A Sheaf Theoretic Framework for Multi-Agent Model Predictive Control

Sam Cohen and Trevor Gross

Background

MPC

- MPC is a broadly useful control method

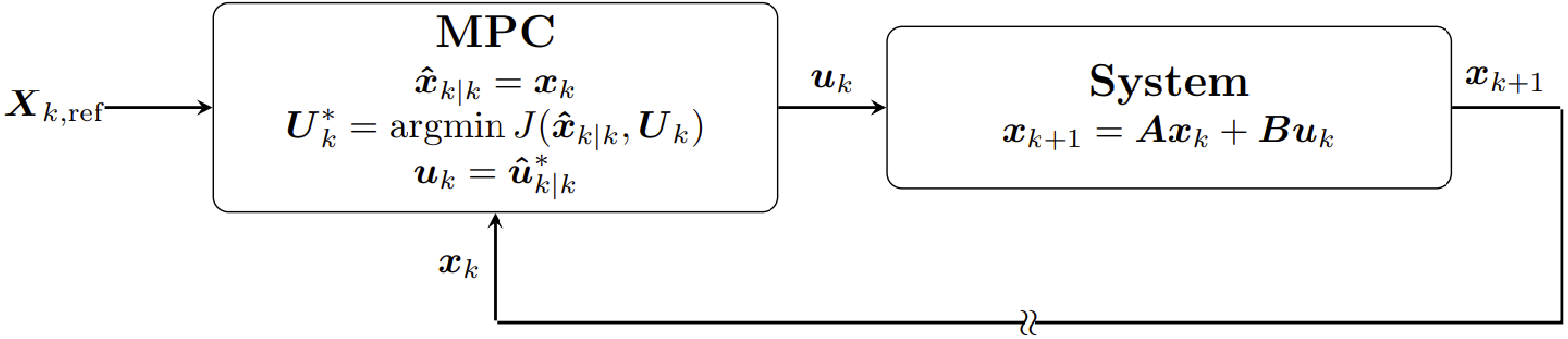
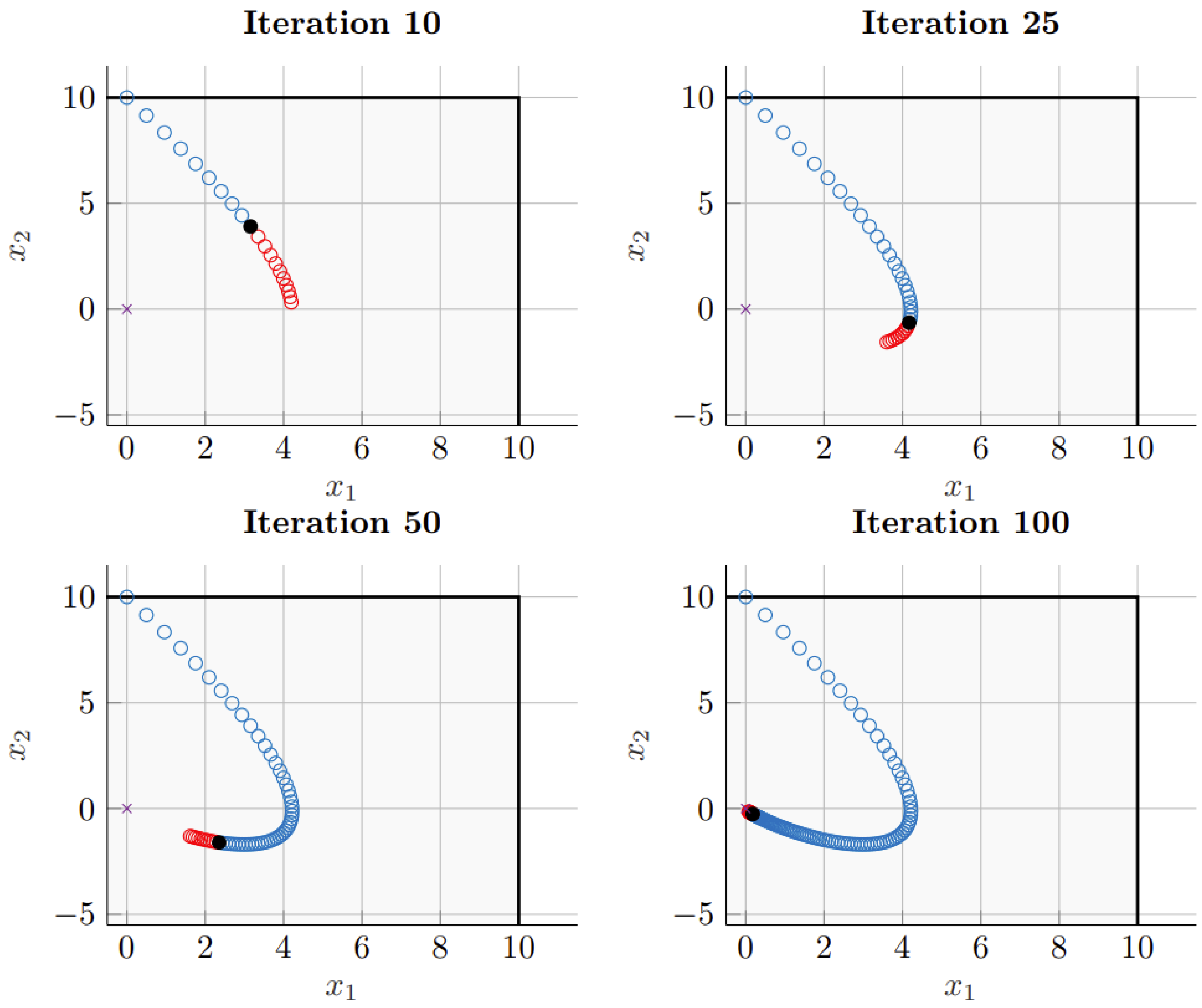


Figure 1: MPC scheme

MPC



Iteration 10
Iteration 25
Iteration 50
Iteration 100
x_1
x_2
10
5
0
−5
0
2
4
6
8
10

Multi-agent MPC

We consider a collection of $m \in \mathbb{N}$ agents with linear, discrete-time dynamics

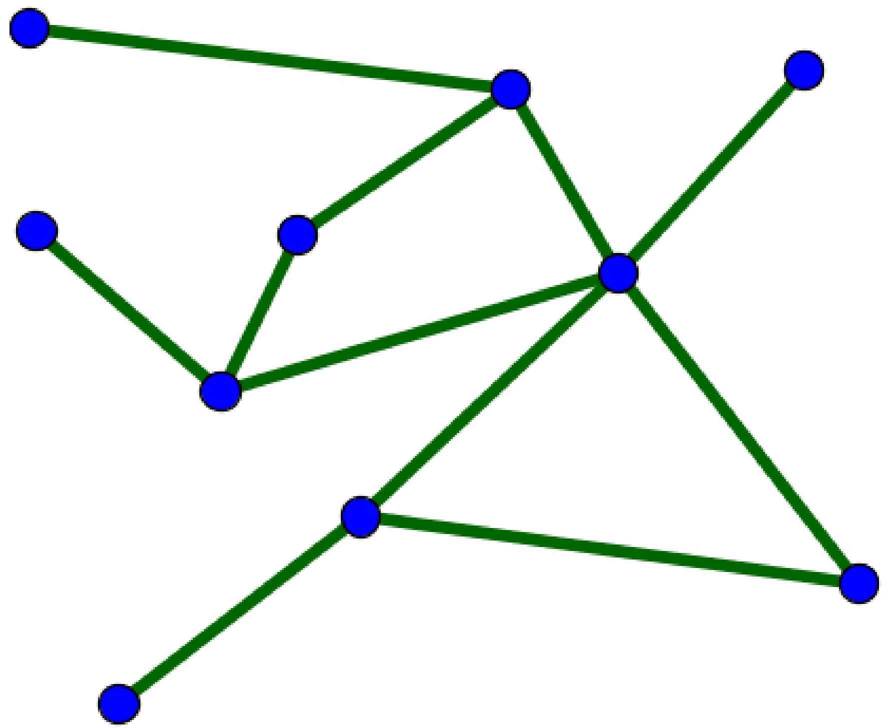
$$x^i(t+1) = A_i x^i(t) + B_i u^i(t), \quad x^i(0) = x_0^i, \tag{1a}$$

$$y^i(t) = C_i x^i(t), \tag{1b}$$

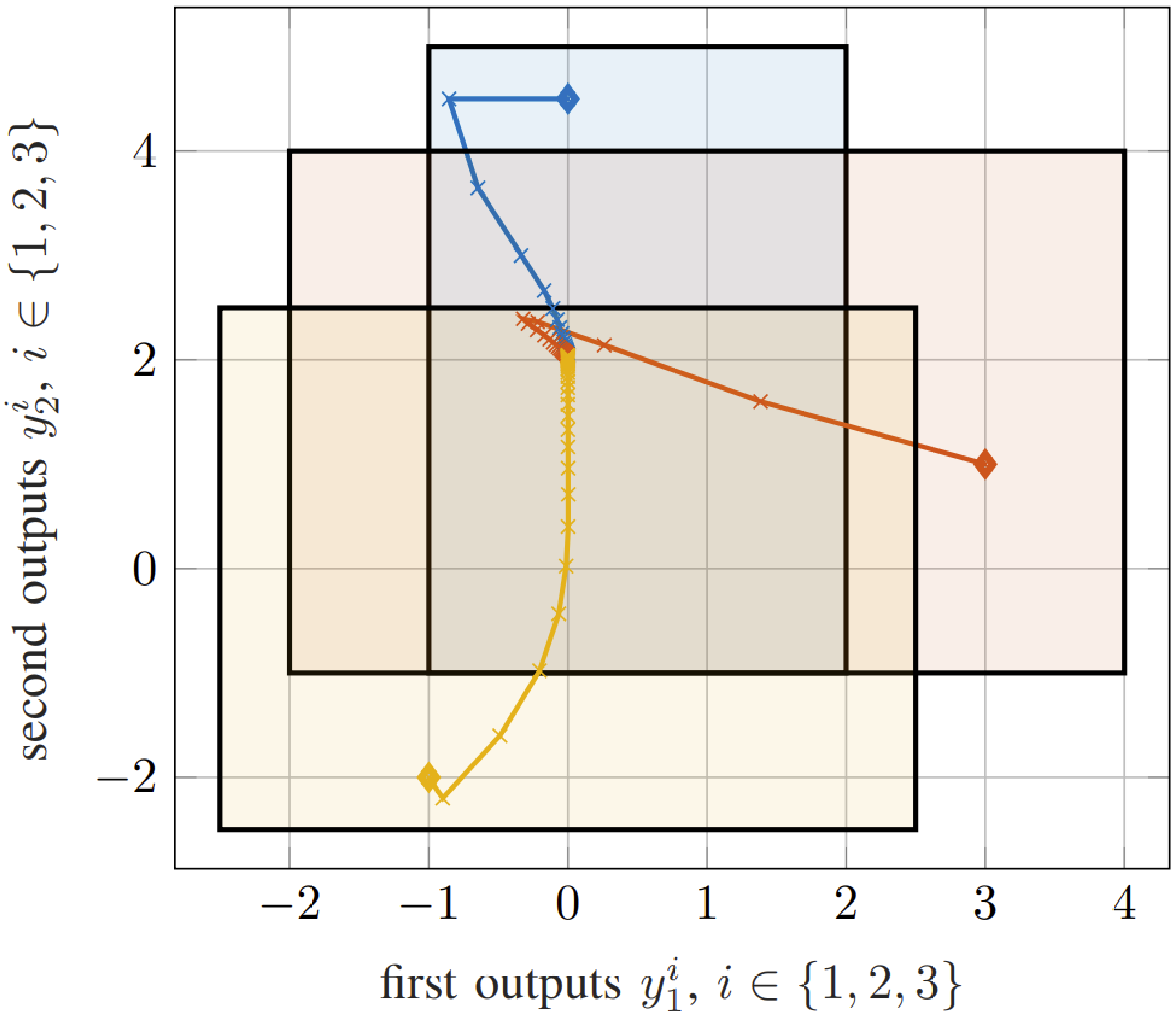
Multi-agent MPC

The communication topology of the multi-agent system is given by a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the set of nodes, i.e. agents, $\mathcal{V} = \{1, \ldots, m\}$ and edges $\mathcal{E}$. Agents may bidirectionally share information with each other if they are connected by an edge. The induced set of neighbours of agent i is given as $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. The symmetric graph Laplacian is defined as $L = [l_{ij}] \in \mathbb{R}^{m \times m}$ where

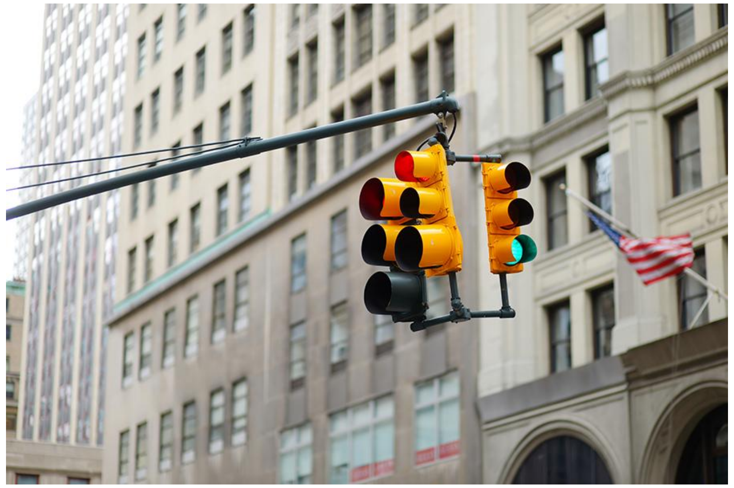
$$l_{ij} = \begin{cases} -1, & j \in \mathcal{N}_i, \\ |\mathcal{N}_i|, & j = i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

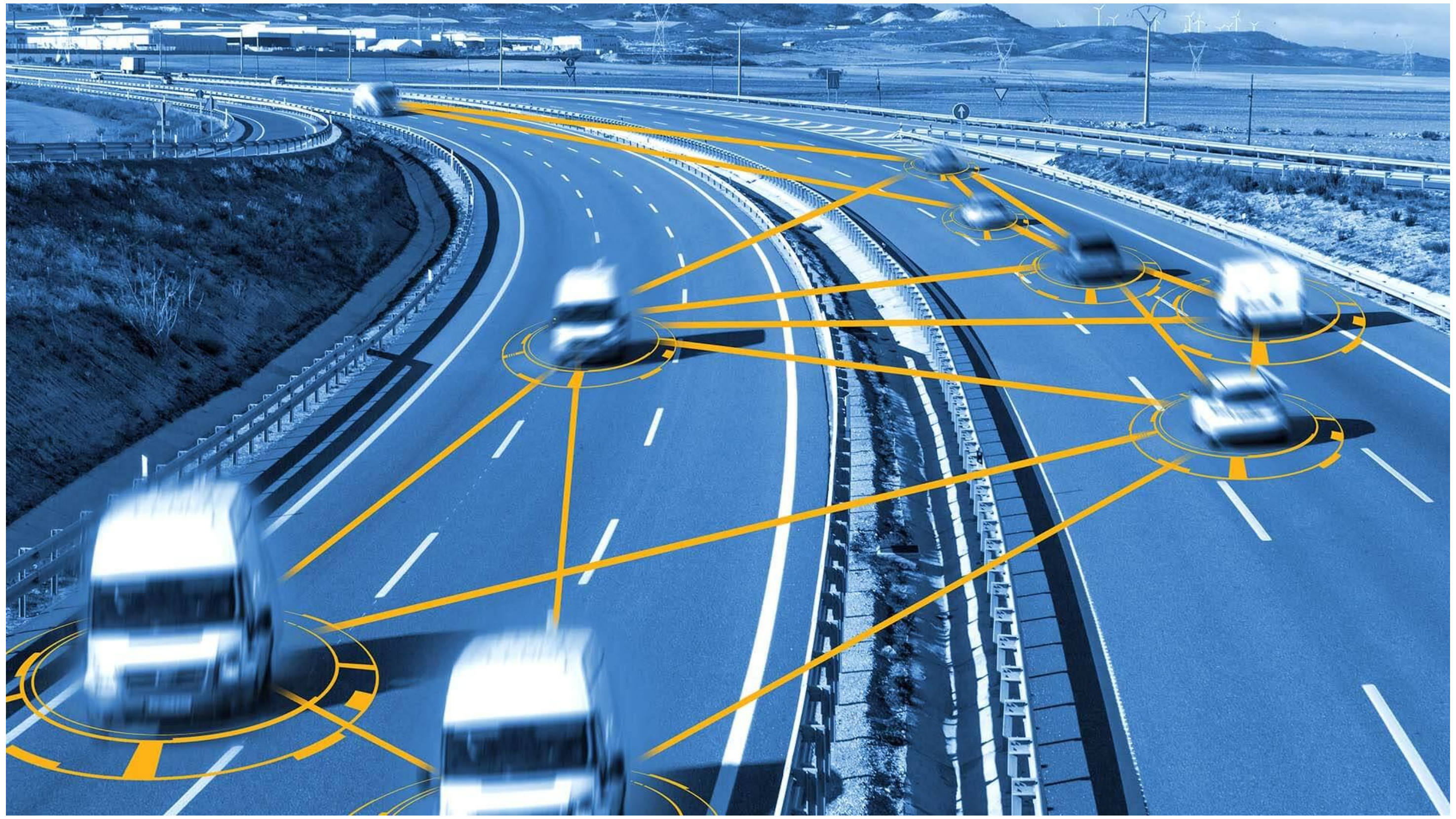


Multi-Agent MPC



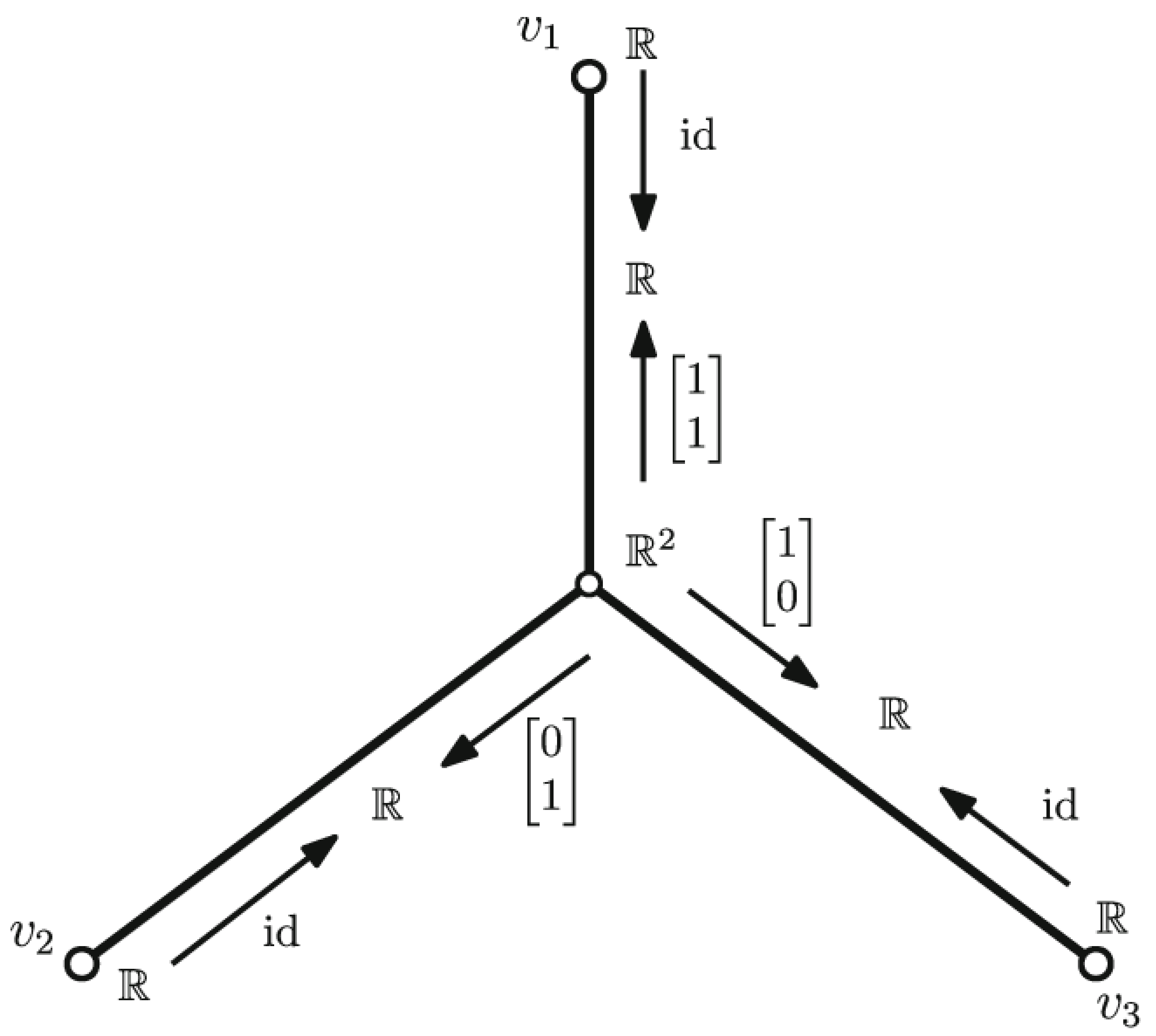
Multi-agent MPC





Cellular Sheaves

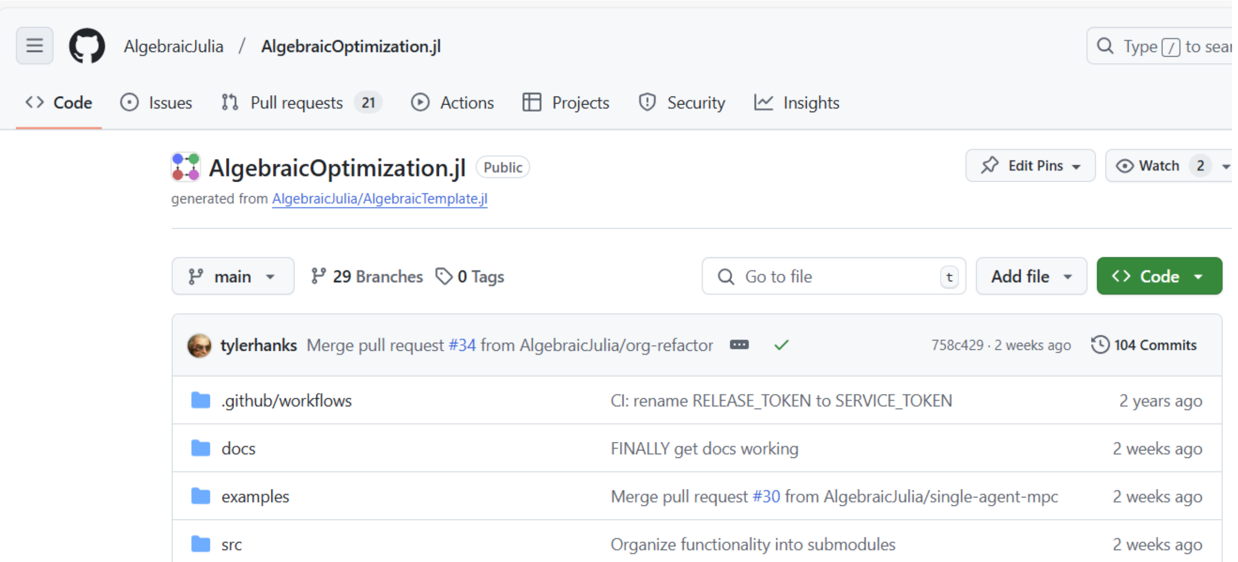
- Connected graph G = (V, E)
- Vertex and edge stalks
- Restriction maps
- Sheaf Laplacian

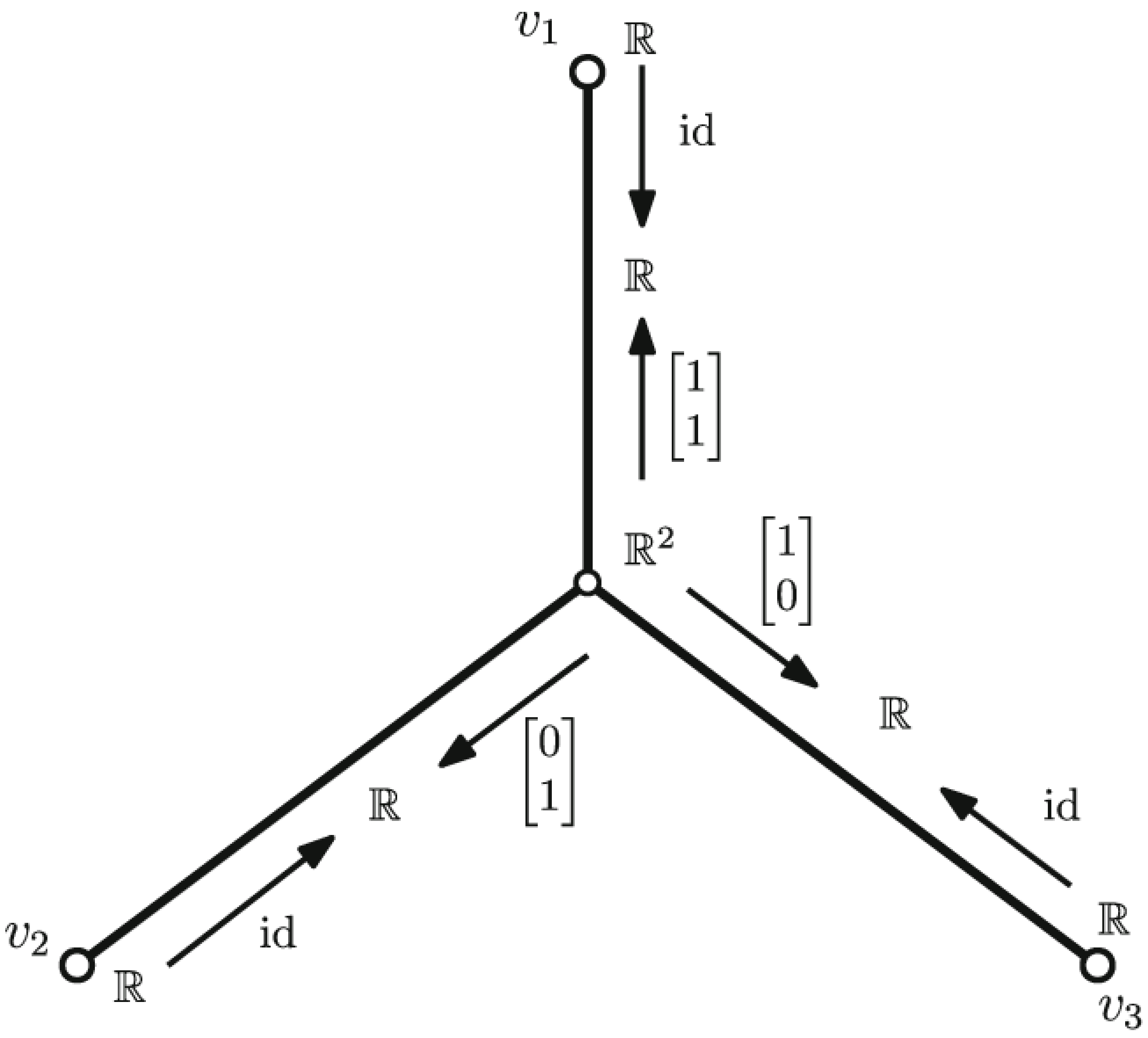


Cellular Sheaf Optimization

Minimize $\sum f(x)$

Subject to $Lx = 0$





A Sheaf Theoretic Framework for Multi-Agent Model Predictive Control

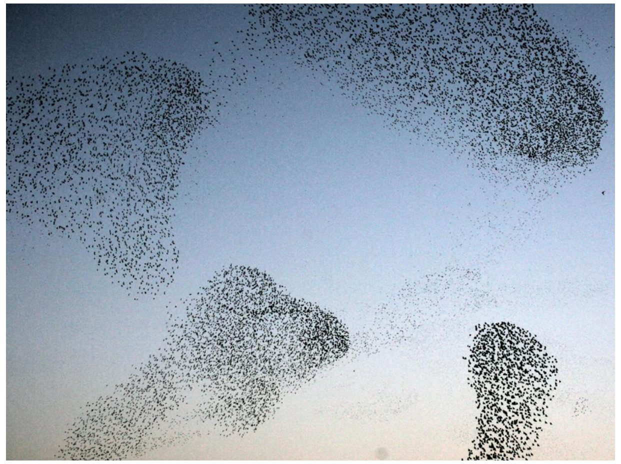
- We can do multi-agent MPC using cellular sheaves

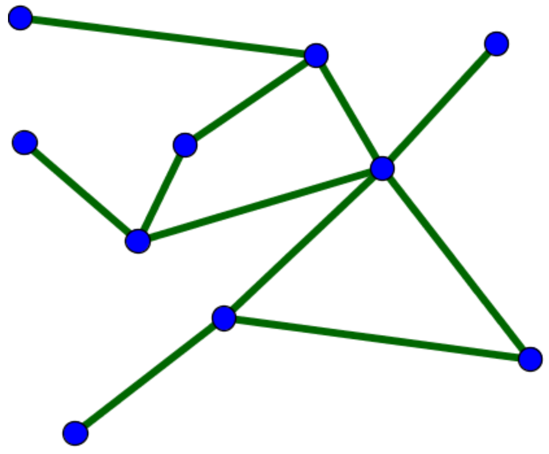
A. Flocking Problem

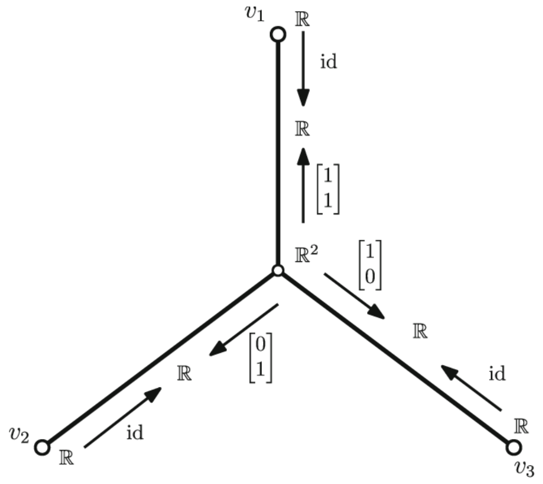
Develop a decentralized consensus protocol $u_i(k) = f_i(p_i(k), p_j(k), q_i(k), q_j(k)), j \in \mathcal{N}_i(k), i \in \mathcal{V}$, satisfying (2) and

$$\begin{aligned} \lim_{k\to\infty} \|q_i(k) - q_j(k)\| &= 0, \\ \lim_{k\to\infty} \|p_i(k) - p_j(k)\| &= 0, \quad \forall i, j \in \mathcal{V}. \end{aligned} \tag{4}$$

Note that "flocking" refers to a group of agents moving or migrating together with the same velocity.







Implementation

ADMM

- Iterative consensus algorithm

$$
\begin{aligned}
x_i^{k+1} &:= \operatorname{argmin}_{x_i} f_i(x_i) + (\rho/2)\|x_i - z_i^k + y_i^k\|_2^2 \\
\mathbf{z}^{k+1} &:= \Pi_{\mathcal{C}}(\mathbf{x}^{k+1} + \mathbf{y}^k) \\
y_i^{k+1} &:= y_i^k + x_i^{k+1} - z_i^{k+1}
\end{aligned}
$$

Implementation

1. Setup dynamical system $s(A, B, C)$ where $x' = Ax + Bu$
2. Setup agent objective function $x^T Q x + u^T R u$
3. Set time horizon and control bounds
4. Setup communication pattern, cellular sheaf c
5. Set initial state, MPC problem, and ADMM algorithm parameters
6. Run solver on these conditions
7. Plot resulting trajectories

JuMP

- Modeling language for mathematical optimization in Julia

```julia
function optimize_step(x_k, Q, R, s::DiscreteLinearSystem, x_target, ρ::Real)
    # Constants
    horizon = 10  # Prediction horizon

    # Define the optimization model using Ipopt solver
    model = Model(Ipopt.Optimizer)
    set_silent(model)  # Suppress solver output

    # Decision variables: state trajectory (x) and control inputs (u)
    @variable(model, x[1:2, 1:horizon])
    #@variable(model, u[1:2, 1:horizon])
    @variable(model, -20 <= u[1:2, 1:horizon] <= 20)  # Control limits

    # Initial state and control constraints
    @constraint(model, x[:, 1] .== x_k)
    #@constraint(model, u[:, 1] .== u_k)

    # System dynamics constraints: x[k+1] = Ax[k] + Bu[k]
    for k = 1:horizon-1
        @constraint(model, x[:, k+1] .== s.A * x[:, k] + s.B * u[:, k])
    end

    # Define the cost function (sum of squared states and inputs over the horizon)
    @objective(model, Min, sum((x[:, k]' * Q * x[:, k]) + (u[:, k]' * R * u[:, k]) for k = 1:horizon)) + ρ / 2 * ((x[:, horizon] - x_target)' * Q * (x[:, horizon] - x_target))

    # Solve the optimization problem
    optimize!(model)

    # Return the optimized control input for the next time step
    return value.(x[:, horizon]), value.(u[:, 1])
end
```

Global Sections

- Computes the projection of x onto the space of global sections of s

- Solve Laplacian: $Lx = b$

- b encodes the displacement

```
function nearest_section(s::CellularSheaf, x, b)
    d = coboundary_map(s)

    eL = LinearOperator(d) * LinearOperator(d')

    rhs = d * x - b

    y, stats = cg(eL, Array(rhs))

    return BlockArray(x - d' * y, s.vertex_stalks)

end
```

Documentation

[Moving Formation Example · AlgebraicOptimization.jl]

Paper submitted

Author or Proposer Samuel Cohen's CDC 2025 Workspace

Home Access Workspace PIN Refresh Log out Contact the CDC 2025 organizers Contact Technical Support

All deadlines are 23:59:59 Pacific Time. Current time 08:02:17

Samuel Cohen 232715 (Author). Your current session expires in 59:22

Cancel the page Download the Get started guide Pdf Test

Samuel Cohen's CDC 2025 Submissions

Check the column 'Status' for the status of your submission, and the column 'Actions for the corresponding author' for pending actions and deadlines

Move your mouse pointer over 'Choose an option' to open a menu with several useful options.
Click anywhere within the browser window to close the menu

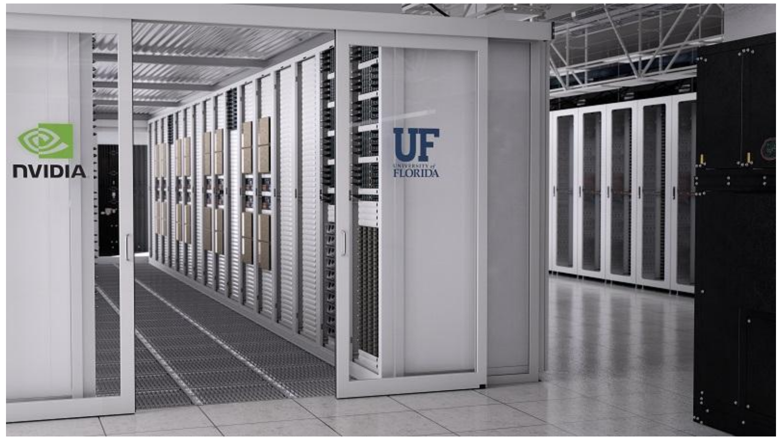
Important notice
Links in the column 'Actions for the corresponding author' are ONLY available to the corresponding author (denoted by * in the column 'Authors or proposers')

Number	Type of submission	Type of presentation	Authors or proposers *Corresponding author	Title	Profile	Status	Actions for the corresponding author ▶ Mandatory action ▶ Optional action Follow the link if available	Options (Submission details, files,...)
1443	Invited Session Paper		Tyler Hanks*, Hans Riess, Cohen Samuel, Trevor Gross, Matthew Hale, James Fairbanks (171575, 143002, 232715, 232723, 97396, 170931)	Distributed Multi-agent Coordination over Cellular Sheaves (Code 14m53)	Invited Papers	Under review	• No action is required at this time	Choose an option

Up Next

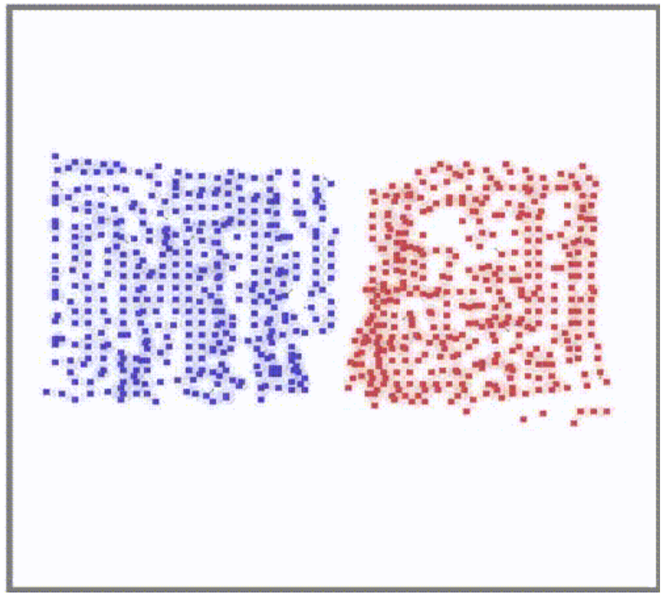
Paper extension

- Multithreaded implementation
- Distributed implementation
- Runtime analysis



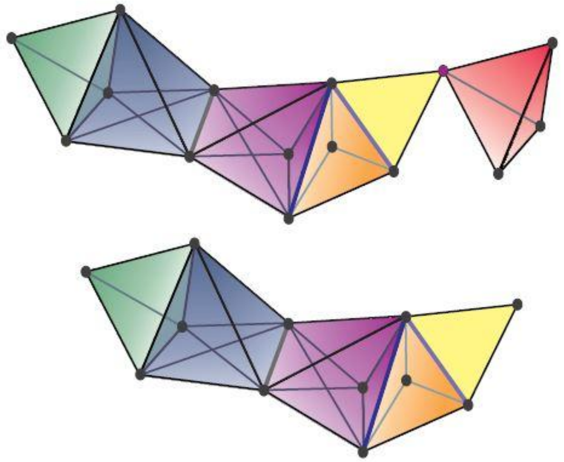
ML Applications

- Multitask learning
- Multiagent reinforcement learning



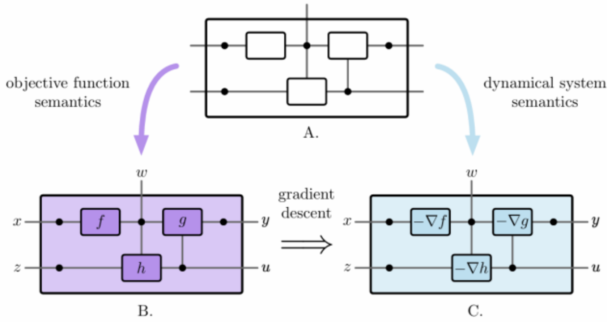
More sheaves

- Simplicial complexes
- Structures beyond graphs



More optimization

- Newton's Method
- Black boxing gradient descent
- Unified functorial gradient descent framework



Thanks for listening!