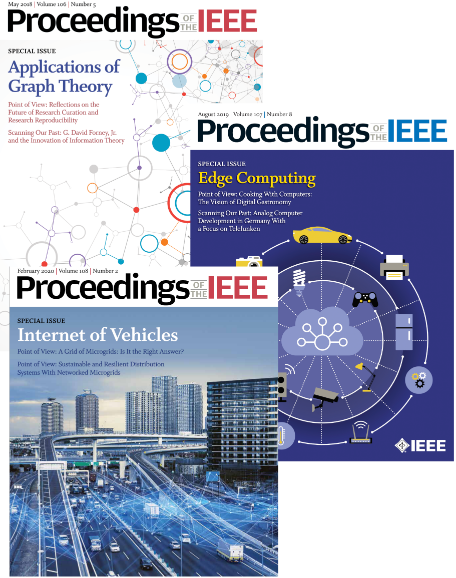
Conclusions & Future Work

- Random asynchronous fixed-point iterations
- Mean-Squared convergence
- Can compute eigenvectors with polynomials
- Find the Fiedler vector for clustering

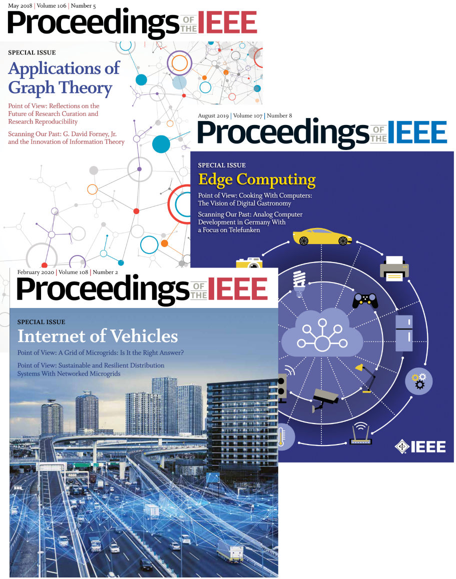
- *Heuristic approach for the filter design*
- *Optimal filter design?*
- *Optimal probabilities?*
- *Graph independent filter design?*

Thank you!

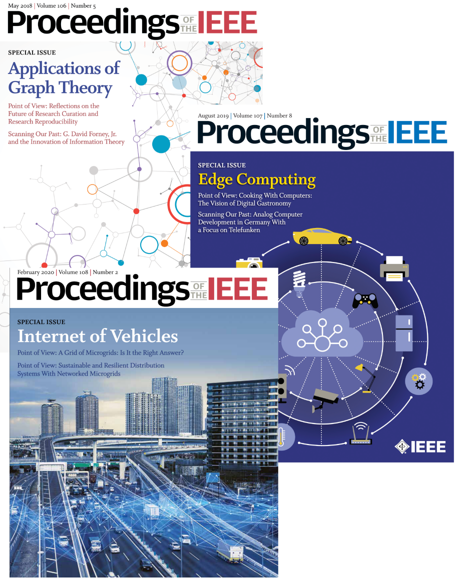
New Horizons ...



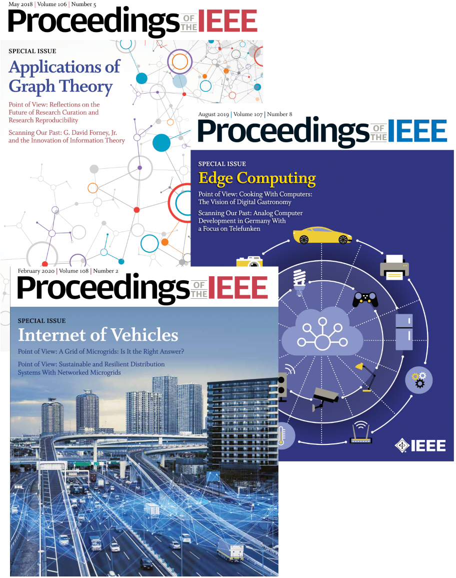
New Horizons ...



New Horizons ...



New Horizons ...



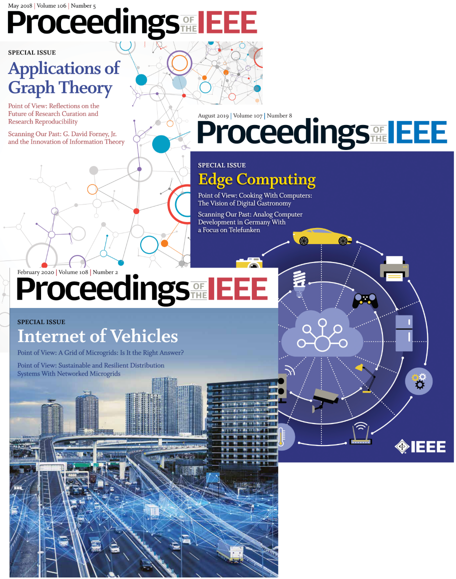
Randomized numerical linear algebra

- *Emerging field!*
- *Approximate v.s. Exact*

[1] Drineas & Mahoney, "Lectures on randomized numerical linear algebra," *The Mathematics of Data, 2018*

[2] Martinsson & Tropp, "Randomized Numerical Linear Algebra: Foundations and Algorithms," *arXiv, 2020*

New Horizons ...



Randomized numerical linear algebra

- *Emerging field!*
- *Approximate v.s. Exact*

> *Randomization opens up a new dimension!*
>
> *How can we exploit it for better?*

[1] Drineas & Mahoney, "Lectures on randomized numerical linear algebra," *The Mathematics of Data, 2018*

[2] Martinsson & Tropp, "Randomized Numerical Linear Algebra: Foundations and Algorithms," *arXiv, 2020*