Optimization over Trained Neural Networks: Difference-of-Convex Algorithm and Application to Data Center Scheduling

Xinwei Liu, Vladmir Dvorkin

Department of Electrical Engineering and Computer Science

University of Michigan – Ann Arbor

INFORMS 2025, Atlanta, GA

October 24, 2025

Overview

1. **Motivation and Background**
2. **Mathematical Formulation**
3. **Application**

Optimization over trained neural networks

$$\begin{aligned} \underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}} \quad & c(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) \leqslant \mathbf{0} \end{aligned}$$

- Usually, we know all elements $(c, g, \mathcal{X})$ of an optimization problem.
- **What should we do if the constraint function g is unknown?**
- In practice, we can approximate the function g from the data.

Optimization over trained neural networks

$$\begin{aligned} \underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}} \quad & c(\mathbf{x}) \\ \text{subject to} \quad & g(\mathbf{x}) \leqslant \mathbf{0} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}} \quad & c(\mathbf{x}) \\ \text{subject to} \quad & \mathbb{NN}(\mathbf{x}) \leqslant \mathbf{0} \end{aligned}$$

- Usually, we know all elements $(c, g, \mathcal{X})$ of an optimization problem.
- **What should we do if the constraint function g is unknown?**
- In practice, we can approximate the function g from the data.

- **Consider an approximation by a neural network** $\mathbb{NN} : \mathcal{X} \mapsto \mathbb{R}$
- The problem becomes an optimization over trained neural network.

Embedding trained neural networks into optimization

A neural network mapping $\mathbf{d}$ to $\boldsymbol{\lambda}$ is mathematically represented as

$$\boldsymbol{\lambda} = \mathbf{W}_{N+1}\mathrm{ReLU}(\dots \mathrm{ReLU}(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1)\dots) + \mathbf{b}_{N+1}$$

with fixed weights and biases $\mathbf{W}$ and $\mathbf{b}$, and variable input $\mathbf{d}$ and output $\boldsymbol{\lambda}$

[Turner et al., 2024]

Embedding trained neural networks into optimization

A neural network mapping $\mathbf{d}$ to $\boldsymbol{\lambda}$ is mathematically represented as

$$\boldsymbol{\lambda} = \mathbf{W}_{N+1}\text{ReLU}(\dots\text{ReLU}(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1)\dots) + \mathbf{b}_{N+1}$$

with fixed weights and biases $\mathbf{W}$ and $\mathbf{b}$, and variable input $\mathbf{d}$ and output $\boldsymbol{\lambda}$

We can write the ReLU activation function using complementarity logic:

$$y = \text{ReLU}(a) \quad \Longrightarrow \quad \begin{aligned} &y = a + v \\ &0 \leqslant y \perp v \geqslant 0 \end{aligned}$$

a input variable
y output variable
v auxiliary variable
$\perp$ complementarity

[Turner et al., 2024]

Embedding trained neural networks into optimization

A neural network mapping $\mathbf{d}$ to $\boldsymbol{\lambda}$ is mathematically represented as

$$\boldsymbol{\lambda} = \mathbf{W}_{N+1}\mathrm{ReLU}(\dots \mathrm{ReLU}(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1)\dots) + \mathbf{b}_{N+1}$$

with fixed weights and biases $\mathbf{W}$ and $\mathbf{b}$, and variable input $\mathbf{d}$ and output $\boldsymbol{\lambda}$

We can write the ReLU activation function using complementarity logic:

$$y = \mathrm{ReLU}(a) \quad \Longrightarrow \quad \begin{aligned} & y = a + v \\ & 0 \leqslant y \perp v \geqslant 0 \end{aligned}$$

a input variable
y output variable
v auxiliary variable
$\perp$ complementarity

Extremely difficult problem to solve in practice due to large computational burden

[Turner et al., 2024]

State of the art approaches on complementarity constraint

State of the art approaches

- Mix-integer constraint[Tjeng et al., 2017, Grimstad and Andersson, 2019]
- Big-M reformulation of ReLU–numerical issue
  [Tjeng et al., 2017, Grimstad and Andersson, 2019]
- SOS1 reformulation with decision trees[Turner et al., 2024]
- Semidefinite relaxation of ReLU [Dathathri et al., 2020, Fazlyab et al., 2020]
- Bound propagation [Wang et al., 2021]

Contribution

Our contribution

- Use difference of convex approach to optimize over trained neural network.
- Relax the ReLU constraints by penalizing them in the objective function.
- Avoid the computational complexity of standard MIP formulations.
- Formulate an algorithm to compute the hyperparameter to achieve fast convergence.
- Demonstrate in the data center load allocation example.

Embedding trained neural networks into optimization

$$
\begin{aligned}
\underset{\mathbf{d}\in\mathcal{D},\mathbf{y},\boldsymbol{\lambda}}{\text{minimize}}\ & c(\boldsymbol{\lambda}) \quad \text{convex function} \\
\text{subject to}\ & \mathbf{y}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1), \\
& \mathbf{y}_i = \text{ReLU}(\mathbf{W}_i\mathbf{y}_{i-1} + \mathbf{b}_i), \\
& \forall i = 2, \ldots, N \\
& \boldsymbol{\lambda} = \mathbf{W}_{N+1}\mathbf{y}_N + \mathbf{b}_{N+1},
\end{aligned}
$$

Embedding trained neural networks into optimization

$$
\begin{aligned}
\underset{\mathbf{d}\in\mathcal{D},\mathbf{y},\boldsymbol{\lambda}}{\text{minimize}}\ & c(\boldsymbol{\lambda}) \quad \text{convex function} \\
\text{subject to}\ & \mathbf{y}_1 = \mathrm{ReLU}(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1), \\
& \mathbf{y}_i = \mathrm{ReLU}(\mathbf{W}_i\mathbf{y}_{i-1} + \mathbf{b}_i), \\
& \forall i = 2, \ldots, N \\
& \boldsymbol{\lambda} = \mathbf{W}_{N+1}\mathbf{y}_N + \mathbf{b}_{N+1},
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
\underset{\mathbf{d}\in\mathcal{D},\mathbf{y},\mathbf{v},\boldsymbol{\lambda}}{\text{minimize}}\ & c(\boldsymbol{\lambda}) \\
\text{subject to}\ & \mathbf{y}_1 = \mathbf{W}_1\mathbf{d} + \mathbf{b}_1 + \mathbf{v}_1, \\
& \mathbf{y}_i = \mathbf{W}_i\mathbf{y}_{i-1} + \mathbf{b}_i + \mathbf{v}_i, \\
& \forall i = 2, \ldots, N, \\
& \mathbf{0} \leqslant \mathbf{y}_i \perp \mathbf{v}_i \geqslant \mathbf{0}, \\
& \forall i = 1, \ldots, N, \\
& \boldsymbol{\lambda} = \mathbf{W}_{N+1}\mathbf{y}_N + \mathbf{b}_{N+1},
\end{aligned}
$$

Complementarity Condition Computation Bottleneck

- **Problem:** NP hard.
- **Solution:** Introduce the complementarity condition as a bilinear penalty term in the objective function.

$$
\begin{aligned}
\underset{(\mathbf{d},\mathbf{y},\mathbf{v},\boldsymbol{\lambda})\in\mathcal{O}}{\text{minimize}} \quad & c(\boldsymbol{\lambda}) + \rho \sum_{i=1}^{N} \mathbf{y}_i^\top \mathbf{v}_i \\
\text{subject to} \quad & \mathbf{y}_1 = \mathbf{W}_1 \mathbf{d} + \mathbf{b}_1 + \mathbf{v}_1, \\
& \mathbf{y}_i = \mathbf{W}_i \mathbf{y}_{i-1} + \mathbf{b}_i + \mathbf{v}_i, \ \forall i = 2, \ldots, N \\
& \boldsymbol{\lambda} = \mathbf{W}_{N+1} \mathbf{y}_N + \mathbf{b}_{N+1},
\end{aligned}
$$

- Non-negative condition on $\mathbf{y}_i, \mathbf{v}_i \geqslant \mathbf{0}, \quad \forall i = 1, ..., N$ is included in set $\mathcal{O}$
- **Problem:** Non-convex problem.
- **Solution:** Reformulate the objective function into the difference of two convex functions and solve iteratively using the Difference of Convex Approach!

Difference of Convex Approach

$$c(\boldsymbol{\lambda}) + \rho \sum_{i=1}^{N} \mathbf{y}_i^\top \mathbf{v}_i \quad \Longrightarrow \quad \underbrace{c(\boldsymbol{\lambda}) + \frac{\rho}{4} \sum_{i=1}^{N} \|\mathbf{y}_i + \mathbf{v}_i\|_2^2}_{f_1(\mathbf{y},\mathbf{v}),\text{convex function}} - \underbrace{\frac{\rho}{4} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{v}_i\|_2^2}_{f_2(\mathbf{y},\mathbf{v}),\text{convex function}}$$

[Jara-Moroni et al., 2018]

Difference of Convex Approach

$$c(\boldsymbol{\lambda}) + \rho \sum_{i=1}^{N} \mathbf{y}_i^\top \mathbf{v}_i \quad \Longrightarrow \quad \underbrace{c(\boldsymbol{\lambda}) + \frac{\rho}{4} \sum_{i=1}^{N} \|\mathbf{y}_i + \mathbf{v}_i\|_2^2}_{f_1(\mathbf{y},\mathbf{v}),\text{convex function}} - \underbrace{\frac{\rho}{4} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{v}_i\|_2^2}_{f_2(\mathbf{y},\mathbf{v}),\text{convex function}}$$

Algorithm 1 DCA for solving the bilinear problem

input: feasible guess $\mathbf{d}^0, \mathbf{y}^0, \mathbf{v}^0, \boldsymbol{\lambda}^0$, tolerance $\varepsilon_{\text{tol}} > 0$
output: optimized NN input $\mathbf{d}^\star$
repeat
  set $k \leftarrow k + 1$.
  get $\mathbf{d}^{k+1}, \mathbf{y}^{k+1}, \mathbf{v}^{k+1}, \boldsymbol{\lambda}^{k+1}$ by solving DCA subproblem
until $f(\mathbf{y}^k, \mathbf{v}^k) - f(\mathbf{y}^{k+1}, \mathbf{v}^{k+1}) \leq \varepsilon_{\text{tol}}$
return $\mathbf{d}^\star \leftarrow \mathbf{d}^{k+1}$

$f(\mathbf{y}^k, \mathbf{v}^k) = f_1(\mathbf{y}^k, \mathbf{v}^k) - f_2(\mathbf{y}^k, \mathbf{v}^k)$. Here, we write $\mathbf{d}, \mathbf{y}, \mathbf{v}, \boldsymbol{\lambda}$ to represent $\mathbf{d}_i, \mathbf{y}_i, \mathbf{v}_i, \boldsymbol{\lambda}_i \forall i = 1, ..., N$

[Jara-Moroni et al., 2018]

Difference of Convex Approach Solution

$$\underbrace{-\frac{\rho}{4}\sum_{i=1}^{N}\|\mathbf{y}_i-\mathbf{v}_i\|_2^2}_{-f_2(\mathbf{y},\mathbf{v}),\text{concave function}} \Longrightarrow \underbrace{\frac{-\rho}{4}\sum_{i=1}^{N}(\mathbf{y}_i^k-\mathbf{v}_i^k)^\top(\mathbf{y}_i-\mathbf{v}_i)}_{-f_2(\mathbf{y},\mathbf{v})\ \text{Linear overestimation of the concave part}}$$

[Jara-Moroni et al., 2018]

Difference of Convex Approach Solution

$$\underbrace{-\frac{\rho}{4}\sum_{i=1}^{N}\|\mathbf{y}_i-\mathbf{v}_i\|_2^2}_{-f_2(\mathbf{y},\mathbf{v}),\text{concave function}} \implies \underbrace{\frac{-\rho}{4}\sum_{i=1}^{N}(\mathbf{y}_i^k-\mathbf{v}_i^k)^\top(\mathbf{y}_i-\mathbf{v}_i)}_{-f_2(\mathbf{y},\mathbf{v})\ \text{Linear overestimation of the concave part}}$$

DCA subproblem

$$\begin{aligned}\underset{(\mathbf{d},\mathbf{y},\mathbf{v},\boldsymbol{\lambda})\in\mathcal{O}}{\text{minimize}}\quad & c(\boldsymbol{\lambda})+\frac{\rho}{4}\sum_{i=1}^{N}\|\mathbf{y}_i+\mathbf{v}_i\|_2^2 \\ & -\frac{\rho}{2}\sum_{i=1}^{N}(\mathbf{y}_i^k-\mathbf{v}_i^k)^\top(\mathbf{y}_i-\mathbf{v}_i) \qquad \textit{Linear overestimation of the concave part}\\ \text{subject to}\quad & \mathbf{y}_1=\mathbf{W}_1\mathbf{d}+\mathbf{b}_1+\mathbf{v}_1,\\ & \mathbf{y}_i=\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i+\mathbf{v}_i,\ \forall i=2,\dots,N\\ & \boldsymbol{\lambda}=\mathbf{W}_{N+1}\mathbf{y}_N+\mathbf{b}_{N+1},\end{aligned}$$

[Jara-Moroni et al., 2018]

Sensitivity to penalty parameter ρ

The solution is extremely sensitive to ρ

- **Too Large:** Convergence slow
- **Too Small:** Violate complimetarity condition

Need to find the smallest ρ that satisfies the complimentarity condition.

General compact formulation

$$
\begin{aligned}
\underset{\mathbf{d},\mathbf{y},\mathbf{v}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{y} \\
\text{subject to} \quad & \mathbf{A}\mathbf{d} \geqslant \mathbf{f}, \\
& \mathbf{V}\mathbf{d} + \mathbf{W}\mathbf{y} + \mathbf{b} = \mathbf{v}, \\
& \mathbf{0} \leqslant \mathbf{y}_i \perp \mathbf{v}_i \geqslant \mathbf{0}, \\
& \forall i = 1, \ldots, N,
\end{aligned}
$$

$\Longrightarrow$

Bilinear compact formulation

$$
\begin{aligned}
\underset{\mathbf{d},\mathbf{y},\mathbf{v}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{y} + \rho \mathbf{y}^\top \mathbf{v} \\
\text{subject to} \quad & \mathbf{A}\mathbf{d} \geqslant \mathbf{f} \quad : \boldsymbol{\lambda}, \\
& \mathbf{V}\mathbf{d} + \mathbf{W}\mathbf{y} + \mathbf{b} = \mathbf{v} \quad : \boldsymbol{\mu}, \\
& \mathbf{y}_i, \mathbf{v}_i \geqslant 0, \\
& \forall i = 1, \ldots, N
\end{aligned}
$$

Part of the derivation draw inspiration from Prop.16 in[Jara-Moroni et al., 2018]

Sensitivity to penalty parameter ρ

Strongly stationary point seeking relaxation

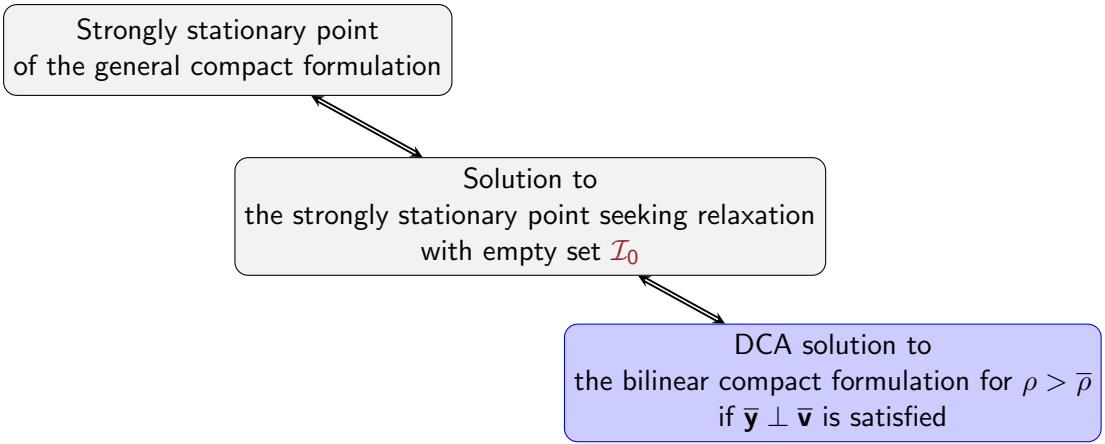
Let $(\tilde{\mathbf{d}}, \tilde{\mathbf{y}}, \tilde{\mathbf{v}})$ be some feasible point for the general compact formulation

$$
\begin{aligned}
\underset{\mathbf{d},\mathbf{y},\mathbf{v}}{\text{minimize}} \quad & \mathbf{c}^\top \mathbf{y} \\
\text{subject to} \quad & \mathbf{A}\mathbf{d} \geqslant \mathbf{f} \quad : \boldsymbol{\lambda}, \\
& \mathbf{V}\mathbf{d} + \mathbf{W}\mathbf{y} + \mathbf{b} = \mathbf{v} \quad : \boldsymbol{\mu}, \\
& y_i = 0 \quad : \mu_i^{y_0} \quad \forall i \in \mathcal{I}_y(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}), \\
& y_i \geqslant 0 \quad : \mu_i^{y_+} \quad \forall i \in \mathcal{I}_v(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}), \\
& v_i = 0 \quad : \mu_i^{v_0} \quad \forall i \in \mathcal{I}_v(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}), \\
& v_i \geqslant 0 \quad : \mu_i^{v_+} \quad \forall i \in \mathcal{I}_y(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}), \\
& y_i, v_i \geqslant 0, \qquad \forall i \in \mathcal{I}_0(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}),
\end{aligned}
$$

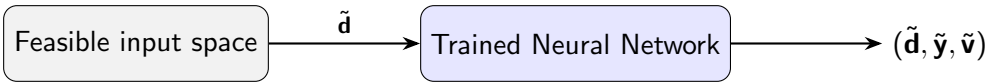
$$
\begin{aligned}
\mathcal{I}_y(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}) &\triangleq \{i \mid \tilde{y}_i = 0 < \tilde{v}_i\}, \\
\mathcal{I}_v(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}) &\triangleq \{i \mid \tilde{y}_i > 0 = \tilde{v}_i\}, \\
\mathcal{I}_0(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}) &\triangleq \{i \mid \tilde{y}_i = 0 = \tilde{v}_i\}.
\end{aligned}
$$

The solution $(\overline{\mathbf{d}}, \overline{\mathbf{y}}, \overline{\mathbf{v}})$ is the strongly stationary point of the general compact formulation if $\overline{\mathbf{y}}_i \perp \overline{\mathbf{v}}_i \quad \forall i = 1, ..., N$, where the strongly stationary point is the point that satisfies the KKTs of the original non-convex optimization.

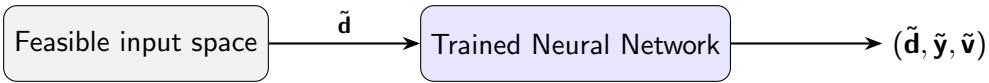
Connections between the three cases



Obtain feasible point $(\tilde{\mathbf{d}}, \tilde{\mathbf{y}}, \tilde{\mathbf{v}})$



Obtain feasible point $(\tilde{\mathbf{d}}, \tilde{\mathbf{y}}, \tilde{\mathbf{v}})$



Feasibility insight of trained neural network

$$\mathcal{I}_0(\tilde{\mathbf{y}}, \tilde{\mathbf{v}}) = \emptyset \longleftrightarrow \tilde{y}_i \neq \tilde{v}_i, \forall i = 1, \ldots, N \longleftrightarrow \mathbf{W}_i \mathbf{y}_{i-1} + \mathbf{b}_i \neq \mathbf{0}, , \forall i = 1, \ldots, N$$

- Often true
- If not, we can losslessly reduce NN size —- no more zero neuron!

Computing ρ from stationary point

- $(\overline{\mathbf{d}}, \overline{\mathbf{y}}, \overline{\mathbf{v}})$**:** From strongly stationary point seeking relaxation
- **Obtain** ρ**:** From partial Karush–Kuhn–Tucker conditions of the strongly stationary point seeking relaxation formulation and the bilinear compact formulation
- Combine lagrangian multipliers $\boldsymbol{\mu}^v = [\boldsymbol{\mu}^{v_0\top}\, \boldsymbol{\mu}^{v_+\top}]^\top$, $\boldsymbol{\mu}^y = [\boldsymbol{\mu}^{y_0\top}\, \boldsymbol{\mu}^{y_+\top}]^\top$

Relaxation partial KKT

$$
\begin{aligned}
&\mathbf{A}^\top \boldsymbol{\lambda} + \mathbf{V}^\top \boldsymbol{\mu}^v = \mathbf{0},\\
&\mathbf{W}^\top \boldsymbol{\mu}^v + \boldsymbol{\mu}^y = \mathbf{c},\\
&\mathbf{0} \leqslant \boldsymbol{\lambda} \perp \mathbf{A}\mathbf{d} - \mathbf{f} \geqslant \mathbf{0},\\
&\mathbf{y}^\top \boldsymbol{\mu}^y = 0, \quad \mathbf{v}^\top \boldsymbol{\mu}^v = 0,\\
&\mu_i^y \geqslant 0, \quad \forall i \in \mathcal{I}_v(\tilde{\mathbf{y}}, \tilde{\mathbf{v}})\\
&\mu_i^v \geqslant 0, \quad \forall i \in \mathcal{I}_y(\tilde{\mathbf{y}}, \tilde{\mathbf{v}})
\end{aligned}
$$

Bilinear partial KKT

$$
\begin{aligned}
&\mathbf{A}^\top \boldsymbol{\lambda} + \mathbf{V}^\top \boldsymbol{\mu} = \mathbf{0},\\
&\mathbf{0} \leqslant \mathbf{y} \perp -\mathbf{W}^\top \boldsymbol{\mu} + \mathbf{c} + \rho \mathbf{v} \geqslant \mathbf{0},\\
&\mathbf{0} \leqslant \boldsymbol{\lambda} \perp \mathbf{A}\mathbf{d} - \mathbf{f} \geqslant \mathbf{0},\\
&\mathbf{0} \leqslant \mathbf{v} \perp \rho \mathbf{y} + \boldsymbol{\mu} \geqslant \mathbf{0}.
\end{aligned}
$$

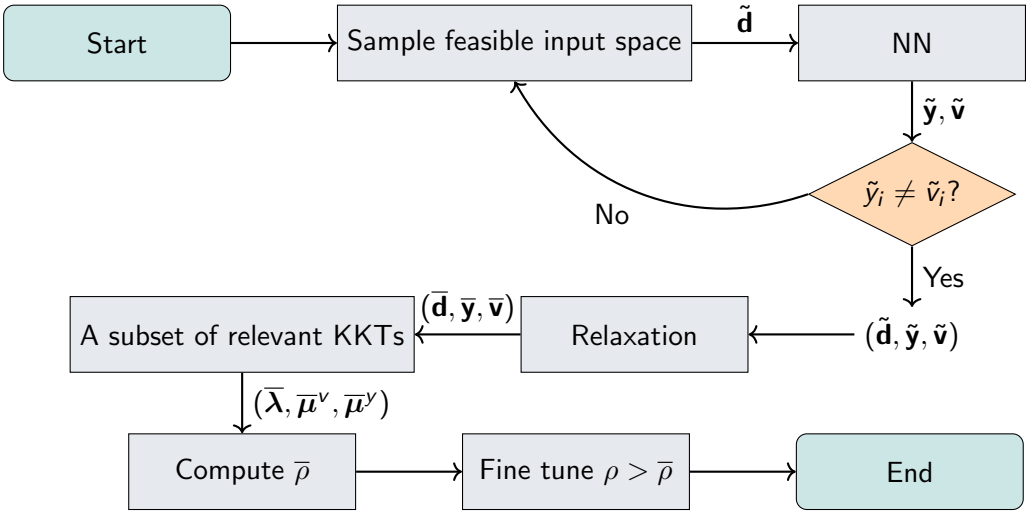
Computation for $\overline{\rho}$

We substitute $\boldsymbol{\mu}^v$ from relation partial KKT into $\boldsymbol{\mu}$ from bilinear partial KKT. Through some algebraic manipulation, we retrieve the following condition on $\overline{\rho}$

Lower bound for ρ

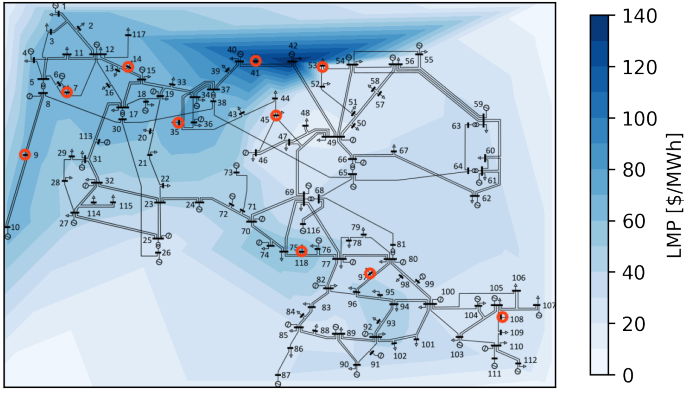
$$\bar{\rho} \triangleq \max\left\{0, \left\{-\frac{\overline{\mu}_i^y}{\bar{v}_i} \,\middle|\, \bar{v}_i > 0\right\}, \left\{-\frac{\overline{\mu}_i^v}{\bar{y}_i} \,\middle|\, \bar{y}_i > 0\right\}\right\}.$$

Computation for DCA penalty term ρ algorithm



Data center workload allocation in power grids

- Consider a network of spatially distributed data centers.
- Operators allocate workloads to minimize the cost of electricity consumption.



- In practice, we do not have network data or other parameters.
- Use a neural network to map the workload allocation to the cost of electricity.

Bilevel optimization to solve data center load allocation

- **Goal:** Allocate data center load to minimize the cost of electricity consumption.
- **Given:** Forecast of total electricity demand Δ.
- Data center allocation formulated as a bilevel optimization problem where $\mathbf{d}$ is the data center loads with upper and lower bounds $\overline{\mathbf{d}}$ and $\underline{\mathbf{d}}$. $\boldsymbol{\pi}$ is the price per demand.
- **Challenge:** Unknown underlying physical model

$$
\begin{aligned}
\underset{\mathbf{d}}{\text{minimize}} \quad & \boldsymbol{\pi}^\top \mathbf{d} \\
\text{subject to} \quad & \underline{\mathbf{d}} \leqslant \mathbf{d} \leqslant \overline{\mathbf{d}} \\
& \mathbf{1}^\top \mathbf{d} = \Delta \\
& \boldsymbol{\pi} \in \\
& \text{Unknown model}
\end{aligned}
$$

Bilevel optimization to solve data center load allocation

- **Goal:** Allocate data center load to minimize the cost of electricity consumption.
- **Given:** Forecast of total electricity demand Δ.
- Data center allocation formulated as a bilevel optimization problem where $\mathbf{d}$ is the data center loads with upper and lower bounds $\overline{\mathbf{d}}$ and $\underline{\mathbf{d}}$. $\boldsymbol{\pi}$ is the price per demand.
- **Challenge:** Unknown underlying physical model
- **Solution:** Replace by a neural network trained with a labeled dataset on historical record $\{(\mathbf{d}_1, \lambda_1), \ldots, (\mathbf{d}_n, \lambda_n)\}$.

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \quad & \boldsymbol{\pi}^\top \mathbf{d} \\ \text{subject to} \quad & \underline{\mathbf{d}} \leqslant \mathbf{d} \leqslant \overline{\mathbf{d}} \\ & \mathbf{1}^\top \mathbf{d} = \Delta \\ & \boldsymbol{\pi} \in \\ & \text{Unknown model} \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \quad & \mathbb{NN}(\mathbf{d}) \\ \text{subject to} \quad & \underline{\mathbf{d}} \leqslant \mathbf{d} \leqslant \overline{\mathbf{d}} \\ & \mathbf{1}^\top \mathbf{d} = \Delta \end{aligned}$$

Data generation for NN training

- **Method:** Using locational marginal prices (LMPs) as π derived from the optimal power flow (OPF) model. Use this data to train neural network.

Data generation for NN training

- **Method:** Using locational marginal prices (LMPs) as $\boldsymbol{\pi}$ derived from the optimal power flow (OPF) model. Use this data to train neural network.

$$
\begin{aligned}
\underset{\underline{\mathbf{p}} \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}}{\text{minimize}} \quad & \mathbf{p}^\top \mathbf{C}\mathbf{p} + \mathbf{c}^\top \mathbf{p} \\
\text{subject to} \quad & \mathbf{1}^\top (\mathbf{p} - \boldsymbol{\ell} - \mathbf{d}) = 0 \quad : \lambda \\
& |\mathbf{F}(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})| \leqslant \overline{\mathbf{f}} \quad : \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}
\end{aligned}
$$

- $\mathbf{p}$: generator dispatch
- $[\underline{\mathbf{p}}, \overline{\mathbf{p}}]$: feasible range of dispatch
- $\boldsymbol{\ell}$: conventional loads
- $\mathbf{d}$: data center loads
- $\mathbf{F}$: the matrix of power transfer distribution factors, which maps net power injections $(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})$ to power flows as $\mathbf{F}(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})$
- $\overline{\mathbf{f}}$: line capacity

Data generation for NN training

- **Method:** Using locational marginal prices (LMPs) as $\boldsymbol{\pi}$ derived from the optimal power flow (OPF) model. Use this data to train neural network.

$$
\begin{aligned}
\underset{\underline{\mathbf{p}} \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}}{\text{minimize}} \quad & \mathbf{p}^\top \mathbf{C}\mathbf{p} + \mathbf{c}^\top \mathbf{p} \\
\text{subject to} \quad & \mathbf{1}^\top(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d}) = 0 \quad : \lambda \\
& |\mathbf{F}(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})| \leqslant \overline{\mathbf{f}} \quad : \underline{\boldsymbol{\mu}}, \overline{\boldsymbol{\mu}}
\end{aligned}
$$

Locational Marginal Price(LMP)

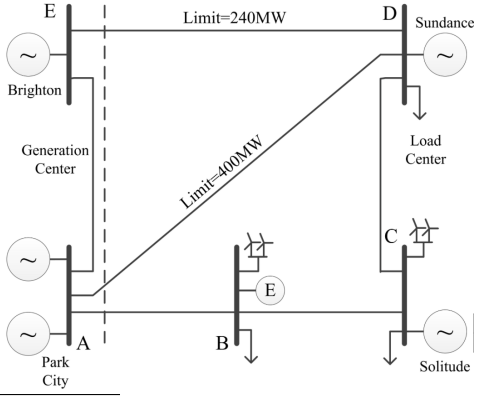
$$\boldsymbol{\pi} = \mathbf{1}\lambda^\star - \mathbf{F}^\top \overline{\boldsymbol{\mu}}^\star + \mathbf{F}^\top \underline{\boldsymbol{\mu}}^\star$$

[Chatzivasileiadis, 2018]

- $\mathbf{p}$: generator dispatch
- $[\underline{\mathbf{p}}, \overline{\mathbf{p}}]$: feasible range of dispatch
- $\boldsymbol{\ell}$: conventional loads
- $\mathbf{d}$: data center loads
- $\mathbf{F}$: the matrix of power transfer distribution factors, which maps net power injections $(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})$ to power flows as $\mathbf{F}(\mathbf{p} - \boldsymbol{\ell} - \mathbf{d})$
- $\overline{\mathbf{f}}$: line capacity

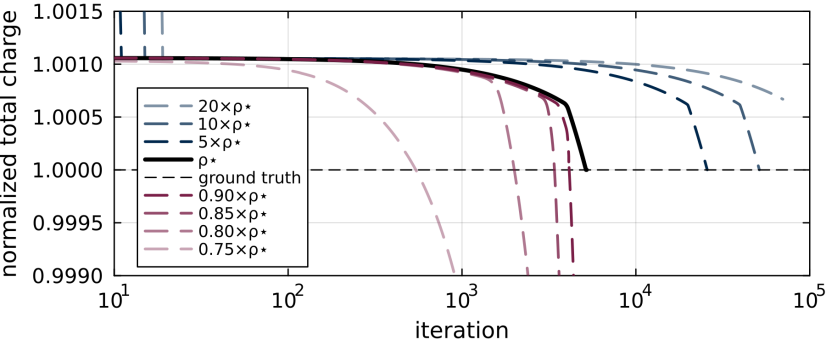
Experiment on PJM-5 bus system

- 3 loads act as data centers with demand in the range of $[0.8, 1]$ of the nominal value.
- NN with 2 hidden layers, 50 neurons each. $10,000$ training samples
- Global solution is computed using SOS1 constraints and Gurobi solver.



[Zhou, 2023]

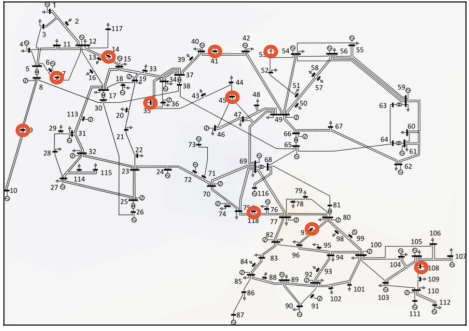
Experiment on PJM-5 bus system



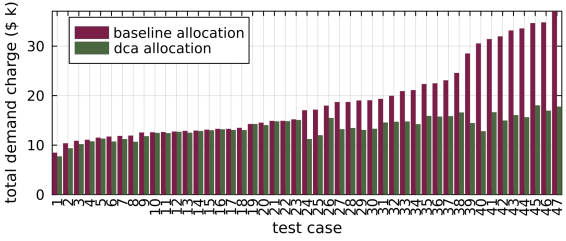
- $\rho^\star$: **fast convergence**
- $\rho < \rho^\star$: Does not satisfy complementarity condition
- $\rho > \rho^\star$: Slow convergence

Experiment on IEEE 118 bus system

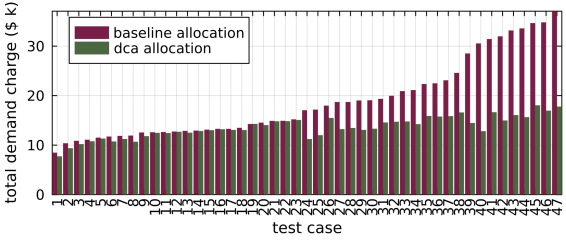
- **NN training:** NN with 5-hidden layer and $1,000$ ReLUs, trained on $12,500$ demand samples in the range of $[0, 100]$ MWh.
- **DCA solution:** Computes a new demand allocation to improve the baseline.
- **Samples:** Randomly picked 50 baseline cases, of which DCA could successfully converge and 47 of them roughly match the output of the OPF model.



Experiment on IEEE 118 bus system

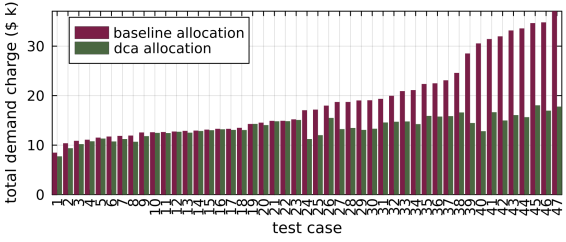


Experiment on IEEE 118 bus system



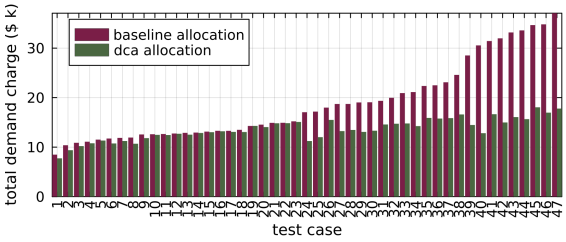
- **First 23 cases:** The savings are small. Lightly congested with little to no benefit from spatial load redistribution
- **Remaining cases:** Significant improvement

Experiment on IEEE 118 bus system



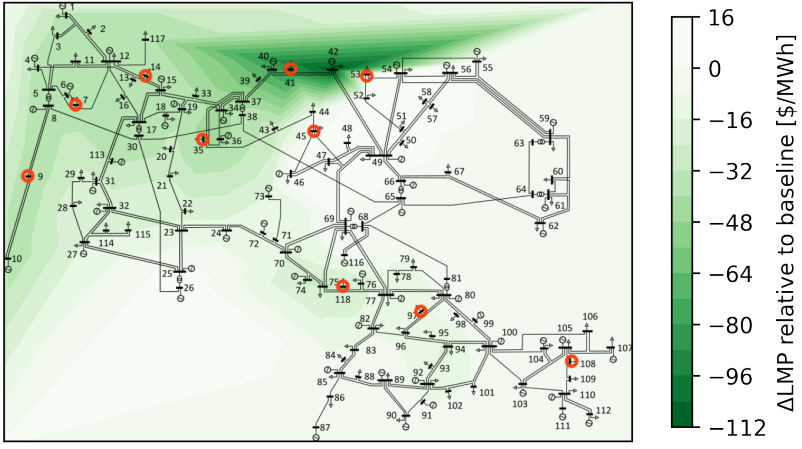
- **First 23 cases:** The savings are small. Lightly congested with little to no benefit from spatial load redistribution
- **Remaining cases:** Significant improvement

Experiment on IEEE 118 bus system



- **First 23 cases:** The savings are small. Lightly congested with little to no benefit from spatial load redistribution
- **Remaining cases:** Significant improvement
- The smaller ρ consistently leads to faster convergence.
- Each iteration takes ≈ 0.37 seconds, allowing for convergence in around 1 hour on average.

Experiment on IEEE 118 bus system



- The difference between the LMPs induced at baseline and the DCA solutions.
- LMPs were significantly reduced at the data centers.

Conclusions

- Developed a difference-of-convex algorithm to solve decision-making problems with objectives or constraints represented by trained neural networks.
- Avoided the computational bottleneck of ReLU logical constraints through informed penalization of such constraints in the objective function.
- Demonstrated on the data center load allocation application and showed significant reduction in cost compared to baseline.

Questions?

References I

Chatzivasileiadis, S. (2018).
Lecture notes on optimal power flow (opf).
arXiv preprint arXiv:1811.00943.

Dathathri, S., , et al. (2020).
Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming.
Advances in Neural Information Processing Systems, pages 5318–5331.

Dvorkin, V. (2025).
Lecture slides: Computational power systems.
Unpublished lecture slides, accessed on 2025-04-30.

Fazlyab, M., Morari, M., and Pappas, G. J. (2020).
Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming.
IEEE Trans. Autom. Control., 67(1):1–15.

References II

Grimstad, B. and Andersson, H. (2019).
ReLU networks as surrogate models in mixed-integer linear programs.
Comput. Chem. Eng., 131:106580.

Jara-Moroni, F., Pang, J.-S., and Wächter, A. (2018).
A study of the difference-of-convex approach for solving linear programs with complementarity constraints.
Mathematical Programming, 169:221–254.

Tjeng, V., Xiao, K., and Tedrake, R. (2017).
Evaluating robustness of neural networks with mixed integer programming.
Preprint.

Turner, M. et al. (2024).
Pyscipopt-ml: Embedding trained machine learning models into mixed-integer programs.
Preprint.

References III

Wang, S. et al. (2021).
Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification.
Advances in neural information processing systems, 34:29909–29921.

Zhou, A. (2023).
Pjm 5 bus data.
https://dx.doi.org/10.21227/7ex2-1t83.
Accessed on 2025-04-30.

The End

Backup I: Derivation of LMP

- **Partial Lagrangian function** (dualize the coupling constraints only):

$$\begin{aligned}\mathcal{L}(\mathbf{p}, \lambda, \overline{\boldsymbol{\mu}}, \underline{\boldsymbol{\mu}}) =& c(\mathbf{p}) - \lambda \mathbf{1}^\top (\mathbf{p} - \mathbf{d}) \\ &+ \overline{\boldsymbol{\mu}}^\top (\mathbf{F}(\mathbf{p} - \mathbf{d}) - \overline{\mathbf{f}}) + \underline{\boldsymbol{\mu}}^\top (-\mathbf{F}(\mathbf{p} - \mathbf{d}) - \overline{\mathbf{f}})\end{aligned}$$

- Group terms corresponding to dispatch $\mathbf{p}$, demand $\mathbf{d}$ and line limits $\overline{\mathbf{f}}$:

$$\mathcal{L} = \mathcal{L}^{\mathbf{p}} + \mathcal{L}^{\mathbf{d}} + \mathcal{L}^{\mathbf{f}}, \quad \text{where}$$

$$\begin{aligned}\mathcal{L}^{\mathbf{p}}(\mathbf{p}, \lambda, \overline{\boldsymbol{\mu}}, \underline{\boldsymbol{\mu}}) &= c(\mathbf{p}) - (\mathbf{1}\lambda - \mathbf{F}^\top \overline{\boldsymbol{\mu}} + \mathbf{F}^\top \underline{\boldsymbol{\mu}})^\top \mathbf{p} \\ \mathcal{L}^{\mathbf{d}}(\lambda, \overline{\boldsymbol{\mu}}, \underline{\boldsymbol{\mu}}) &= (\mathbf{1}\lambda - \mathbf{F}^\top \overline{\boldsymbol{\mu}} + \mathbf{F}^\top \underline{\boldsymbol{\mu}})^\top \mathbf{d} \\ \mathcal{L}^{\mathbf{f}}(\overline{\boldsymbol{\mu}}, \underline{\boldsymbol{\mu}}) &= -(\underline{\boldsymbol{\mu}} + \overline{\boldsymbol{\mu}})^\top \overline{\mathbf{f}}\end{aligned}$$

Power dispatch $\mathbf{p}$ and demand $\mathbf{d}$ share the same multiplier but with oposite signs

[Dvorkin, 2025]

Backup I: Derivation of LMP

$$\boldsymbol{\pi}^\star(\lambda^\star, \overline{\boldsymbol{\mu}}^\star, \underline{\boldsymbol{\mu}}^\star) = \underbrace{\mathbf{1}\lambda^\star}_{\text{uniform}} - \underbrace{\mathbf{F}^\top(\overline{\boldsymbol{\mu}}^\star - \underline{\boldsymbol{\mu}}^\star)}_{\text{congestion}} \in \mathbb{R}^N$$

- $\pi_n^\star$ is the cost of supplying the next unit of demand at node n
- in case of congestion ($\overline{\boldsymbol{\mu}}^\star > 0$ or $\overline{\boldsymbol{\mu}}^\star > 0$), electricity price varies across the grid
- price at the reference bus is $\lambda^\star$ (the ref. column of $\mathbf{F}$ is zero)

[Dvorkin, 2025]

Backup II: DC-OPF (PTDF formulation)

- Use matrix $\mathbf{F} \in \mathbb{R}^{E \times N}$ of power transfer distribution factors (PTDF)
  - how the power flow in line e changes w.r.t. to the change of power injection at node n?
  - obtained by manipulating the DC bus admittance matrix $\mathbf{B}$
- Power flows $f = \mathbf{F}(\mathbf{p} - \mathbf{d})$ (distribution of net injections across power lines)

$$
\begin{aligned}
\underset{\mathbf{p}}{\text{minimize}} \quad & c(\mathbf{p}) && \textit{generation cost} \\
\text{subject to} \quad & \mathbf{1}^\top(\mathbf{p} - \mathbf{d}) = 0 && \textit{active power balance} \\
& |\mathbf{F}(\mathbf{p} - \mathbf{d})| \leqslant \overline{f} && \textit{power flow limits} \\
& \underline{\mathbf{p}} \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}} && \textit{min/max gen p-limits}
\end{aligned}
$$

[Dvorkin, 2025]

Constraints:

$$z \geqslant 0$$
$$\mu \geqslant 0$$
$$z\mu = 0$$

Big-M reformulation:

$$\begin{cases} \mu \leqslant Mu \\ z \leqslant M(1-u) \end{cases}$$

where $u \in \{0, 1\}$ is an auxiliary binary variable, and M is a large enough constant

[Dvorkin, 2025]

$$x_{i+1} = \mathsf{ReLU}\left(\mathbb{L}_i\left(x_i\right)\right).$$

$$x_{i+1} \geqslant 0\left[\lambda_i^{\mathrm{a}}\right], x_{i+1} \geqslant \mathbb{L}_i\left(x_i\right)\left[\lambda_i^{\mathrm{b}}\right]$$

$$x_{i+1} \odot \left(x_{i+1} - \mathbb{L}_i\left(x_i\right)\right) \leqslant 0\left[\lambda_i^{\mathrm{c}}\right], x_i \odot x_i - \left(\ell_i + u_i\right) \odot x_i + \ell_i \odot u_i \leqslant 0\left[\lambda_i^{\mathrm{d}}\right].$$

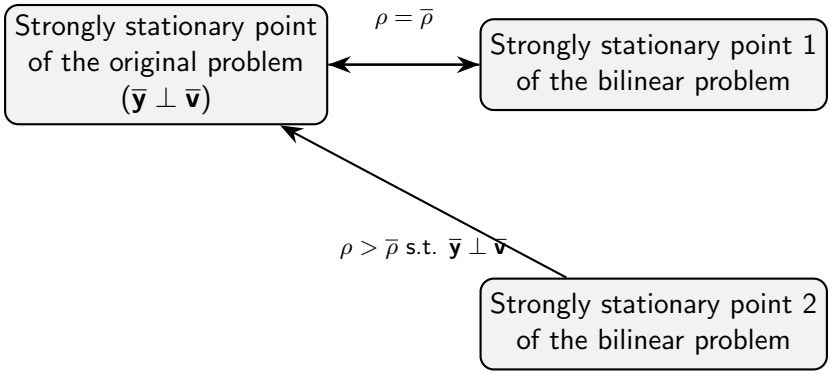
$$\begin{aligned}
\mathcal{L}\left(x_i, x_{i+1}, \lambda_i\right) &= \left(-x_{i+1}\right)^\top \lambda_i^{\mathrm{a}} + \left(\mathbb{L}_i\left(x_i\right) - x_{i+1}\right)^\top \lambda_i^{\mathrm{b}} \\
&\quad + \left(x_{i+1} \odot \left(x_{i+1} - \mathbb{L}_i\left(x_i\right)\right)\right)^\top \lambda_i^{\mathrm{c}} + \left(x_i \odot x_i - \left(\ell_i + u_i\right) \odot x_i + \ell_i \odot u_i\right)^\top \lambda_i^{\mathrm{d}} \\
&= \underbrace{\left(\ell_i \odot u_i\right)^\top \lambda_i^{\mathrm{d}}}_{\text{independent of } x_i, x_{i+1}} \underbrace{- x_{i+1}^\top \lambda_i^{\mathrm{a}} + \left(\mathbb{L}_i\left(x_i\right)\right)^\top \lambda_i^{\mathrm{b}} - x_{i+1}^\top \lambda_i^{\mathrm{b}} - x_i^\top \left(\left(\ell_i + u_i\right) \odot \lambda_i^{\mathrm{d}}\right)}_{\text{linear in } x_i, x_{i+1}} \\
&\quad \underbrace{+ x_{i+1}^\top \operatorname{diag}\left(\lambda_i^{\mathrm{c}}\right) x_{i+1} - x_{i+1}^\top \operatorname{diag}\left(\lambda_i^{\mathrm{c}}\right) \mathbb{L}_i\left(x_i\right) + x_i^\top \operatorname{diag}\left(\lambda_i^{\mathrm{d}}\right) x_i}_{\text{Quadratic in } x_i, x_{i+1}}.
\end{aligned}$$

>>> X [Dathathri et al., 2020]

Backup V: Neuron split and bound propagation

- ReLU relaxation: $w^\top \text{ReLU}(v) \geq w^\top \mathbf{D}v + b'$ where $\mathbf{D}$ is a diagonal matrix containing free variables $0 \leq \boldsymbol{\alpha}_j \leq 1$ only when $\mathbf{u}_j > 0 > \mathbf{l}_j$ and $w_j \geq 0$, while its rest values as well as constant b' are determined by $\mathbf{l}, \mathbf{u}, w$.
- $\min_{x \in \mathcal{C}} f(x) \geq \min_{x \in \mathcal{C}} \boldsymbol{a}_{\text{CROWN}}^\top x + c_{\text{CROWN}}$ where $\boldsymbol{a}_{\text{CROWN}}$ and c_{CROWN} can be computed using $\mathbf{W}^{(i)}, \mathbf{b}^{(i)}, \mathbf{l}^{(i)}, \mathbf{u}^{(i)}$ in polynomial time.

Backup VI: Additional argument for ρ



- The fine tuning to make sure complementarity condition is satisfied is easy because intuitively, if $\overline{\rho}$ works as a penalty parameter in one case, the penalty is penalize the bilinear part enough to satisfy the complementarity condition.
- This result is also proven experimentally.