

# ► L2O-MINLP

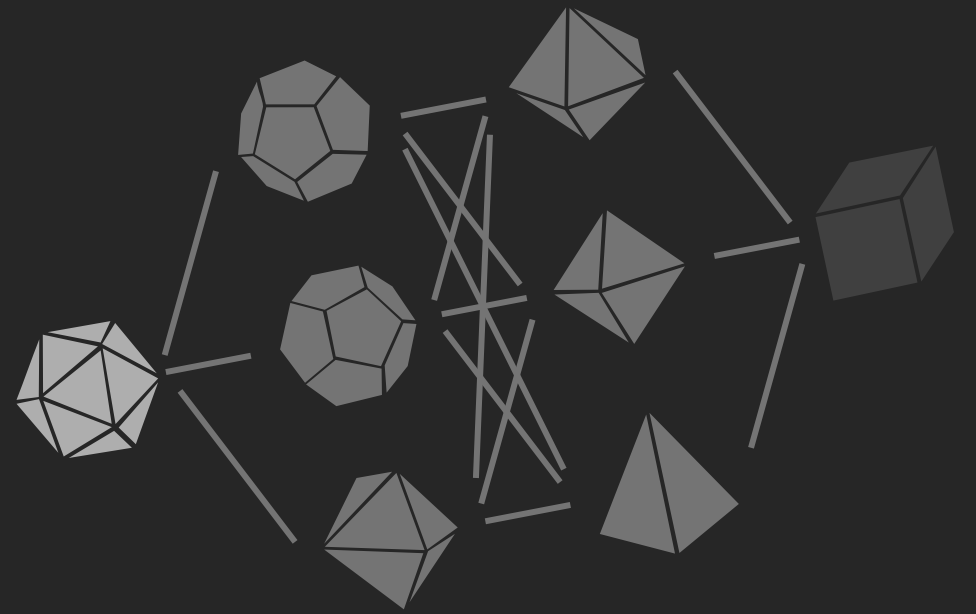
Learning to Optimize for  
Mixed-Integer Non-Linear Programming



Mechanical & Industrial Engineering  
UNIVERSITY OF TORONTO



**Pacific Northwest**  
NATIONAL LABORATORY



**Presented by Bo Tang**  
**Toronto, Mar 14, 2025**

# Authors



**Bo Tang**

PhD Candidate  
Department of Mechanical &  
Industrial Engineering  
University of Toronto



**Elias B. Khalil**

Assistant Professor  
Department of Mechanical &  
Industrial Engineering  
University of Toronto  
SCALE AI Research Chair  
Data-Driven Algorithms for Modern  
Supply Chains



**Ján Drgoňa**

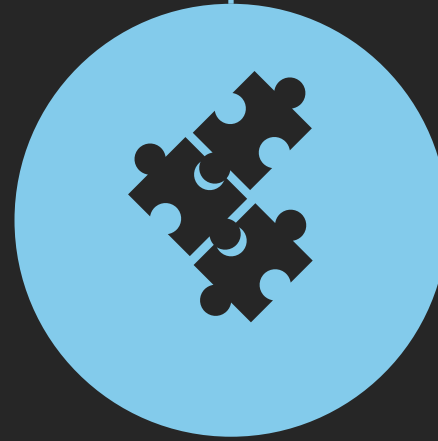
Associate Professor  
Department of Civil and Systems  
Engineering and the Ralph S.  
O'Connor Sustainable Energy  
Institute  
Johns Hopkins University

# Motivation

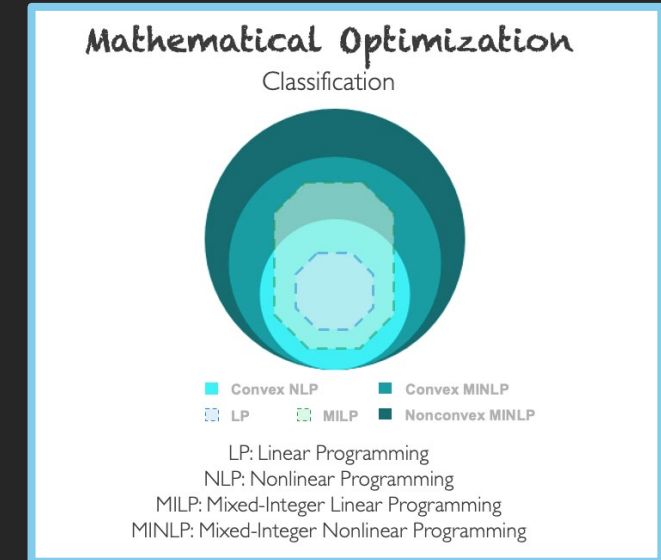
**Why does MINLP  
matter?**

# Motivation

## Why does MINLP matter?



**MINLP is general but very hard:**  
Combinatorial Complexity + Non-Convexity.



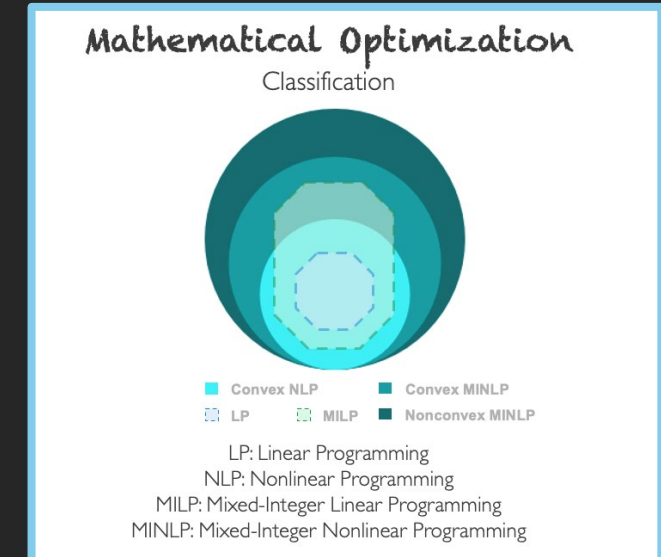
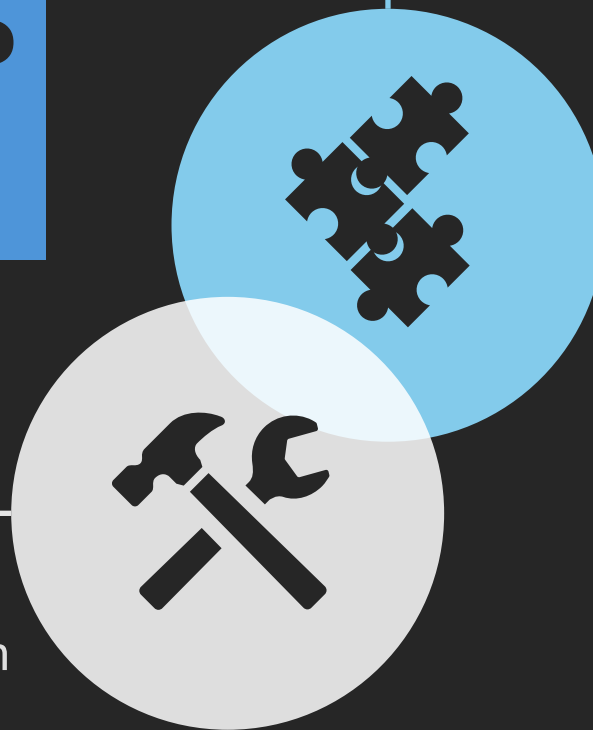
# Motivation

## Why does MINLP matter?

**MINLP is general but very hard:**  
Combinatorial Complexity + Non-Convexity.

### Solvers are limited:

Traditional solvers struggle with large-scale MINLP problems.



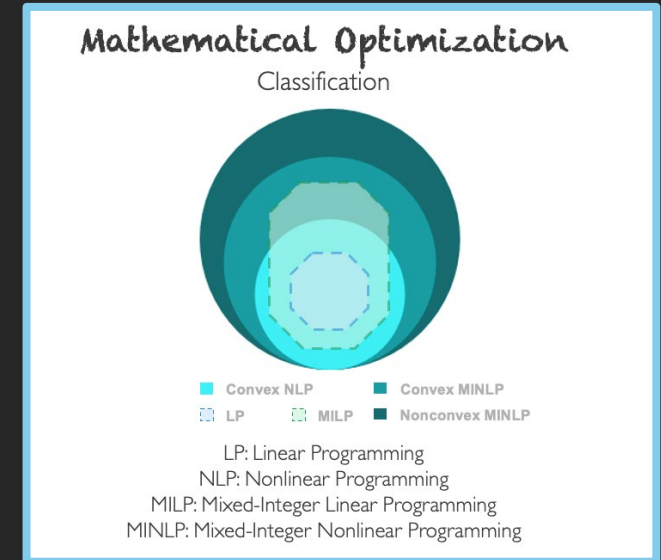
**GUROBI**  
OPTIMIZATION



# Motivation

## Why does MINLP matter?

**MINLP is general but very hard:**  
Combinatorial Complexity + Non-Convexity.



### Solvers are limited:

Traditional solvers struggle with large-scale MINLP problems.

### Solution time is tight:

Real-world applications often require solutions within a short time window.

e.g., Optimal Control, Energy Management, etc.



**GUROBI**  
OPTIMIZATION



# Introduction to L2O

## 01 What is L2O?

L2O is a data-driven approach that leverages machine learning to improve optimization processes.

# Introduction to L2O

## 01 What is L2O?

L2O is a data-driven approach that leverages machine learning to improve optimization processes.

L2O enables high-quality solutions with significantly reduced computational cost.

## 02 Why do we use L2O?



# Introduction to L2O

## 01 What is L2O?

L2O is a data-driven approach that leverages machine learning to improve optimization processes.

L2O enables high-quality solutions with significantly reduced computational cost.

## 02 Why do we use L2O?

L2O learns from optimal or near-optimal solutions to either generate solutions directly or guide solvers in narrowing the search space.

What we did here!

## 03 How does L2O work?

# Traditional Methods vs. L2O Approaches

## Traditional Methods

- Require costly iterative procedures (e.g., branch-and-bound, matrix inversion) that scale poorly with problem size.
- Solve from scratch for every instance, even if similar problems have been solved before.
- Require manual tuning of heuristics and parameters for good performance.
- Optimality is theoretical guaranteed through exact methods.

## L2O Approaches

- Bypass expensive iterations by learning direct mappings or guiding solvers for faster convergence.
- Leverage past patterns to generalize and quickly generate solutions for new instances.
- Automate tuning by learning optimization strategies from data.
- No guarantee of optimality or even feasibility.



# Traditional Methods vs. L2O Approaches

## Traditional Methods

- Require costly iterative procedures (e.g., branch-and-bound, matrix inversion) that scale poorly with problem size.
- Solve from scratch for every instance, even if similar problems have been solved before.
- Require manual tuning of heuristics and parameters for good performance.
- Optimality is theoretical guaranteed through exact methods.

## L2O Approaches

- Bypass expensive iterations by learning direct mappings or guiding solvers for faster convergence.
- Leverage past patterns to generalize and quickly generate solutions for new instances.
- Automate tuning by learning optimization strategies from data.
- No guarantee of optimality or even feasibility.



# Traditional Methods vs. L2O Approaches

## Traditional Methods

- Require costly iterative procedures (e.g., branch-and-bound, matrix inversion) that scale poorly with problem size.
- Solve from scratch for every instance, even if similar problems have been solved before.
- Require manual tuning of heuristics and parameters for good performance.
- Optimality is theoretical guaranteed through exact methods.

## L2O Approaches

- Bypass expensive iterations by learning direct mappings or guiding solvers for faster convergence.
- Leverage past patterns to generalize and quickly generate solutions for new instances.
- Automate tuning by learning optimization strategies from data.
- No guarantee of optimality or even feasibility.



# Traditional Methods vs. L2O Approaches

## Traditional Methods

- Require costly iterative procedures (e.g., branch-and-bound, matrix inversion) that scale poorly with problem size.
- Solve from scratch for every instance, even if similar problems have been solved before.
- Require manual tuning of heuristics and parameters for good performance.
- Optimality is theoretical guaranteed through exact methods.

## L2O Approaches

- Bypass expensive iterations by learning direct mappings or guiding solvers for faster convergence.
- Leverage past patterns to generalize and quickly generate solutions for new instances.
- Automate tuning by learning optimization strategies from data.
- No guarantee of optimality or even feasibility.



# Problem Formulation

## MINLP Formulation:

$$\min_x f(x)$$

$$\text{s. t. } g(x) \leq 0$$

$$x \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}$$

# Problem Formulation

## Parametric MINLP Formulation:

$$\begin{aligned} \min_x \quad & f(x, \xi) \\ \text{s. t.} \quad & g(x, \xi) \leq 0 \\ & x \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z} \\ & \xi \in \Xi \end{aligned}$$

Parameters  $\xi$  influence the objective and constraints of optimization problems.

Multiple Instances  $\left\{ \begin{array}{l} \xi_1 \rightarrow x_1^* \\ \xi_2 \rightarrow x_2^* \\ \vdots \\ \xi_m \rightarrow x_m^* \end{array} \right.$

# Problem Formulation

## Prediction for Parametric MINLP Formulation:

$$\min_{\Theta} \mathbb{E}_{\Xi} [f(\hat{\mathbf{x}}, \xi)]$$

Expected objective function:

$$\text{s. t. } g(\hat{\mathbf{x}}, \xi) \leq 0$$

$$\mathbb{E}_{\Xi} [f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}_i, \xi_i)$$

$$\hat{\mathbf{x}} \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}$$

$$\xi \in \Xi$$

Machine learning model  $\psi_{\Theta}$  with inputs  $\xi$  and weights  $\Theta$

Solution prediction

$$\hat{\mathbf{x}} = \psi_{\Theta}(\xi)$$



# Problem Formulation

Prediction for Parametric MINLP Formulation:

$$\min_{\theta} \mathbb{E}_{\xi} [f(\hat{x}, \xi)]$$

Minimize expected objective function:

**IT SEEMS IDEAL:**

What Prevents L2O from  
Extending to this MINLP?

$$\mathbb{E}_{\xi} [f(\hat{x}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{x}_i, \xi_i)$$

$$\hat{x} \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}$$

$$\xi \in \Xi$$

Machine learning model  $\psi_{\theta}$  with input  $\xi$  and weights  $\theta$

Solution prediction

$$\hat{x} = \psi_{\theta}(\xi)$$

# **L**imitations of Existing L2O

---

**01** Collecting Solutions as Training Label is Very Expensive

**02** Neural Networks Cannot Directly Output Integer Values

**03** It is Difficult to Ensure Feasibility, Especially in Integers

# Limitations of Existing L2O

## 01 Collecting Solutions as Training Label is Very Expensive



Our Solution: We propose a Self-Supervised Approach without requiring solutions for training.

## 02 Neural Networks Cannot Directly Output Integer Values

## 03 It is Difficult to Ensure Feasibility, Especially in Integers

# Limitations of Existing L2O

## 01 Collecting Solutions as Training Label is Very Expensive



Our Solution: We propose a Self-Supervised Approach without requiring solutions for training.

## 02 Neural Networks Cannot Directly Output Integer Values



Our Solution: We introduce Integer Correction Layers to ensure integer feasibility.

## 03 It is Difficult to Ensure Feasibility, Especially in Integers

# Limitations of Existing L2O

## 01 Collecting Solutions as Training Label is Very Expensive



Our Solution: We propose a Self-Supervised Approach without requiring solutions for training.

## 02 Neural Networks Cannot Directly Output Integer Values



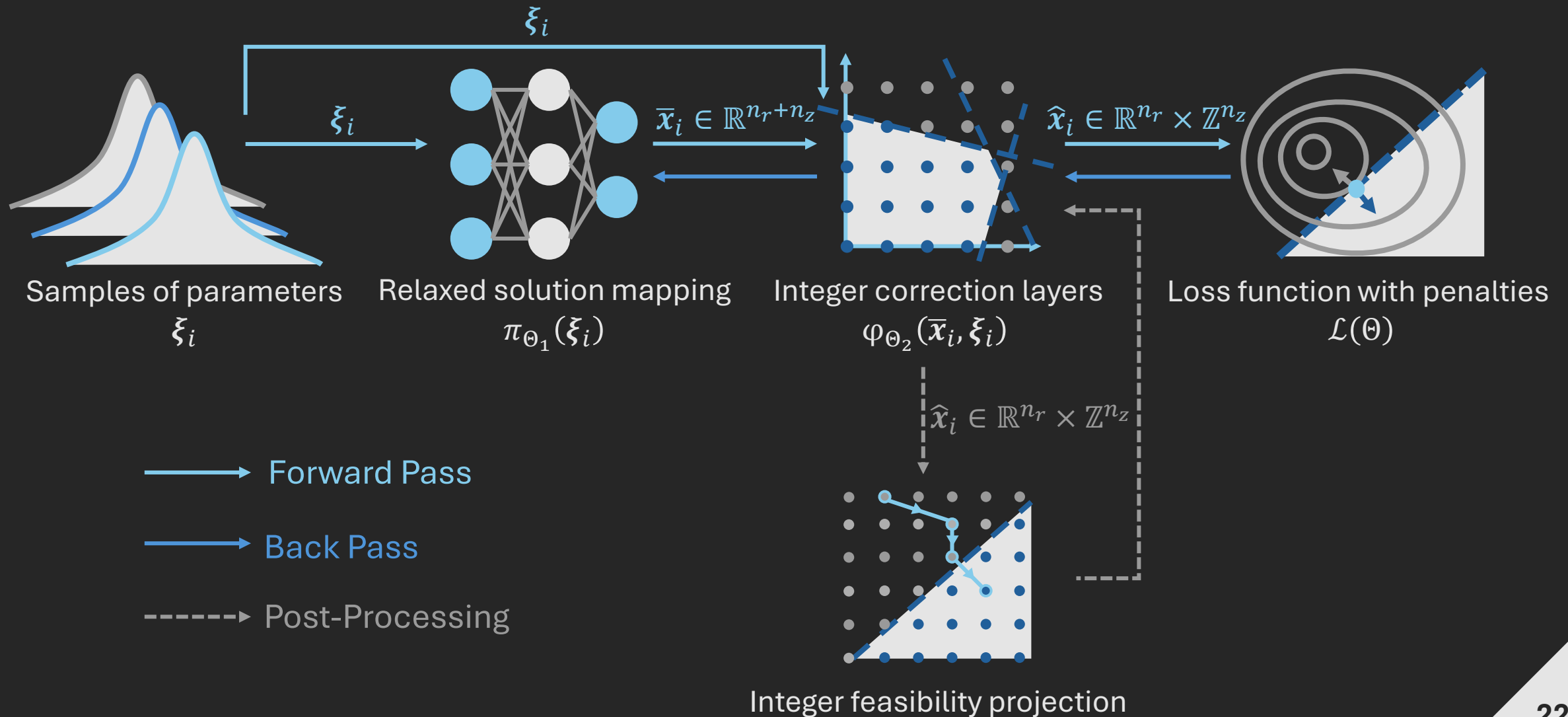
Our Solution: We introduce Integer Correction Layers to ensure integer feasibility.

## 03 It is Difficult to Ensure Feasibility, Especially in Integers

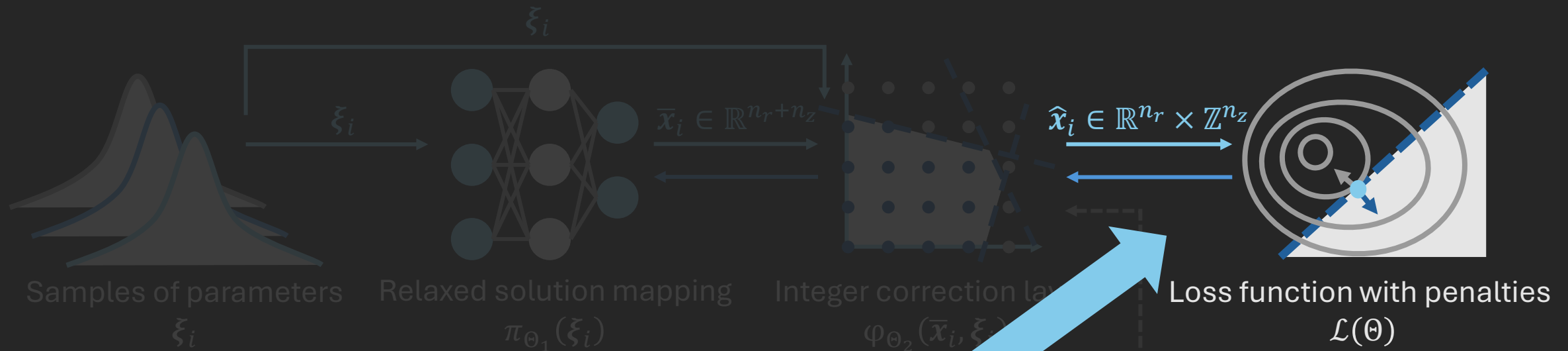


Our Solution: We introduce a gradient-based Feasibility Projection to adjust integer solutions.

# Methodology



# Loss Function



## Self-Supervised Learning:



The objective function naturally serves as a loss function, while constraint violations can be incorporated as penalty terms to enforce feasibility.

→ Forward

→ Backward

-----> Post-Processing

Integer feasibility projection

# Loss Function

## Loss Function with Penalty

$$\mathcal{L}(\Theta) = \sum_{i=1}^m [f(\hat{\mathbf{x}}_i, \xi_i) + \lambda \cdot \|g(\hat{\mathbf{x}}_i, \xi_i)_+\|_1]$$



# Loss Function

## Loss Function with Penalty

$$\mathcal{L}(\Theta) = \sum_{i=1}^m [\underbrace{f(\hat{x}_i, \xi_i)}_{\text{Objective Function}} + \lambda \cdot \|g(\hat{x}_i, \xi_i)_+\|_1]$$

# Loss Function

## Loss Function with Penalty

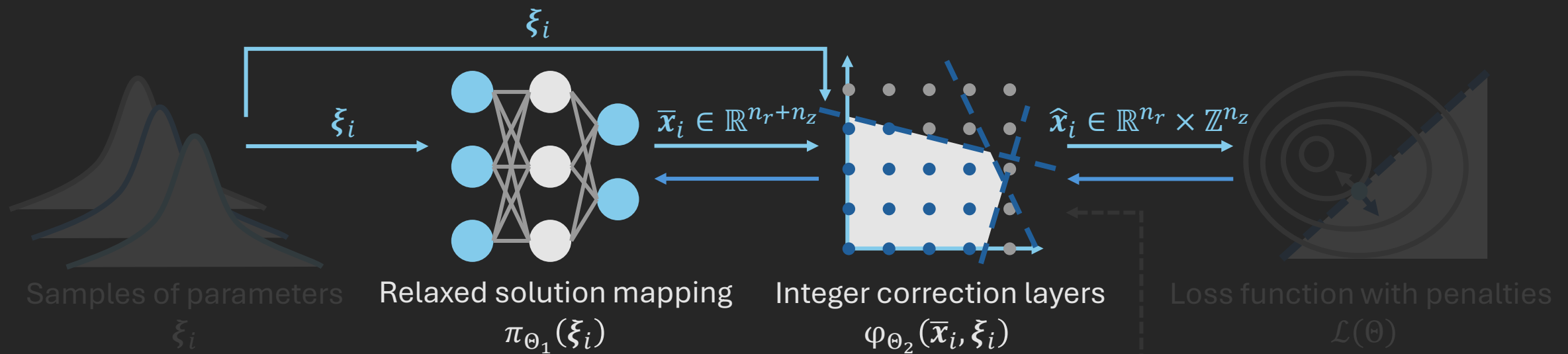
$$\mathcal{L}(\Theta) = \sum_{i=1}^m \left[ \underbrace{f(\hat{x}_i, \xi_i)}_{\text{Objective Function}} + \lambda \cdot \underbrace{\|g(\hat{x}_i, \xi_i)_+\|_1}_{\text{Constraints Violation}} \right]$$

# Loss Function

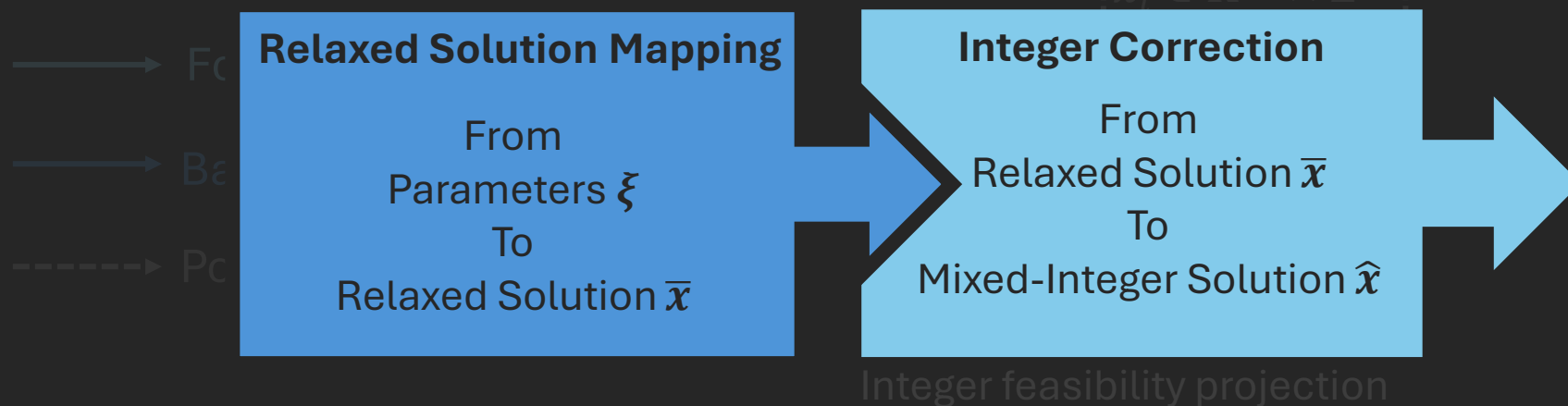
## Loss Function with Penalty

$$\mathcal{L}(\Theta) = \sum_{i=1}^m \left[ \underbrace{f(\hat{x}_i, \xi_i)}_{\text{Objective Function}} + \overbrace{\lambda}^{\text{Penalty Weight}} \cdot \underbrace{\|g(\hat{x}_i, \xi_i)_+\|_1}_{\text{Constraints Violation}} \right]$$

# Integer Correction Layers

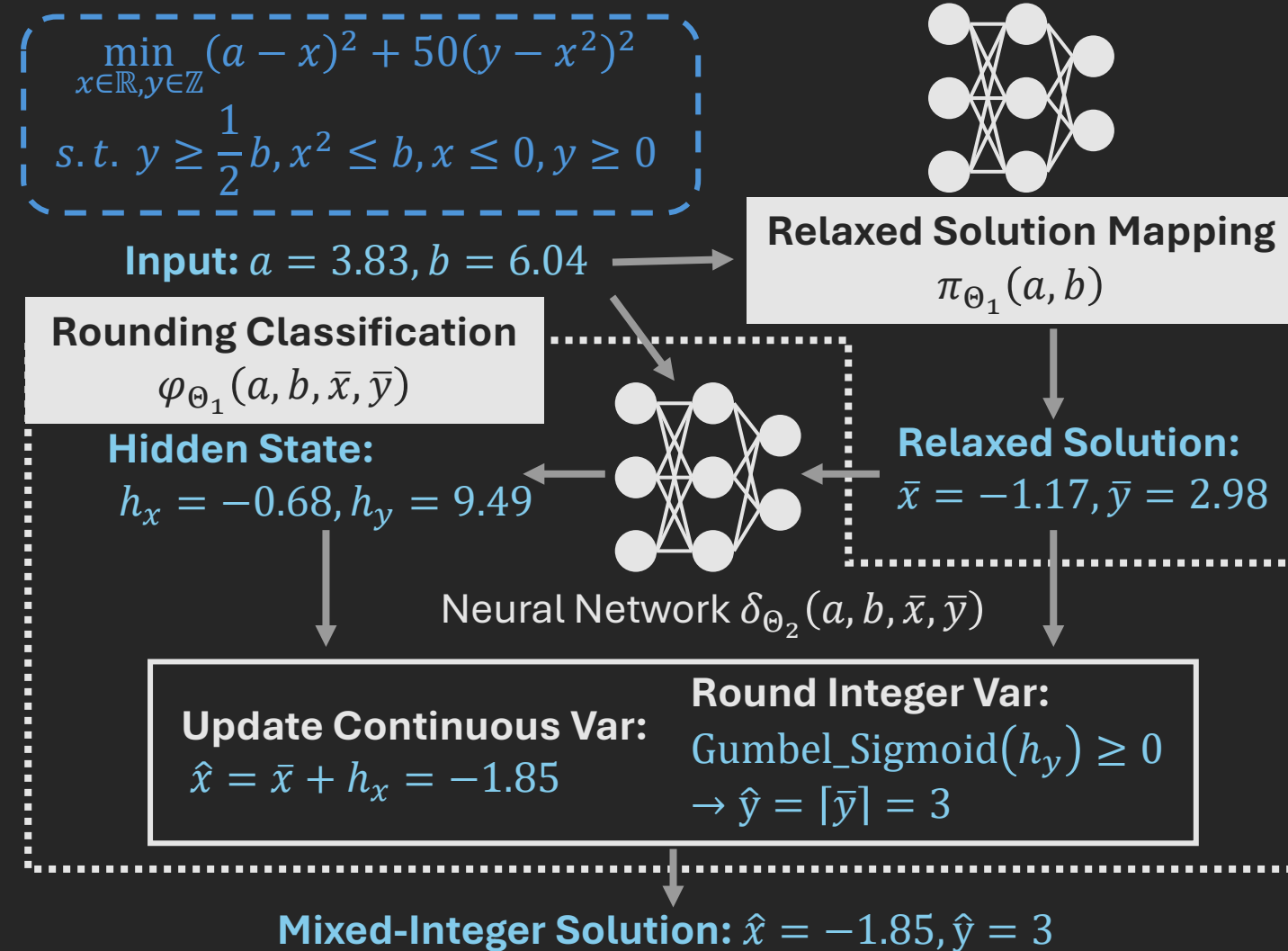


## Bi-Level Neural Network Architecture



# Integer Correction Layers

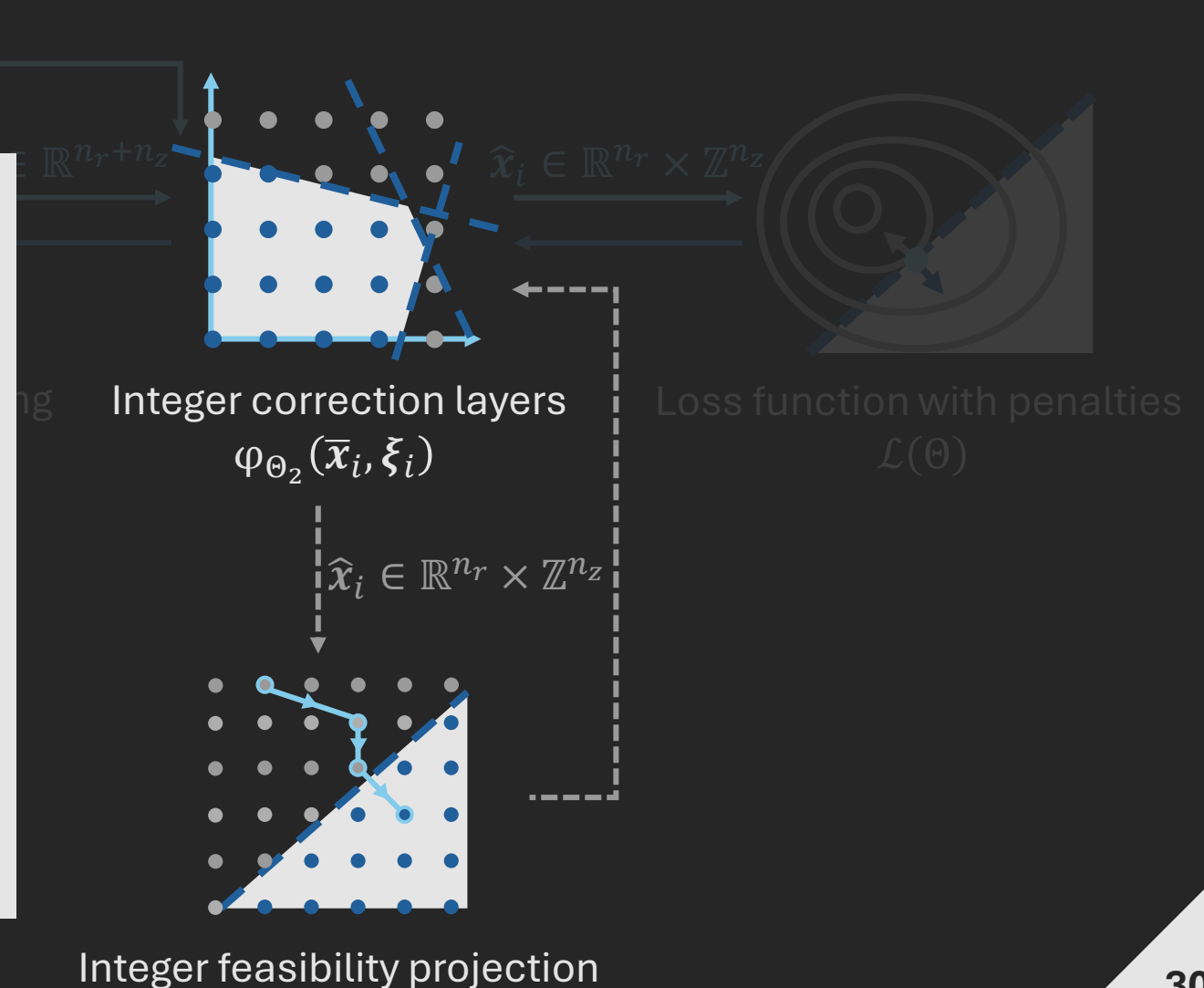
## Example



# Integer Feasibility Projection

Feasibility projection is a computationally efficient post-processing heuristic that refines neural network outputs while preserving integer constraints.

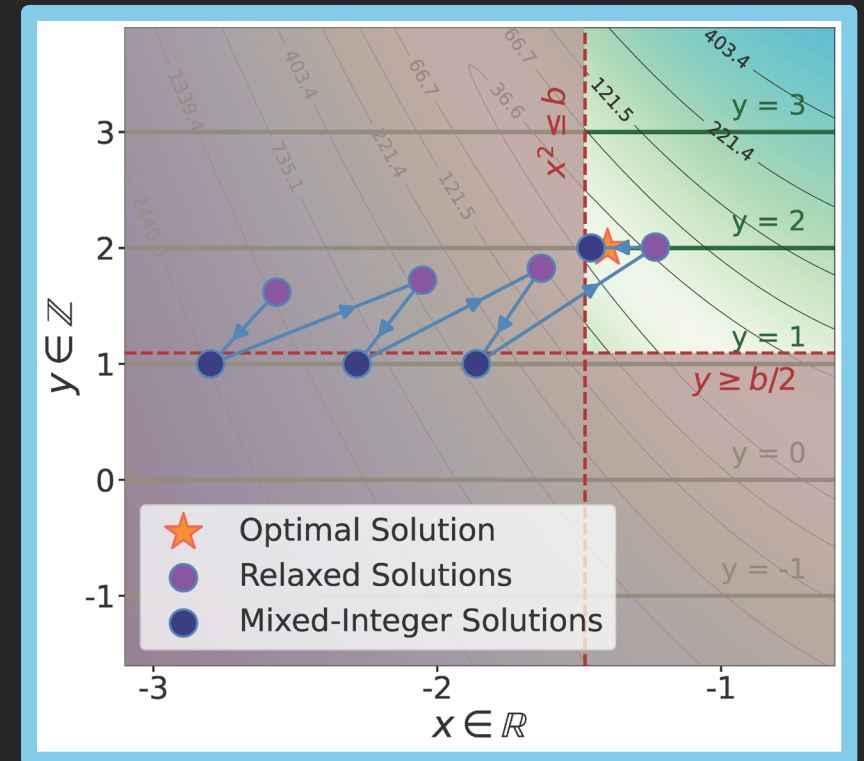
- **Computational Efficiency:** Avoids the need for repeated projections and second-order gradient computation during training.
- **Stable Training:** Keeps projection separate from training, preventing interference with the output of model.



# Integer Feasibility Projection

## Algorithm 2 Integer Feasibility Projection: Inference

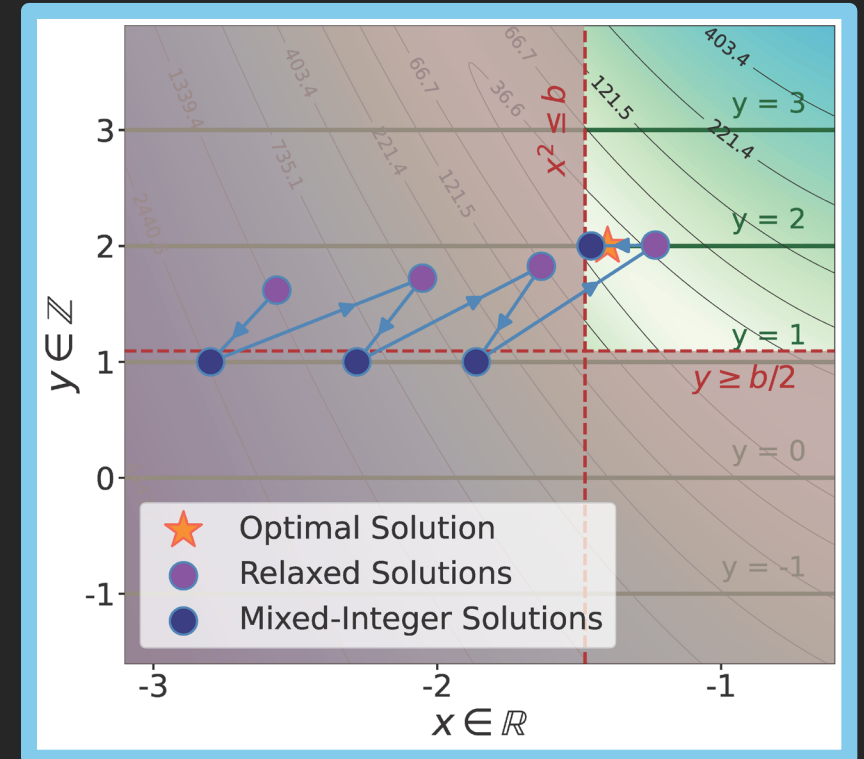
- 1: **Input:** parameters  $\xi^i$ , layers  $\pi_{\Theta_1}(\cdot)$  and  $\varphi_{\Theta_2}(\cdot)$ , step size  $\eta$
- 2: Predict a continuously relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \pi_{\Theta_1}(\xi^i)$
- 3: **while** True **do**
- 4:   Obtain a mixed-integer solution  $\hat{\mathbf{x}}^i \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}^i, \xi^i)$
- 5:   Compute feasibility violation  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1$
- 6:   **if**  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) = 0$  **then**
- 7:     Break
- 8:   **else**
- 9:     Update relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \bar{\mathbf{x}}^i - \eta \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}^i, \xi^i)$
- 10:   **end if**
- 11: **end while**
- 12: **Output:** a mixed-integer solution  $\hat{\mathbf{x}}^i$



# Integer Feasibility Projection

## Algorithm 2 Integer Feasibility Projection: Inference

- 1: **Input:** parameters  $\xi^i$ , layers  $\pi_{\Theta_1}(\cdot)$  and  $\varphi_{\Theta_2}(\cdot)$ , step size  $\eta$
- 2: Predict a continuously relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \pi_{\Theta_1}(\xi^i)$
- 3: **while** True **do**
- 4:   Obtain a mixed-integer solution  $\hat{\mathbf{x}}^i \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}^i, \xi^i)$
- 5:   Compute feasibility violation  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1$
- 6:   **if**  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) = 0$  **then**
- 7:     Break
- 8:   **else**
- 9:     Update relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \bar{\mathbf{x}}^i - \eta \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}^i, \xi^i)$
- 10:   **end if**           (1) Update relaxed solution to reduce violation
- 11: **end while**
- 12: **Output:** a mixed-integer solution  $\hat{\mathbf{x}}^i$

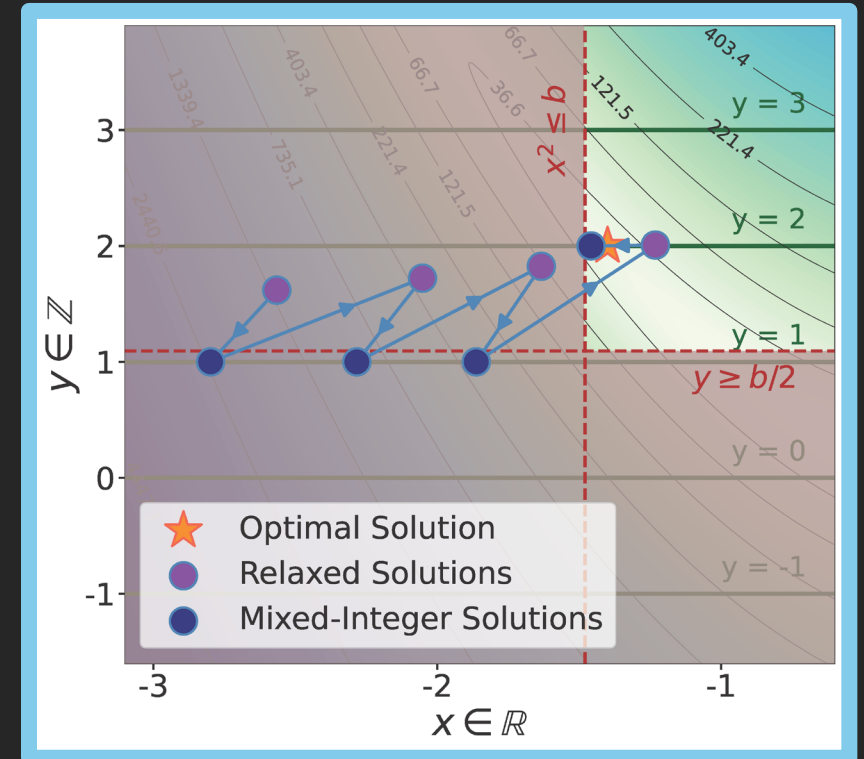




# Integer Feasibility Projection

## Algorithm 2 Integer Feasibility Projection: Inference

- 1: **Input:** parameters  $\xi^i$ , layers  $\pi_{\Theta_1}(\cdot)$  and  $\varphi_{\Theta_2}(\cdot)$ , step size  $\eta$
- 2: Predict a continuously relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \pi_{\Theta_1}(\xi^i)$
- 3: **while** True **do** (2) Differentiable integer correction
- 4:   Obtain a mixed-integer solution  $\hat{\mathbf{x}}^i \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}^i, \xi^i)$
- 5:   Compute feasibility violation  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1$
- 6:   **if**  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) = 0$  **then**
- 7:     Break
- 8:   **else**
- 9:     Update relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \bar{\mathbf{x}}^i - \eta \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}^i, \xi^i)$
- 10:   **end if**
- 11: **end while**
- 12: **Output:** a mixed-integer solution  $\hat{\mathbf{x}}^i$



### Algorithm 2 Integer Feasibility Projection: Inference

```

1: Input: parameters  $\xi^i$ , layers  $\pi_{\Theta_1}(\cdot)$  and  $\varphi_{\Theta_2}(\cdot)$ , step size  $\eta$ 
2: Predict a continuously relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \pi_{\Theta_1}(\xi^i)$ 
3: while True do
4:   Obtain a mixed-integer solution  $\hat{\mathbf{x}}^i \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}^i, \xi^i)$ 
5:   Compute feasibility violation  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1$ 
6:   if  $\mathcal{V}(\hat{\mathbf{x}}^i, \xi^i) = 0$  then           (3) Compute constraint violation
7:     Break                                based on mixed-integer solution
8:   else
9:     Update relaxed solution  $\bar{\mathbf{x}}^i \leftarrow \bar{\mathbf{x}}^i - \eta \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}^i, \xi^i)$ 
10:  end if
11: end while
12: Output: a mixed-integer solution  $\hat{\mathbf{x}}^i$ 

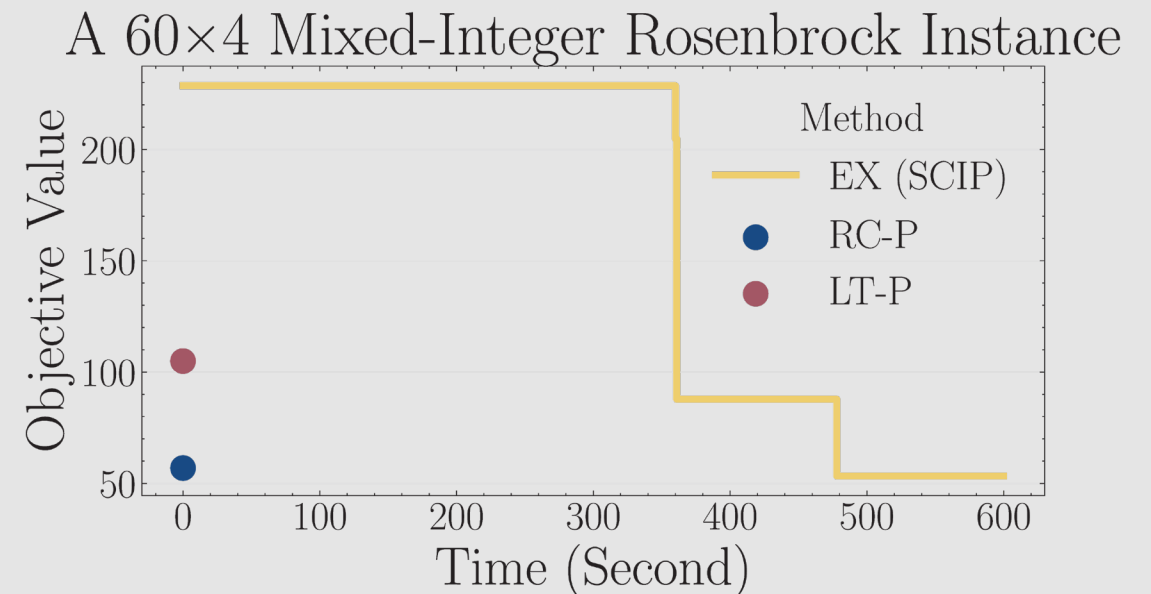
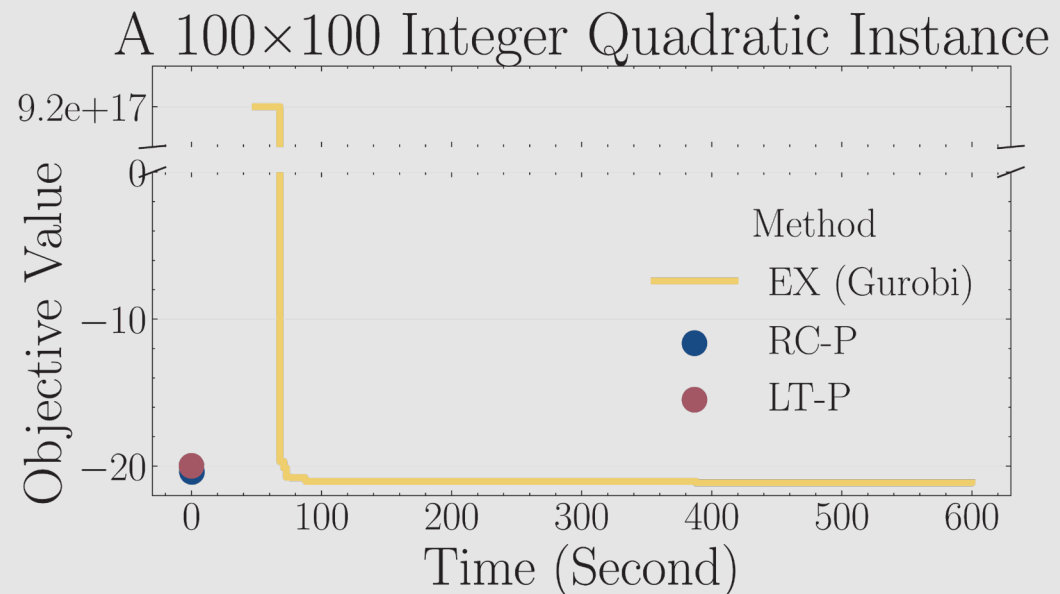
```



# Key Methods in Comparison

Method	Description
EX (Exact Solver)	Solves problems exactly using traditional solver with 1000-sec time-limit as a benchmark.
N1 (Root Node Solution)	Finds the first feasible solution from the root node of the solver, combining various heuristics.
RC (Rounding Classification)	A neural network-based correction layer that learns a classification to determine how to round each integer variable.
LT (Learnable Thresholding)	A neural network-based correction layer that learns a threshold value to decide to round up or down for each integer variable.
RC-P (RC + Feasibility Projection)	RC combined with feasibility projection, which corrects infeasibilities while preserving integer constraints.
LT-P (LT + Feasibility Projection)	LT combined with feasibility projection, which corrects infeasibilities while preserving integer constraints.

# Subsecond Solution



Exact solvers such as Gurobi and SCIP find better solutions over time but can be somewhat slow. In contrast, our methods (RC-P & LT-P) achieve high-quality feasible solutions within milliseconds.

# More Experiments

Result for IQPs. Each problem size is evaluated on a test set of 100 instance

Method	RC				RC-P				LT			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
100×100	-13.54	-13.6	96%	0.0022	-13.54	-13.57	100%	0.005	-13.65	-13.77	93%	0.0023
200×200	-31.62	-31.71	97%	0.0021	-31.62	-31.71	100%	0.005	-31.34	-31.61	95%	0.0022
500×500	-73.31	-73.38	86%	0.0025	-73.31	-73.38	100%	0.0065	-72.36	-72.48	94%	0.0026
1000×1000	-142.7	-142.7	82%	0.0042	-142.7	-142.7	100%	0.009	-142.6	-142.6	100%	0.0047
Method	LT-P				EX				N1			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
100×100	-13.65	-13.77	100%	0.01	-20.79	-20.78	100%	1237	1.5E+18	1.4E+18	100%	104.2
200×200	-31.34	-31.61	100%	0.0064	-	-	-	-	-	-	-	-
500×500	-72.36	-72.48	100%	0.0063	-	-	-	-	-	-	-	-
1000×1000	-142.6	-142.6	100%	0.0086	-	-	-	-	-	-	-	-

# More Experiments

Result for INPs. Each problem size is evaluated on a test set of 100 instance

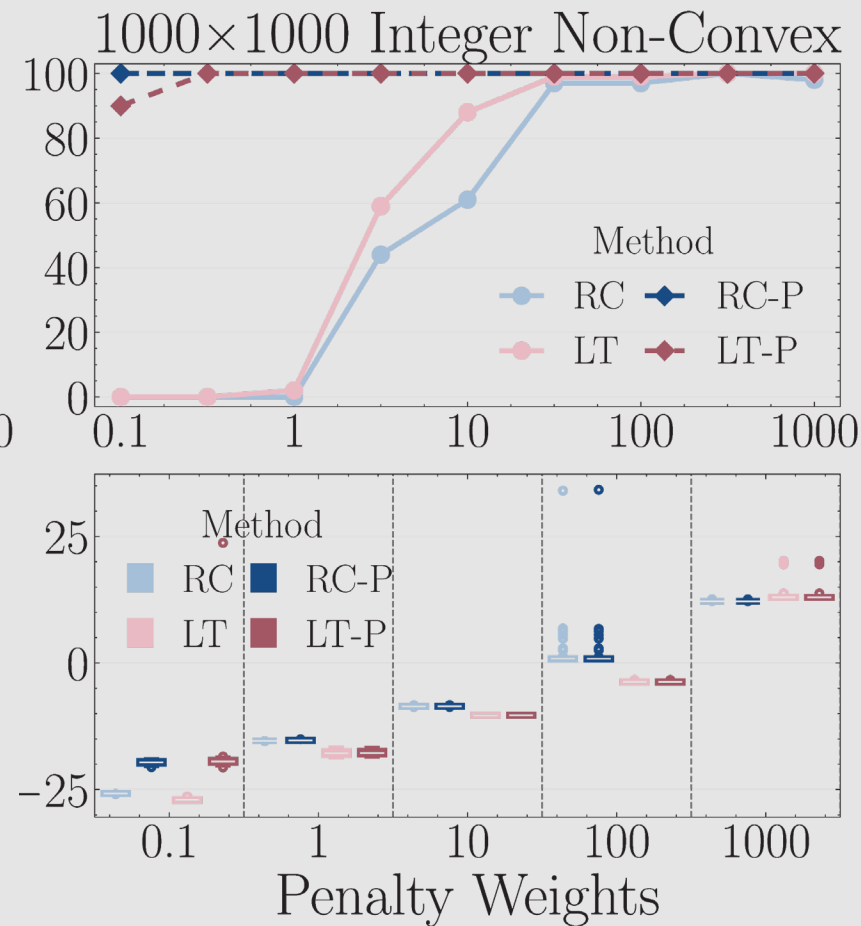
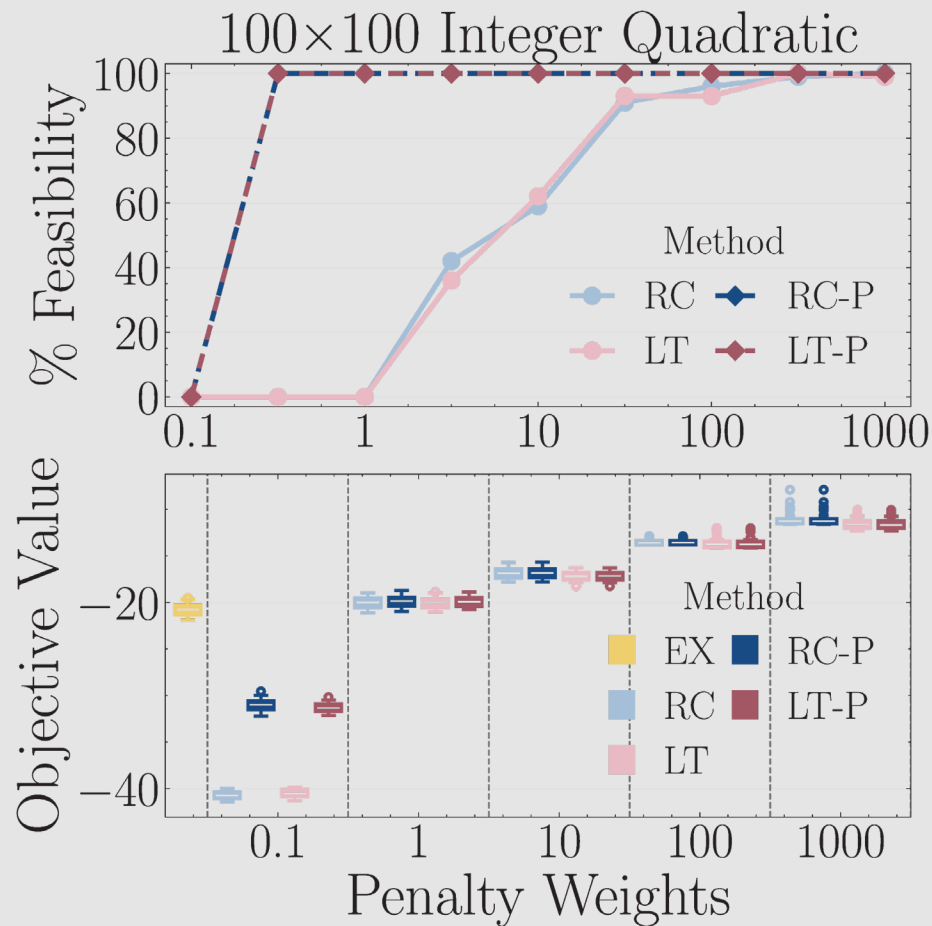
Method	RC				RC-P				LT			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
100×100	1.664	1.594	100%	0.0022	1.664	1.594	100%	0.0060	0.669	0.649	96%	0.0021
200×200	1.472	1.436	99%	0.0022	1.471	1.436	100%	0.0054	-0.356	-0.373	100%	0.0023
500×500	0.526	0.526	96%	0.0029	0.524	0.526	100%	0.0061	-1.374	-1.594	98%	0.0029
1000×1000	1.423	0.809	97%	0.0040	1.423	0.809	100%	0.0115	-3.744	-3.716	99%	0.0050
Method	LT-P				EX				N1			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
100×100	0.669	0.649	100%	0.0058	256.93	134.62	14%	1001	4411	155.2	14%	940.4
200×200	-0.356	-0.373	100%	0.0056	-	-	-	-	-	-	-	-
500×500	-1.374	-1.594	100%	0.0072	-	-	-	-	-	-	-	-
1000×1000	-3.744	-3.716	100%	0.0117	-	-	-	-	-	-	-	-

# More Experiments

**Result for MIRBs. Each problem size is evaluated on a test set of 100 instance**

Method	RC				RC-P				LT			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
20×4	59.39	48.86	100%	0.0019	59.39	48.86	100%	0.0048	62.51	63.40	100%	0.0020
200×4	503.5	461.7	99%	0.0021	504.2	461.7	100%	0.0052	622.8	626.0	100%	0.0026
50×4	5938	5792	99%	0.0033	5942	5792	100%	0.0070	5612	5558	97%	0.0030
1000×4	6.7E+4	6.7E+4	76%	0.0121	9.8E+4	7.3E+4	100%	0.0824	4.8E+4	3.5E+4	66%	0.0127
Method	LT-P				EX				N1			
	Obj	Obj	%	Time	Obj	Obj	%	Time	Obj	Obj	%	Time
Metric	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)	Mean	Median	Feasible	(Sec)
20×4	62.51	63.40	100%	0.0055	64.67	59.16	100%	1005	87.83	77.34	100%	0.0813
200×4	622.8	626.0	100%	0.0062	8.4E+5	908.8	100%	1002	3.7E+8	957.4	100%	0.2608
2000×4	5615	5558	100%	0.0071	4.7E+10	9262	96%	1002	8.3E+12	9379	95%	71.91
20000×4	8.0E+4	4.5E+4	100%	0.0639	1.1E+15	1.0E5	78%	1040	1.2E+15	1.0E5	78%	782

# Effect of Penalty Weight



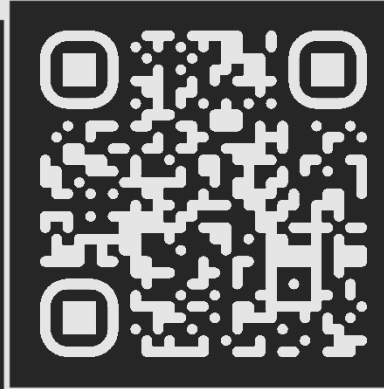
There is an inherent trade-off between achieving more feasible solutions and lower objective values prior to the integer feasibility projection (RC and LT).

After applying the integer feasibility projection (RC-P and LT-P), the high infeasibility rates observed even with smaller penalty weights.

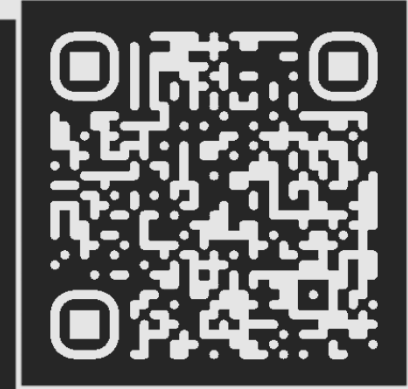


# Thank You

Paper



GitHub



Mechanical & Industrial Engineering  
**UNIVERSITY OF TORONTO**



**Pacific Northwest**  
NATIONAL LABORATORY