Unrolled Neural Networks for Constrained Optimization

Samar Hadou and Alejandro Ribeiro

University of Pennsylvania

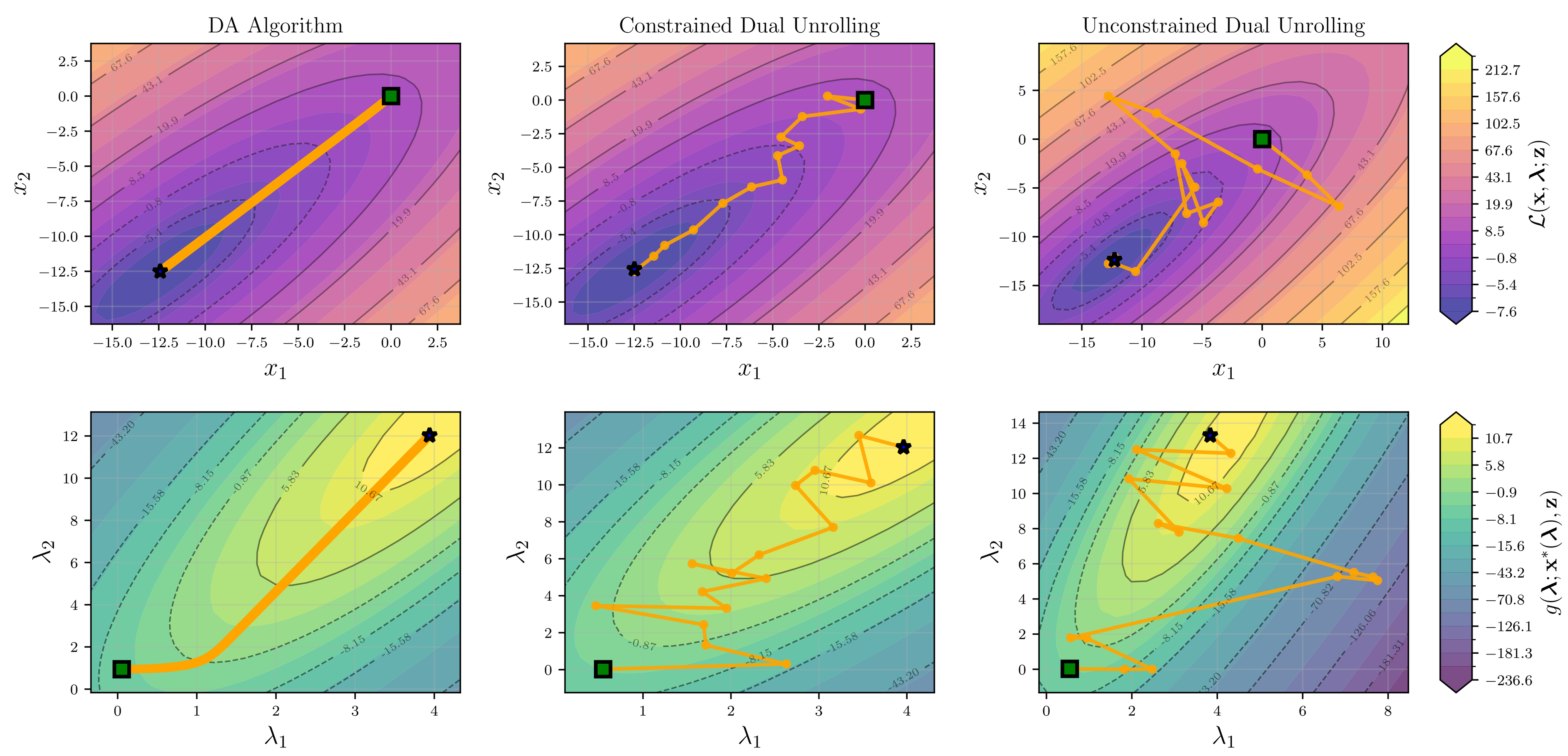
- **Goal**: Train interacting neural networks whose layers imitate dual ascent
- **Challenge**: When we train a neural network to imitate a descent algorithm, we expect trajectories like the middle,
  - $\Rightarrow$ but instead we observe the one on the right.
- **Our Solution**: We enforce primal descent and dual ascent during training
  - $\Rightarrow$ **Advantage**: Better robustness to distribution shifts.

- **Constrained optimization** is a family of problems that

$$P^*(\mathbf{z}) \;=\; \min_{\mathbf{x}\in\mathbb{R}^n} \; f_0(\mathbf{x};\mathbf{z}) \quad \text{s.t.} \quad \mathbf{f}(\mathbf{x};\mathbf{z}) \le \mathbf{0},$$

  - $\Rightarrow$ $\mathbf{z}$ is a problem instance.
- Define the dual problem as ($\boldsymbol{\lambda}$ is the dual variable):

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^m_+} \min_{\mathbf{x}} \; \mathcal{L}(\mathbf{x},\boldsymbol{\lambda};\mathbf{z}) \;:=\; f_0(\mathbf{x};\mathbf{z}) + \boldsymbol{\lambda}^\top \mathbf{f}(\mathbf{x};\mathbf{z}).$$



Constrained-Optimization Unrolling

- The DA algorithm finds the solution through iterations of two steps,

$$\mathbf{x}_l^*(\boldsymbol{\lambda}_l) \;\in\; \operatorname*{argmin}_{\mathbf{x}} \; \mathcal{L}(\mathbf{x},\boldsymbol{\lambda}_l;\mathbf{z}), \tag{P1}$$

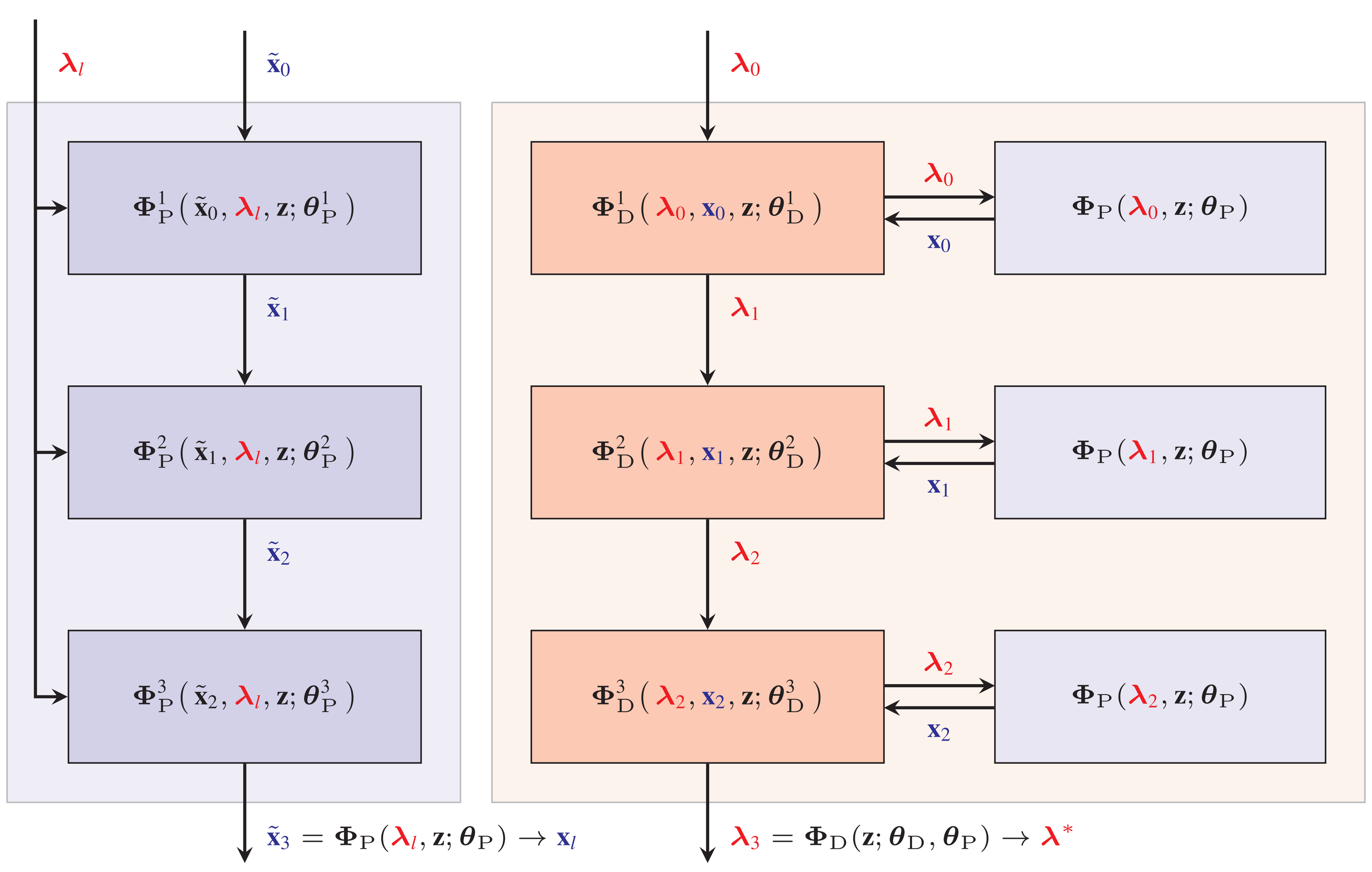
$$\boldsymbol{\lambda}_{l+1} \;=\; \Big[\boldsymbol{\lambda}_l + \eta\, \mathbf{f}\big(\mathbf{x}_l^*,\mathbf{z}\big)\Big]_+. \tag{D1}$$

- Our architecture consists of a **primal** $\Phi_{\mathrm{P}}(\cdot\,,\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}})$ and a **dual** $\Phi_{\mathrm{D}}(\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}},\boldsymbol{\theta}_{\mathrm{D}})$ network
- The primal network finds the stationary point of (P1) for a given $\boldsymbol{\lambda}$:

$$\tilde{\mathbf{x}}_k \;=\; \boldsymbol{\Phi}_{\mathrm{P}}^k\Big(\tilde{\mathbf{x}}_{k-1},\boldsymbol{\lambda}_l,\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}^k\Big). \tag{P2}$$

- Each dual layer returns a dual variable in response to the feasibility violation:

$$\boldsymbol{\lambda}_l \;=\; \boldsymbol{\Phi}_{\mathrm{D}}^l\Big(\boldsymbol{\lambda}_{l-1},\, \boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda}_{l-1},\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}),\, \mathbf{z};\, \boldsymbol{\theta}_{\mathrm{D}}^l\Big). \tag{D2}$$



Constrained Dual Unrolling

- We use an unsupervised loss that mimics the dual problem:

$$\boldsymbol{\theta}_{\mathrm{D}}^* \;\in\; \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathrm{D}}} \; \mathbb{E}_{\mathbf{z}}\Big[\mathcal{L}\big(\underbrace{\boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda}_L,\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}^*)}_{\mathbf{x}_L},\, \boldsymbol{\lambda}_L;\mathbf{z}\big)\Big], \tag{D-train}$$

$$\text{with} \quad \boldsymbol{\theta}_{\mathrm{P}}^* \in \operatorname*{argmin}_{\boldsymbol{\theta}_{\mathrm{P}}} \mathbb{E}_{\mathbf{z},\boldsymbol{\lambda}}\Big[\mathcal{L}\big(\boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda},\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}),\, \boldsymbol{\lambda};\mathbf{z}\big)\Big], \tag{P-train}$$

  where $\boldsymbol{\lambda}_L$ is the final output of the dual network.

- We enforce **descent and ascent constraints** on the primal and dual layers.
  - $\Rightarrow$ The primal training objective incorporates descent constraints that **decrease the gradient norm** of the Lagrangian across the layers:

$$\boldsymbol{\theta}_{\mathrm{P}}^* \;\in\; \operatorname*{argmin}_{\boldsymbol{\theta}_{\mathrm{P}}} \; \mathbb{E}_{\mathbf{z},\boldsymbol{\lambda}}\Big[\mathcal{L}\big(\boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda},\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}),\, \boldsymbol{\lambda};\mathbf{z}\big)\Big], \tag{P-constrained}$$

$$\text{s.t.} \quad \mathbb{E}_{\mathbf{z},\boldsymbol{\lambda}}\Big[\big\|\nabla_{\mathbf{x}}\mathcal{L}(\tilde{\mathbf{x}}_k,\boldsymbol{\lambda};\mathbf{z})\big\| - \alpha_k\big\|\nabla_{\mathbf{x}}\mathcal{L}(\tilde{\mathbf{x}}_{k-1},\boldsymbol{\lambda};\mathbf{z})\big\|\Big] \le 0, \;\; \forall k.$$

  - $\Rightarrow$ The dual training objective incorporates ascent constraints in the form of **decreasing the constraint violations** across the layers:

$$\boldsymbol{\theta}_{\mathrm{D}}^* \;\in\; \operatorname*{argmax}_{\boldsymbol{\theta}_{\mathrm{D}}} \; \mathbb{E}_{\mathbf{z}}\Big[\mathcal{L}\big(\underbrace{\boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda}_L,\mathbf{z};\boldsymbol{\theta}_{\mathrm{P}}^*)}_{\mathbf{x}_L},\, \boldsymbol{\lambda}_L;\mathbf{z}\big)\Big], \tag{D-constrained}$$

$$\text{s.t.} \quad \mathbb{E}_{\mathbf{z}}\Big[\big\|\mathbf{f}(\mathbf{x}_l;\mathbf{z})\big\| - \beta_l\big\|\mathbf{f}(\mathbf{x}_{l-1};\mathbf{z})\big\|\Big] \le 0, \;\; \forall l.$$

  - $\Rightarrow$ The scalars α_k and β_l are hyperparameters governing the step size.

- **Key issue**: The primal network needs to be trained on the multipliers that would be seen during the execution of the dual network.
- We alternate between training the primal and dual networks:
  - $\Rightarrow$ For each network, we construct the corresponding Lagrangian and perform a few training epochs through a primal-dual algorithm (cf. (P1)–(D1))

Numerical Results

- We consider mixed integer quadratic programs (MIQPs) with n variables, m linear constraints and r integer constraints.
  - $\Rightarrow$ We relax the integer constraints into linear *box* constraints:

$$\min_{\mathbf{x}} \; \frac{1}{2}\,\mathbf{x}^\top\mathbf{P}\mathbf{x} + \mathbf{q}^\top\mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \le \mathbf{b}.$$

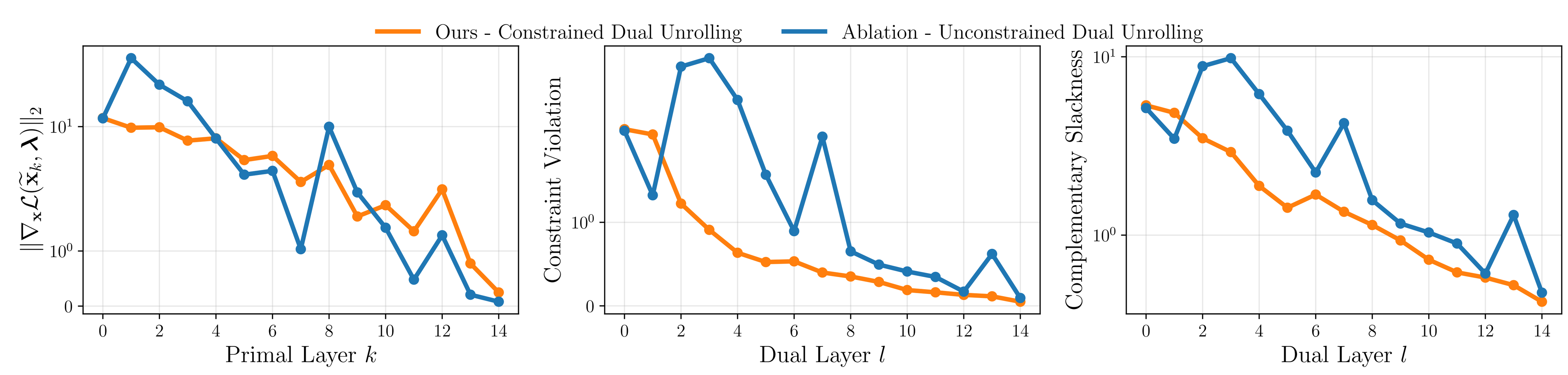
- We design GNN-based primal and dual networks:

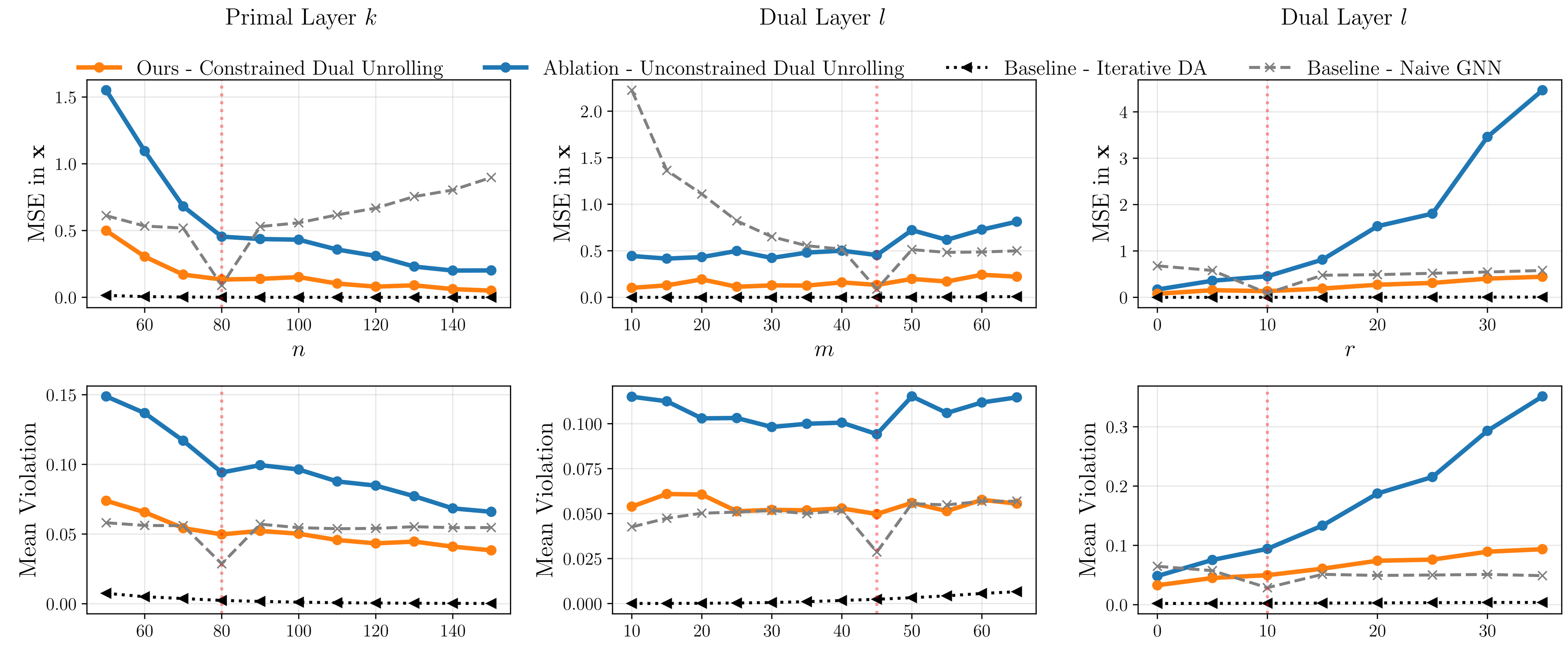
$$\mathbf{X}_\ell = \varphi\left(\sum_{h=0}^{K_{\mathrm{h}}} \mathbf{S}^h\mathbf{X}_{\ell-1}\boldsymbol{\Theta}_{\ell,h}\right), \quad \mathbf{S} \;=\; \begin{bmatrix}\mathbf{P} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0}\end{bmatrix},$$

  where K_{h} determines the neighborhood size.

- Under the constraints, we observe a consistent decrease in the Lagrangian gradient norm and in the mean constraint violations across the layers.

- **OOD Performance**: We vary one problem parameter while keeping the others fixed.
  - $\Rightarrow$ We consistently outperform the unconstrained model and naive GNN in optimality and feasibility across all OOD scenarios.
  - $\Rightarrow$ The gap widens as the distribution shift becomes more severe (i.e., $(m+2r)/n$ increases).





The red dotted line represents the in-distribution scenario.

selaraby@seas.upenn.edu