Unrolled graph neural networks for constrained optimization

Samar Hadon, Alejandro Ribeiro

2025-11-03

Outline

- Task: Constrained optimization with Dual Ascent(DA)
- Method: Unroll the dynamics of DA in two coupled GNNS
- Experiment: Mixed-integer quadratic program(MIQP)
- Conclusion: Learnable unsupervised method

Constrained optimization

- Consider a constrained problem that poses the task of minimizing a scalar objective function $f_0 : \mathbb{R}^n \to \mathbb{R}$ subject to m constraints, formulated as

$$P^*(\mathbf{z}) = \min_{\mathbf{x} \in \mathbb{R}^n} f_0(\mathbf{x}; \mathbf{z}) \quad \text{s.t.} \quad \mathbf{f}(\mathbf{x}; \mathbf{z}) \leq \mathbf{o}, \tag{1}$$

  where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is a vector-valued function representing the problem constraints, and $\mathbf{z}$ represents a problem instance.
- The Lagrangian function, $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m_+ \to \mathbb{R}$ is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}; \mathbf{z}) = f_0(\mathbf{x}; \mathbf{z}) + \boldsymbol{\lambda}^\top \mathbf{f}(\mathbf{x}; \mathbf{z}), \tag{2}$$

  where $\boldsymbol{\lambda}$ contains the dual multipliers.

Dual ascent(DA)

- The dual problem is defined as

$$D^*(\mathbf{z}) = \max_{\boldsymbol{\lambda}\in\mathbb{R}_+^m} \min_{\mathbf{x}} \; \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}; \mathbf{z}), \tag{3}$$

and the duality theory affirms that $D^*(\mathbf{z}) \leq P^*(\mathbf{z})$. Under this assumption, the Lagrangian has a saddle point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$, with $\mathbf{x}^*$ and $\boldsymbol{\lambda}^*$ optimal for (1) and (3), respectively–primal and dual domain.

  - The dual ascent (DA) algorithm retrieves the dual optimum $\boldsymbol{\lambda}^*$ through the iterations:

$$\mathbf{x}_l^*(\boldsymbol{\lambda}_l) \in \operatorname*{argmin}_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_l; \mathbf{z}), \tag{4}$$

$$\boldsymbol{\lambda}_{l+1} = \left[\boldsymbol{\lambda}_l + \eta\, \mathbf{f}(\mathbf{x}_l^*, \mathbf{z})\right]_+, \tag{5}$$

    where η is a step size, and the operator $[\cdot]_+$ denotes a projection onto $\mathbb{R}_+^m$.
  - Lagrangian stationary point is attained by $\mathbf{x}^* \in \mathbf{x}^*(\boldsymbol{\lambda}^*)$.

Unrolled networks for Constrained optimization-Primal

- The primal network, denoted by $\Phi_{\mathrm{P}}(\cdot, \cdot; \theta_{\mathrm{P}})$, predicts a K-step trajectory from an initial point $\widetilde{\mathbf{x}}_0$ towards $\widetilde{\mathbf{x}}_K \approx \mathbf{x}^*(\boldsymbol{\lambda})$ across its K unrolled layers—$\{\widetilde{\mathbf{x}}_0, \widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_K\}$.
- For a given dual multiplier $\boldsymbol{\lambda}$ and a problem instance $\mathbf{z}$, the k-th primal layer refines the estimate $\widetilde{\mathbf{x}}_{k-1}$ into

$$\widetilde{\mathbf{x}}_k = \Phi_{\mathrm{P}}^k \left(\widetilde{\mathbf{x}}_{k-1}, \boldsymbol{\lambda}, \mathbf{z}; \theta_{\mathrm{P}}^k \right), \tag{6}$$

where θ_{P}^k contains the parameters of the primal layer.

Unrolled networks for Constrained optimization-Dual

- The dual network, denoted by $\Phi_{\mathrm{D}}(\cdot;\theta_{\mathrm{D}},\theta_{\mathrm{P}})$, has L layers whose outputs constitute a trajectory starting from an initial point $\boldsymbol{\lambda}_0$ and ending at an estimate of the optimal multiplier, $\boldsymbol{\lambda}_L \approx \boldsymbol{\lambda}^*$— $\{\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_L\}$
- The l-th dual layer is defined as

$$\boldsymbol{\lambda}_l = \Phi_{\mathrm{D}}^{l}\left(\boldsymbol{\lambda}_{l-1}, \Phi_{\mathrm{P}}(\boldsymbol{\lambda}_{l-1}, \mathbf{z}; \theta_{\mathrm{P}}), \mathbf{z}; \theta_{\mathrm{D}}^{l}\right), \tag{7}$$

  where θ_{D}^{l} is the learnable parameters, the l-th dual layer queries the primal network for its estimate $\Phi_{\mathrm{P}}(\boldsymbol{\lambda}_{l-1}, \mathbf{z}; \theta_{\mathrm{P}}) \approx \mathbf{x}_{l-1}$.
- The nonlinearity at the end of each dual layer is chosen as a relu function to ensure that the predicted multipliers are nonnegative.

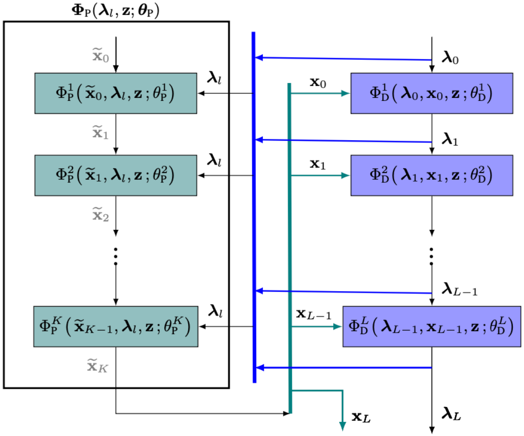
Unrolled networks for Constrained optimization

- Finally, the solution to (1) is obtained by feeding the final dual estimate, $\boldsymbol{\lambda}_L = \Phi_{\mathrm{D}}(\mathbf{z}; \theta_{\mathrm{D}}, \theta_{\mathrm{P}}^*)$, to the primal network,

$$\mathbf{x}_L = \Phi_{\mathrm{P}}\left(\Phi_{\mathrm{D}}(\mathbf{z}; \theta_{\mathrm{D}}, \theta_{\mathrm{P}}), \mathbf{z}; \theta_{\mathrm{P}}\right). \tag{8}$$

  The goal is to train the primal and dual networks such that the output of the primal network satisfies $\widetilde{\mathbf{x}}_K \approx \mathbf{x}^*(\boldsymbol{\lambda})$ for any $\boldsymbol{\lambda}$, and the output of the dual network satisfies $\boldsymbol{\lambda}_L \approx \boldsymbol{\lambda}^*$ for a family of optimization problems.

Framework



$\boldsymbol{\Phi}_{\mathrm{P}}(\boldsymbol{\lambda}_l, \mathbf{z}; \boldsymbol{\theta}_{\mathrm{P}})$
$\widetilde{\mathbf{x}}_0$
$\Phi_{\mathrm{P}}^1(\widetilde{\mathbf{x}}_0, \boldsymbol{\lambda}_l, \mathbf{z}; \theta_{\mathrm{P}}^1)$
$\boldsymbol{\lambda}_l$
$\widetilde{\mathbf{x}}_1$
$\Phi_{\mathrm{P}}^2(\widetilde{\mathbf{x}}_1, \boldsymbol{\lambda}_l, \mathbf{z}; \theta_{\mathrm{P}}^2)$
$\boldsymbol{\lambda}_l$
$\widetilde{\mathbf{x}}_2$
$\Phi_{\mathrm{P}}^K(\widetilde{\mathbf{x}}_{K-1}, \boldsymbol{\lambda}_l, \mathbf{z}; \theta_{\mathrm{P}}^K)$
$\boldsymbol{\lambda}_l$
$\widetilde{\mathbf{x}}_K$
$\boldsymbol{\lambda}_0$
$\mathbf{x}_0$
$\Phi_{\mathrm{D}}^1(\boldsymbol{\lambda}_0, \mathbf{x}_0, \mathbf{z}; \theta_{\mathrm{D}}^1)$
$\boldsymbol{\lambda}_1$
$\mathbf{x}_1$
$\Phi_{\mathrm{D}}^2(\boldsymbol{\lambda}_1, \mathbf{x}_1, \mathbf{z}; \theta_{\mathrm{D}}^2)$
$\boldsymbol{\lambda}_{L-1}$
$\mathbf{x}_{L-1}$
$\Phi_{\mathrm{D}}^L(\boldsymbol{\lambda}_{L-1}, \mathbf{x}_{L-1}, \mathbf{z}; \theta_{\mathrm{D}}^L)$
$\mathbf{x}_L$
$\boldsymbol{\lambda}_L$

Objective function-Primal-Dual

- The nested training problem is defined as

$$\theta_{\mathrm{D}}^* \in \underset{\theta_{\mathrm{D}}}{\operatorname{argmax}}\, \mathbb{E}_{\mathbf{z}} \left[\mathcal{L}\left(\Phi_{\mathrm{P}}(\boldsymbol{\lambda}_L, \mathbf{z}; \theta_{\mathrm{P}}^*), \boldsymbol{\lambda}_L; \mathbf{z}\right)\right], \tag{9}$$

$$\text{with} \quad \theta_{\mathrm{P}}^* \in \underset{\theta_{\mathrm{P}}}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{\lambda},\mathbf{z}} \left[\mathcal{L}\left(\Phi_{\mathrm{P}}(\boldsymbol{\lambda}, \mathbf{z}; \theta_{\mathrm{P}}), \boldsymbol{\lambda}; \mathbf{z}\right)\right], \tag{10}$$

  where $\boldsymbol{\lambda}_L = \Phi_{\mathrm{D}}(\mathbf{z}; \theta_{\mathrm{D}}, \theta_{\mathrm{P}}^*)$ is the final output of the dual network.
- This does not guarantee that the intermediate layer trajectories are monotonically descending (primal) or ascending (dual).

Objective function-Primal-descent constrains

- Primal objective function:

$$\theta_{\mathrm{P}}^{*} \in \underset{\theta_{\mathrm{P}}}{\operatorname{argmin}} \, \mathbb{E}\left[\mathcal{L}\left(\Phi_{\mathrm{P}}(\boldsymbol{\lambda}, \mathbf{z}; \theta_{\mathrm{P}}), \boldsymbol{\lambda}; \mathbf{z}\right)\right], \tag{11}$$

$$\text{s.t. } \mathbb{E}\left[\left\|\nabla_{\mathbf{x}} \mathcal{L}(\widetilde{\mathbf{x}}_{k}, \boldsymbol{\lambda}; \mathbf{z})\right\| - \alpha_{k} \left\|\nabla_{\mathbf{x}} \mathcal{L}(\widetilde{\mathbf{x}}_{k-1}, \boldsymbol{\lambda}; \mathbf{z})\right\|\right] \leq 0, \forall k \tag{12}$$

  where α_k is a design parameter that controls the descent rate, $\widetilde{\mathbf{x}}_0$ is initialized randomly.

- The gradient norm of Lagrangian with respect to $\boldsymbol{X}$ is forced to decrease (Regularization term).

Primal Networks

Algorithm 1 Primal Network Training

Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\mu}, \epsilon_{\mathrm{P}}, \eta_{\mathrm{P}}$
for each epoch **do**
  for each primal batch **do**
    Sample $\{\mathbf{z}_{(j)}\}_{j=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
    Sample $\{\boldsymbol{\lambda}_{(i,j)}\}_{i=1,j=1}^{M,N}$ from the trajectories by $\boldsymbol{\theta}_{\mathrm{D}}$
    Execute the primal network to generate $\{\widetilde{\mathbf{x}}_{k,(i,j)}\}_{k,i,j}$
    $\ell(\boldsymbol{\theta}_{\mathrm{P}}) \leftarrow \widehat{\mathcal{L}}(\widetilde{\mathbf{x}}_K, \boldsymbol{\lambda}; \mathbf{z})$
    $\mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \leftarrow \mu_k \left(\|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_k, \boldsymbol{\lambda}; \mathbf{z})\| - \alpha_k \|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_{k-1}, \boldsymbol{\lambda}; \mathbf{z})\| \right)$
    $\boldsymbol{\theta}_{\mathrm{P}} \leftarrow \boldsymbol{\theta}_{\mathrm{P}} - \epsilon_{\mathrm{P}} \cdot \left(\nabla \ell(\boldsymbol{\theta}_{\mathrm{P}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{P}}} \sum_k \mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \right)$
    $\boldsymbol{\mu} \leftarrow \left[\boldsymbol{\mu} + \eta_{\mathrm{P}} \cdot \nabla_{\boldsymbol{\mu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \right]_{+}$
  end for
end for
return $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}$

Objective function-Dual

- Dual objective function:

$$\theta_{\mathrm{D}}^{*} \in \underset{\theta_{\mathrm{D}}}{\operatorname{argmax}}\, \mathbb{E}\left[\mathcal{L}\left(\Phi_{\mathrm{P}}(\boldsymbol{\lambda}_L, \mathbf{z}; \theta_{\mathrm{P}}^{*}), \boldsymbol{\lambda}_L; \mathbf{z}\right)\right], \tag{13}$$

$$\text{s.t. } \mathbb{E}\left[\|\|\mathbf{f}(\mathbf{x}_l; \mathbf{z})\| - \beta_l \|\mathbf{f}(\mathbf{x}_{l-1}; \mathbf{z})\|\|\right] \leq 0, \forall l \tag{14}$$

where β_l is a design parameter, $\boldsymbol{\lambda}_0$ is randomly initialized.
- The gradient norm of Lagrangian with respect to $\boldsymbol{\lambda}$ is forced to decrease (Regularization term).

Dual Networks

Algorithm 2 Dual Network Training

```
Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}, \epsilon_{\mathrm{D}}, \eta_{\mathrm{D}}$
for each epoch do
    for each dual batch do
        Sample $\{\mathbf{z}_{(i)}\}_{i=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
        Execute the networks to generate $\{(\mathbf{x}_{l,(i)}, \boldsymbol{\lambda}_{l,(i)})\}_{l,i}$
        $\ell(\boldsymbol{\theta}_{\mathrm{D}}) \leftarrow -\widehat{\mathcal{L}}(\mathbf{x}_L, \boldsymbol{\lambda}_L; \mathbf{z})$
        $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \leftarrow \sum_l \nu_l \cdot \widehat{\mathbb{E}}\Big[\big\|\mathbf{f}(\mathbf{x}_l; \mathbf{z}\big\| - \beta_l \big\|\mathbf{f}(\mathbf{x}_{l-1}; \mathbf{z})\big\|\Big]$
        $\boldsymbol{\theta}_{\mathrm{D}} \leftarrow \boldsymbol{\theta}_{\mathrm{D}} - \epsilon_{\mathrm{D}} \cdot \Big(\nabla \ell(\boldsymbol{\theta}_{\mathrm{D}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{D}}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu})\Big)$
        $\boldsymbol{\nu} \leftarrow \big[\boldsymbol{\nu} + \eta_{\mathrm{D}} \cdot \nabla_{\boldsymbol{\nu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu})\big]_+$
    end for
end for
return $\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}$
```

Objective function

Algorithm 1 Primal Network Training

Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\mu}, \epsilon_{\mathrm{P}}, \eta_{\mathrm{P}}$
for each epoch **do**
  for each primal batch **do**
    Sample $\{\mathbf{z}_{(j)}\}_{j=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
    Sample $\{\boldsymbol{\lambda}_{(i,j)}\}_{i=1,j=1}^{M,N}$ from the trajectories by $\boldsymbol{\theta}_{\mathrm{D}}$
    Execute the primal network to generate $\{\widetilde{\mathbf{x}}_{k,(i,j)}\}_{k,i,j}$
    $\ell(\boldsymbol{\theta}_{\mathrm{P}}) \leftarrow \widehat{\mathcal{L}}(\widetilde{\mathbf{x}}_K, \boldsymbol{\lambda}; \mathbf{z})$
    $\mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \leftarrow \mu_k \Big(\|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_k, \boldsymbol{\lambda}; \mathbf{z})\| - \alpha_k \|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_{k-1}, \boldsymbol{\lambda}; \mathbf{z})\| \Big)$
    $\boldsymbol{\theta}_{\mathrm{P}} \leftarrow \boldsymbol{\theta}_{\mathrm{P}} - \epsilon_{\mathrm{P}} \cdot \Big(\nabla \ell(\boldsymbol{\theta}_{\mathrm{P}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{P}}} \sum_k \mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \Big)$
    $\boldsymbol{\mu} \leftarrow \big[\boldsymbol{\mu} + \eta_{\mathrm{P}} \cdot \nabla_{\boldsymbol{\mu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \big]_+$
  end for
end for
return $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}$

Algorithm 2 Dual Network Training

Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}, \epsilon_{\mathrm{D}}, \eta_{\mathrm{D}}$
for each epoch **do**
  for each dual batch **do**
    Sample $\{\mathbf{z}_{(i)}\}_{i=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
    Execute the networks to generate $\{(\mathbf{x}_{l,(i)}, \boldsymbol{\lambda}_{l,(i)})\}_{l,i}$
    $\ell(\boldsymbol{\theta}_{\mathrm{D}}) \leftarrow -\widehat{\mathcal{L}}(\mathbf{x}_L, \boldsymbol{\lambda}_L; \mathbf{z})$
    $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \leftarrow \sum_l \nu_l \cdot \widehat{\mathbb{E}} \Big[\|\mathbf{f}(\mathbf{x}_l; \mathbf{z}\| - \beta_l \|\mathbf{f}(\mathbf{x}_{l-1}; \mathbf{z})\| \Big]$
    $\boldsymbol{\theta}_{\mathrm{D}} \leftarrow \boldsymbol{\theta}_{\mathrm{D}} - \epsilon_{\mathrm{D}} \cdot \Big(\nabla \ell(\boldsymbol{\theta}_{\mathrm{D}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{D}}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \Big)$
    $\boldsymbol{\nu} \leftarrow \big[\boldsymbol{\nu} + \eta_{\mathrm{D}} \cdot \nabla_{\boldsymbol{\nu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \big]_+$
  end for
end for
return $\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}$

Experiment-numerical results

- Mixed-integer quadratic program(MIQP) with linear inequality constraints can be formulated as:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^\top \mathbf{P}\mathbf{x} + \mathbf{q}^\top \mathbf{x} \tag{15}$$

$$\text{s.t.} \quad \bar{\mathbf{A}}\mathbf{x} \leq \bar{\mathbf{b}}, \tag{16}$$

$$x_i \in \{-1, 1\}, \ \forall i \in \mathcal{I}, \tag{17}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{q} \in \mathbb{R}^n$, $\bar{\mathbf{A}} \in \mathbb{R}^{m \times n}$, $\bar{\mathbf{b}} \in \mathbb{R}^m$, $\mathbf{P} \in \mathbb{R}^{n \times n}$ is PSD, and $\mathcal{I}$ is a set that contains the indices of the binary variables with cardinality $|\mathcal{I}| = r \leq n$;

Experiment-numerical results

- We consider a convex relaxation of (17) in the form of box constraints, i.e., $-1 \leq x_i \leq 1,\ \forall i \in \mathcal{I}$. The relaxed MIQP problem:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^\top \mathbf{P}\mathbf{x} + \mathbf{q}^\top \mathbf{x} \quad \text{s.t} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{18}$$

  where $\mathbf{A} \in \mathbb{R}^{(m+2r)\times n}$, $\mathbf{b} \in \mathbb{R}^{m+2r}$, $\mathbf{A} = \left[\bar{\mathbf{A}}; \mathbf{M}; -\mathbf{M}\right]$ and $\mathbf{b} = \left[\bar{\mathbf{b}}; \mathbf{1}_r; \mathbf{1}_r\right]$.
- $\mathbf{M} \in \{0,1\}^{r\times n}$ is a selection matrix whose j-th row is the standard basis vector $\mathbf{e}_{i_j}^\top$ for $i_j \in \mathcal{I}$, and $\mathbf{1}_r$ is an r-dimensional all-ones vector.
- The Lagrangian function is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^\top \mathbf{P}\mathbf{x} + \mathbf{q}^\top \mathbf{x} + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}). \tag{19}$$

Graph Neural Networks

- Graph adjacency:

$$\mathbf{S} = \begin{bmatrix} \mathbf{P} & \mathbf{A}^\top \\ \mathbf{A} & \mathbf{0} \end{bmatrix}. \tag{20}$$

- The ℓ-th unrolled layer consists of a cascade of T graph convolutional sub-layers. The t-th sub-layer filters:

$$\mathbf{X}_t^{(\ell)} = \varphi\left(\sum_{h=0}^{K_\mathrm{h}} \mathbf{S}^h \mathbf{X}_{t-1}^{(\ell)} \Theta_{t,h}^{(\ell)}\right), \tag{21}$$

where $\Theta_{t,h}^{(\ell)} \in \mathbb{R}^{F_{t-1} \times F_t}$ is the set of learnable parameters, K_h represents the filter taps, and φ is a nonlinear activation function.

Primal forward process

- In the primal network, the input to the kth unrolled layer is

$$\widetilde{\mathbf{X}}_0^{(k)} = \begin{bmatrix} \widetilde{\mathbf{x}}_{k-1} & \mathbf{q} \\ \boldsymbol{\lambda} & \mathbf{b} \end{bmatrix}, \tag{22}$$

  where $\widetilde{\mathbf{x}}_{k-1}$ is the output of the previous unrolled layer, and $\boldsymbol{\lambda}$, $\mathbf{q}$ and $\mathbf{b}$ are input data.
- The output of the unrolled layer is then

$$\widetilde{\mathbf{x}}_k = \widetilde{\mathbf{x}}_{k-1} + \mathbf{M}_{\mathrm{P}} \widetilde{\mathbf{X}}_T^{(k)} \mathbf{W}_k + \mathbf{c}_k, \tag{23}$$

  where $\widetilde{\mathbf{X}}_T^{(k)}$ is the output of the T-th graph sub-layer, and $\mathbf{W}_k \in \mathbb{R}^{F_T}$ and $\mathbf{c}_k \in \mathbb{R}^n$ are the parameters of the readout layer. The selection matrix $\mathbf{M}_{\mathrm{P}}$ extracts the outputs associated with the n variable nodes.

Dual forward process

- The input to each unrolled dual layer is constructed as

$$\mathbf{X}_0^{(l)} = \begin{bmatrix} \mathbf{x}_{l-1} & \mathbf{q} \\ \boldsymbol{\lambda}_{l-1} & \mathbf{b} \end{bmatrix}, \tag{24}$$

  where $\boldsymbol{\lambda}_{l-1}$ is the previous dual estimate and $\mathbf{x}_{l-1}$ is the corresponding estimate of the primal network.

- The output of the unrolled layer is expressed as

$$\boldsymbol{\lambda}_l = \varphi_{\text{relu}}\left(\mathbf{y}_{l-1} + \mathbf{M}_{\text{D}}\mathbf{X}_T^{(l)}\mathbf{W}_l + \mathbf{c}_l\right), \tag{25}$$

  where $\mathbf{M}_{\text{D}}$ selects the constraint-node values, and $\mathbf{W}_l \in \mathbb{R}^{F_T}$ and $\mathbf{c}_l \in \mathbb{R}^{m+2r}$ are learnable parameters—distinct from those of the primal layers despite the shared notation.

Objective function-Primal

Algorithm 1 Primal Network Training

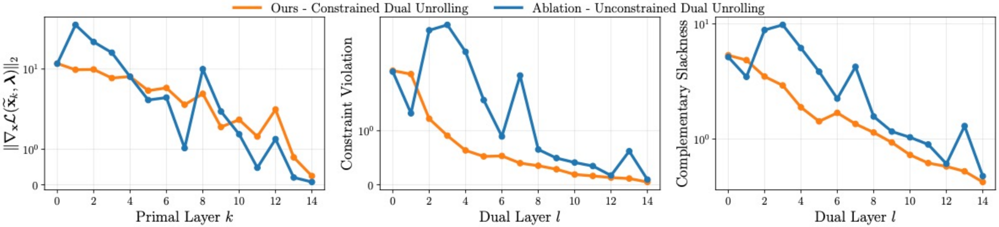
Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\mu}, \epsilon_{\mathrm{P}}, \eta_{\mathrm{P}}$
for each epoch **do**
  for each primal batch **do**
    Sample $\{\mathbf{z}_{(j)}\}_{j=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
    Sample $\{\boldsymbol{\lambda}_{(i,j)}\}_{i=1,j=1}^{M,N}$ from the trajectories by $\boldsymbol{\theta}_{\mathrm{D}}$
    Execute the primal network to generate $\{\widetilde{\mathbf{x}}_{k,(i,j)}\}_{k,i,j}$
    $\ell(\boldsymbol{\theta}_{\mathrm{P}}) \leftarrow \widehat{\mathcal{L}}(\widetilde{\mathbf{x}}_K, \boldsymbol{\lambda}; \mathbf{z})$
    $\mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \leftarrow \mu_k \left(\|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_k, \boldsymbol{\lambda}; \mathbf{z})\| - \alpha_k \|\widehat{\nabla} \mathcal{L}(\widetilde{\mathbf{x}}_{k-1}, \boldsymbol{\lambda}; \mathbf{z})\| \right)$
    $\boldsymbol{\theta}_{\mathrm{P}} \leftarrow \boldsymbol{\theta}_{\mathrm{P}} - \epsilon_{\mathrm{P}} \cdot \left(\nabla \ell(\boldsymbol{\theta}_{\mathrm{P}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{P}}} \sum_k \mathcal{C}_k(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \right)$
    $\boldsymbol{\mu} \leftarrow \left[\boldsymbol{\mu} + \eta_{\mathrm{P}} \cdot \nabla_{\boldsymbol{\mu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}) \right]_+$
  end for
end for
return $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\mu}$

Algorithm 2 Dual Network Training

Inputs: $\boldsymbol{\theta}_{\mathrm{P}}, \boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}, \epsilon_{\mathrm{D}}, \eta_{\mathrm{D}}$
for each epoch **do**
  for each dual batch **do**
    Sample $\{\mathbf{z}_{(i)}\}_{i=1}^{N} \sim \mathcal{D}_{\mathbf{z}}$
    Execute the networks to generate $\{(\mathbf{x}_{l,(i)}, \boldsymbol{\lambda}_{l,(i)})\}_{l,i}$
    $\ell(\boldsymbol{\theta}_{\mathrm{D}}) \leftarrow -\widehat{\mathcal{L}}(\mathbf{x}_L, \boldsymbol{\lambda}_L; \mathbf{z})$
    $\mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \leftarrow \sum_l \nu_l \cdot \widehat{\mathbb{E}} \left[\|\mathbf{f}(\mathbf{x}_l; \mathbf{z}\| - \beta_l \|\mathbf{f}(\mathbf{x}_{l-1}; \mathbf{z})\| \right]$
    $\boldsymbol{\theta}_{\mathrm{D}} \leftarrow \boldsymbol{\theta}_{\mathrm{D}} - \epsilon_{\mathrm{D}} \cdot \left(\nabla \ell(\boldsymbol{\theta}_{\mathrm{D}}) + \nabla_{\boldsymbol{\theta}_{\mathrm{D}}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \right)$
    $\boldsymbol{\nu} \leftarrow \left[\boldsymbol{\nu} + \eta_{\mathrm{D}} \cdot \nabla_{\boldsymbol{\nu}} \mathcal{C}(\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}) \right]_+$
  end for
end for
return $\boldsymbol{\theta}_{\mathrm{D}}, \boldsymbol{\nu}$

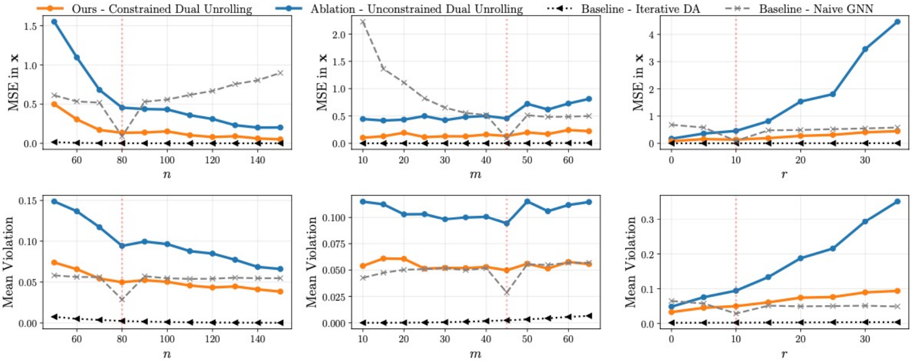
$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{2} \mathbf{x}^\top \mathbf{P} \mathbf{x} + \mathbf{q}^\top \mathbf{x} + \boldsymbol{\lambda}^\top (\mathbf{A} \mathbf{x} - \mathbf{b}). \tag{26}$$

Result



- Metrics:
  - Gradient norm of the Lagrangian
  - Constraint violation
  - The complementary slackness: $\boldsymbol{\lambda}_L^T f(\boldsymbol{x}_l)$
- The constrained model exhibits a consistent decrease in all three metrics across layers.

Result



- The number od optimization variables n, the number of linear constrains m, the number of integer-valued variables r; (Varying one problem parameter while keeping the others fixed).
- The constrained unrolling outperforms the other learning-based methods across all OOD scenarios.

Conlusion

- Replace the iterative DA with unrolled GNNs framework;
- Unsupervised learning method;
- Mixed-integer quadratic program(MIQP)， with good generalization ability;