Cellular Sheaves

How do we attach data to a space?

Take a regular cell complex $X$.

**Stalks:** vector space $\mathcal{F}(\sigma)$ for each cell $\sigma \in X$

**Restriction maps:** $\mathcal{F}_{\sigma \leq \tau}: \mathcal{F}(\sigma) \to \mathcal{F}(\tau)$ for $\sigma \leq \tau$

**Commutativity:** $\mathcal{F}_{\tau \leq \chi} \circ \mathcal{F}_{\sigma \leq \tau} = \mathcal{F}_{\nu \leq \chi} \circ \mathcal{F}_{\sigma \leq \nu}$

Sheaves specify **consistency relations** for data on $X$:

$$\mathcal{F}_{u \leq e} x_u = \mathcal{F}_{v \leq e} x_v$$
Sheaf Cohomology

**Cochains:** valued in stalks over \( \text{dim}-k \) cells

\[
C^k(X; \mathcal{F}) = \bigoplus_{\text{dim}(\sigma)=k} \mathcal{F}(\sigma)
\]

**Coboundary** \( \delta^k : C^k(X; \mathcal{F}) \to C^{k+1}(X; \mathcal{F}) \)

\[
(\delta^k x)(\sigma) = \sum_{\tau \subseteq \sigma} [\tau : \sigma] \mathcal{F}_{\tau \subseteq \sigma} x(\tau)
\]

\[
C^0(X; \mathcal{F}) \xrightarrow{\delta} C^1(X; \mathcal{F}) \xrightarrow{\delta} C^2(X; \mathcal{F}) \xrightarrow{\delta} C^3(X; \mathcal{F})
\]

Check: \( \delta^2 = 0 \). Sheaf cohomology is \( H^k(X; \mathcal{F}) = \ker \delta^k / \text{im} \delta^{k-1} \).

\( H^0(X; \mathcal{F}) \) represents **global sections**:
choices of \( x_v \in \mathcal{F}(v) \) for all \( v \) such that
\( \mathcal{F}_{v \leq e} x_v = \mathcal{F}_{u \leq e} x_u \) for every edge \( e = u \sim v \)
Sheaf Laplacians are local linear operators on spaces of cochains. In particular, $L_{\mathcal{F}}$ is a local graph operator:

$$(L_{\mathcal{F}}x)_v = \sum_{e=u \sim v} \mathcal{F}_{v \leq e}(\mathcal{F}_{v \leq e}x_v - \mathcal{F}_{u \leq e}x_u)$$

Useful for understanding networks and networked systems.
What can sheaf Laplacians do?

**Distributed Systems**
- Distributed consensus
- Distributed optimization

**Network Analysis**
- Learning network structure
- Richer network models
Distributed Consensus

Agents in a graph send messages to neighbors to reach agreement

Follow a dynamical system:
\[ \dot{x} = -Lx \]

Trajectories converge to constant functions
Messages are local infinitesimal differences

Now with sheaves...
Consensus on a section:
\[ \dot{x} = -L_\mathcal{F}x \]

converges to the projection of \( x_0 \) onto \( H^0(G; \mathcal{F}) \)
e.g.: mesh network of sensors flock choosing a direction
Distributed Optimization

Independent processing units optimizing a joint function
Each node has a local function $f_v : \mathbb{R}^k \rightarrow \mathbb{R}$

Want to solve

$$\min_x \sum_v f_v(x)$$

Replace this with a local version

$$\min_{x_v} \sum_v f_v(x_v) \text{ s.t. } x_u = x_v \text{ for all } u \sim v$$

Typical approach: **local optimization** (gradient descent)
+ **consensus** (local averaging)
Homological Programming

Constraint: parameters are *locally constant* \( x_u = x_v \) for \( u \sim v \)

What about more complex constraints? \( \mathcal{F}_{u \subseteq e} x_u = \mathcal{F}_{v \subseteq e} x_v \) for \( u \sim v \)

These local linear constraints integrate into a sheaf homological constraint

\[
\min \sum_v f_v(x_v) \quad \text{s.t. } x \in H^0(G; \mathcal{F})
\]

**Why?** Perhaps: sensors with overlapping observations, different coordinate systems

Sheaf Laplacian provides the consensus process to integrate local gradient descent for distributed optimization
Local constraints $A_{uv}x_u = A_{vu}x_v$ integrate into a constraint $x \in H^0(G; \mathcal{F})$.

$x \in H^0(G; \mathcal{F})$ if and only if $L_\mathcal{F}x = 0$

$$\min \sum_v f_v(x_v) \text{ s.t. } L_\mathcal{F}x = 0$$

$$\mathcal{L}(x, z) = \sum_v f_v(x_v) + x^T L_\mathcal{F}x + z^T L_\mathcal{F}x$$

Saddle-point dynamics (local!):

$$\dot{x} = -\frac{\partial \mathcal{L}}{\partial x} = -\sum_v \nabla f_v(x_v) - 2L_\mathcal{F}x - L_\mathcal{F}z$$

$$\dot{z} = \frac{\partial \mathcal{L}}{\partial z} = L_\mathcal{F}x$$

By general results on saddle-point dynamics, this converges to optimum.
What can sheaf Laplacians do?

Distributed Systems
- Distributed consensus
- Distributed optimization

Network Analysis
- Learning network structure
- Richer network models
Smooth signals on a graph

Small $\ell^2$ variation over edges: $\sum_{u \sim v} w_{uv} (x_u - x_v)^2 = \langle x, Lx \rangle$ is small.

The smoothest signals are constant.
Smooth signals on a sheaf

Small $\ell^2$ discrepancy over edges: $\langle x, L_F x \rangle = \sum_{u \sim v} \| F_{u \leq e} x_u - F_{v \leq e} x_v \|^2$ is small.

The smoothest signals are sections.
Learning Sheaves

From a collection of smooth vertex signals $x^i$, recover the graph or sheaf. (assume vertex stalks are known)

**Graphs:** Edges between nodes where signals tend to be nearly **equal**

**Sheaves:** Edges between nodes where signals tend to nearly **satisfy a linear relation**
An optimization problem:

$$\min \sum_i \langle x^i, L_F x^i \rangle \quad \text{s.t. } L_F \text{ is a sheaf Laplacian}$$

How do we tell if $L_F$ is a sheaf Laplacian?

$$L_F = \delta^* \delta = \sum_e \delta_e^* \delta_e = \sum_e L_e$$

$L_e$ can be any symmetric positive semidefinite matrix (with correct sparsity)

Sheaf Laplacians are a convex cone $\rightarrow$ efficient algorithms with SDP
Learning Sheaves

\[
\min \sum_i \langle x^i, L_F x^i \rangle + f_c(L_F) + f_s(L_F) \quad \text{s.t. } L_F \text{ is a sheaf Laplacian}
\]

\[
f_c(L_F) = -\sum_v \log \text{tr}(L_F(v, v)) \quad f_s(L_F) = \sum_e \|L_e\|_F^2
\]

Ensures graph is connected and \( L_F \neq 0 \)
Controls sparsity of \( L_F \)

Can also optimize over smaller classes of sheaves:

\[
\begin{bmatrix}
W_{ij} & -W_{ij} \\
-W_{ij} & W_{ij}
\end{bmatrix}
\quad \begin{bmatrix}
I & -\rho_{ij} \\
-\rho^*_{ij} & I
\end{bmatrix}
\]

Matrix-weighted graphs
\( O(n) \) bundles
From Sheaves to Applications

slides + papers at jakobhansen.org

Cellular Sheaves

Hodge Theory

Sheaf Consensus

Sheaf Laplacians

Distributed Systems

Sheaf Learning

Network Science

slides + papers at jakobhansen.org