Differentiable integer linear programming

ICLR'25

Zijie Geng , Jie Wang, Xijun Li, Fangzhou Zhu, Jianye Hao, Bin Li, Feng Wu

ML for integer programming

Mixed integer linear programs (MILP):

- Flexible modeling tool for NP-hard combinatorial optimization
- E.g., scheduling, network design, …
- Solvers are powerful but very computationally expensive

Challenge of ML-based heuristics (e.g., last class)**:**

Supervision is expensive: requires solving NP-hard problems

This paper: unsupervised learning approach

via end-to-end differentiable pipeline

Overview of approach: DiffILO

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0, 1\}^n \end{array}$$

1. Relax to **probabilistic, continuous** equivalent form
2. Convert from **constrained** optimization to **unconstrained**
3. Reparameterize so objective is **differentiable** almost everywhere

1: Relax to probabilistic, continuous equivalent form

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

$$\downarrow$$

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \hat{\mathbf{x}} \\ \text{subject to} & \hat{\mathbf{x}} \in [0,1]^n \longrightarrow x_i \sim \text{Bernoulli}(\hat{x}_i) \\ & \mathbb{E}[\max\{A\mathbf{x} - \mathbf{b}, \mathbf{0}\}] = \mathbf{0} \end{array}$$

Justification of probabilistic form:

- **Thm 1** (informal)**:** top is feasible & solvable iff bottom is too
- **Thm 2** (informal)**:** opt solution of top ≡ (rounded) solutions of bottom

Overview of approach: DiffILO

$$\begin{array}{ll}\text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n\end{array}$$

1. Relax to **probabilistic, continuous** equivalent form
2. Convert from **constrained** optimization to **unconstrained**
3. Reparameterize so objective is **differentiable** almost everywhere

2: Convert from constrained to unconstrained

minimize $\mathbf{c}^\top \hat{\mathbf{x}}$
subject to $\hat{\mathbf{x}} \in [0,1]^n$
$\mathbb{E}[\max\{A\mathbf{x} - \mathbf{b}, \mathbf{0}\}] = \mathbf{0}$

$\mathbf{a}_j$: j[th] row of A

$\hat{\phi}_j(\hat{\mathbf{x}}) = \mathbb{E}_{\mathbf{x} \sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$: expected violation of j[th] constraint

Independent Bernoullis

minimize $\mathbf{c}^\top \hat{\mathbf{x}} + \mu \sum_{j=1}^{m} \hat{\phi}_j(\hat{\mathbf{x}})$

subject to $\hat{\mathbf{x}} \in [0,1]^n$

Overview of approach: DiffILO

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

1. Relax to **probabilistic, continuous** equivalent form
2. Convert from **constrained** optimization to **unconstrained**
3. Reparameterize so objective is **differentiable** almost everywhere

3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

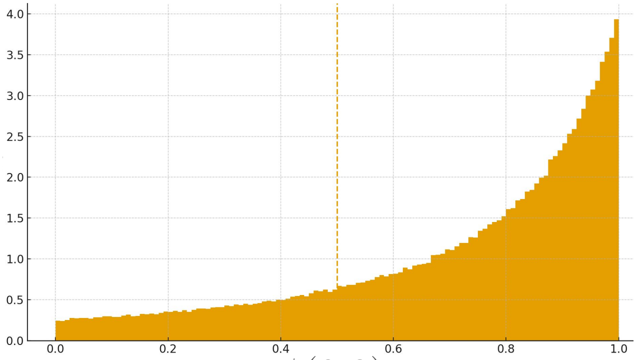
3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

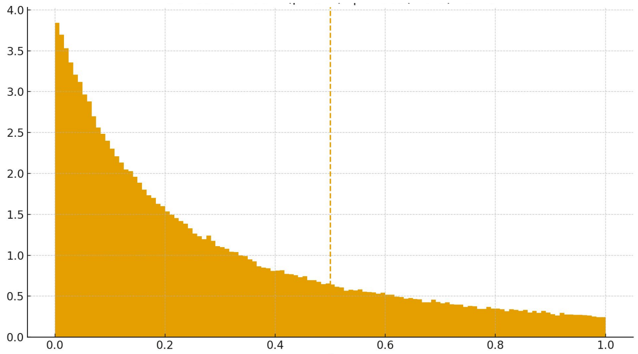
Very messy to differentiate!

Instead: relax to continuous RV $\xi(\hat{x}_i)$ such that $\mathbb{P}[\xi(\hat{x}_i) > 0.5] = \hat{x}_i$

Density



$\xi(0.8)$



$\xi(0.2)$

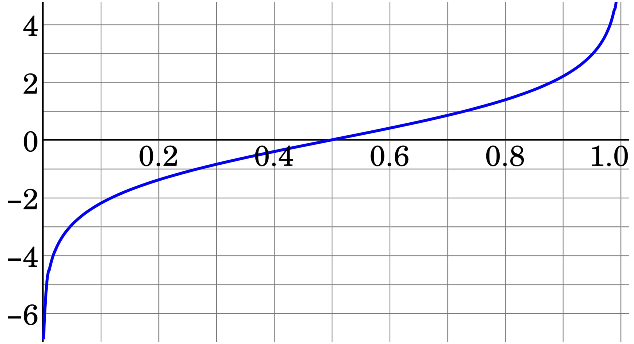
3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

Instead: relax to continuous RV $\xi(\hat{x}_i)$ such that $\mathbb{P}[\xi(\hat{x}_i) > 0.5] = \hat{x}_i$

1. Apply logit function $\tau(\hat{x}_i) = \log \frac{\hat{x}_i}{1 - \hat{x}_i}$ (inverse of sigmoid)



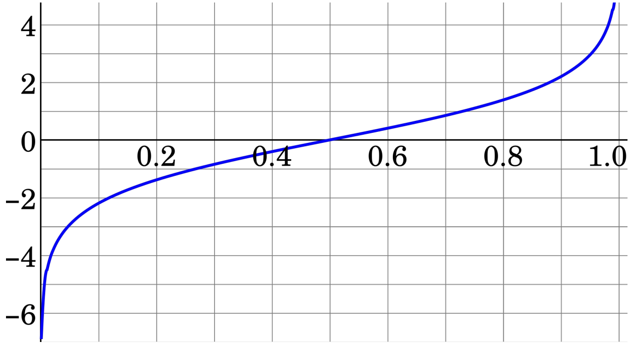
3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top\mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

Instead: relax to continuous RV $\xi(\hat{x}_i)$ such that $\mathbb{P}[\xi(\hat{x}_i) > 0.5] = \hat{x}_i$

1. Apply logit function $\tau(\hat{x}_i) = \log\frac{\hat{x}_i}{1-\hat{x}_i}$ (inverse of sigmoid)
2. Sample $\epsilon \sim U(0,1)$



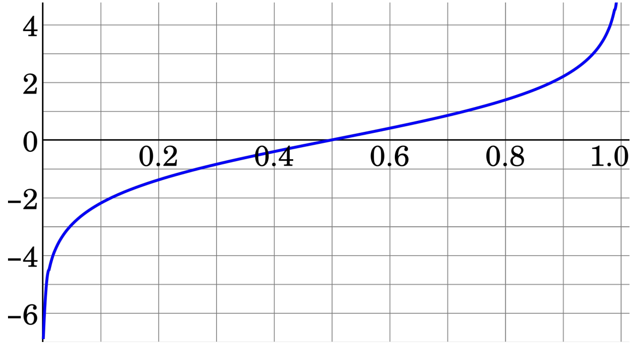
3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}} \hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}} \mathbb{E}_{\mathbf{x} \sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

Instead: relax to continuous RV $\xi(\hat{x}_i)$ such that $\mathbb{P}[\xi(\hat{x}_i) > 0.5] = \hat{x}_i$

1. Apply logit function $\tau(\hat{x}_i) = \log \dfrac{\hat{x}_i}{1 - \hat{x}_i}$ (inverse of sigmoid)
2. Sample $\epsilon \sim U(0,1)$
3. Perturb logit: $\tau(\hat{x}_i) + \tau(\epsilon)$



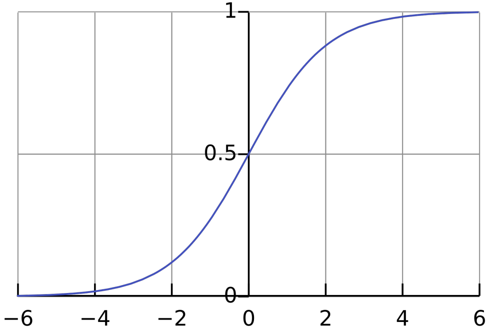
3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

Instead: relax to continuous RV $\xi(\hat{x}_i)$ such that $\mathbb{P}[\xi(\hat{x}_i) > 0.5] = \hat{x}_i$

1. Apply logit function $\tau(\hat{x}_i) = \log \frac{\hat{x}_i}{1-\hat{x}_i}$ (inverse of sigmoid)
2. Sample $\epsilon \sim U(0,1)$
3. Perturb logit: $\tau(\hat{x}_i) + \tau(\epsilon)$
4. Map back to (0,1): $\xi(\hat{x}_i;\epsilon) = \sigma(\tau(\hat{x}_i) + \tau(\epsilon))$



3: Reparameterize so objective is differentiable a.e.

Challenge to applying SGD: $\nabla_{\hat{\mathbf{x}}}\hat{\phi}_j(\hat{\mathbf{x}}) = \nabla_{\hat{\mathbf{x}}}\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}]$

Very messy to differentiate!

Surrogate that's differentiable almost everywhere:

$$\mathbb{E}_{\mathbf{x}\sim p(\cdot|\hat{\mathbf{x}})}[\max\{\mathbf{a}_j^\top \mathbf{x} - b_j, 0\}] \approx \mathbb{E}_{\epsilon}[\max\{\mathbf{a}_j^\top \xi(\hat{\mathbf{x}};\epsilon) - b_j, 0\}] := \hat{\varphi}_j(\hat{\mathbf{x}})$$

$$\begin{array}{ll}\text{minimize} & \mathbf{c}^\top\hat{\mathbf{x}} + \mu\sum_{j=1}^m \hat{\phi}_j(\hat{\mathbf{x}}) \\ \text{subject to} & \hat{\mathbf{x}} \in [0,1]^n\end{array} \longrightarrow \begin{array}{ll}\text{minimize} & \mathbf{c}^\top\hat{\mathbf{x}} + \mu\sum_{j=1}^m \hat{\varphi}_j(\hat{\mathbf{x}}) \\ \text{subject to} & \hat{\mathbf{x}} \in [0,1]^n\end{array}$$

Graph neural network

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

Represent IP with a constraint-variable bipartite graph $\mathcal{G}$ (like last class)

Graph neural network

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

Represent IP with a constraint-variable bipartite graph $\mathcal{G}$ (like last class)

GNN $f_\theta(\mathcal{G}) = \hat{\mathbf{x}} \in [0,1]^n$

Graph neural network

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

Represent IP with a constraint-variable bipartite graph $\mathcal{G}$ (like last class)

GNN $f_\theta(\mathcal{G}) = \hat{\mathbf{x}} \in [0,1]^n$

Loss function $\mathcal{L}(\theta; \mathcal{G}) = \mathbf{c}^\top f_\theta(\mathcal{G}) + \mu \sum_{j=1}^{m} \hat{\varphi}_j(f_\theta(\mathcal{G}))$

Inference

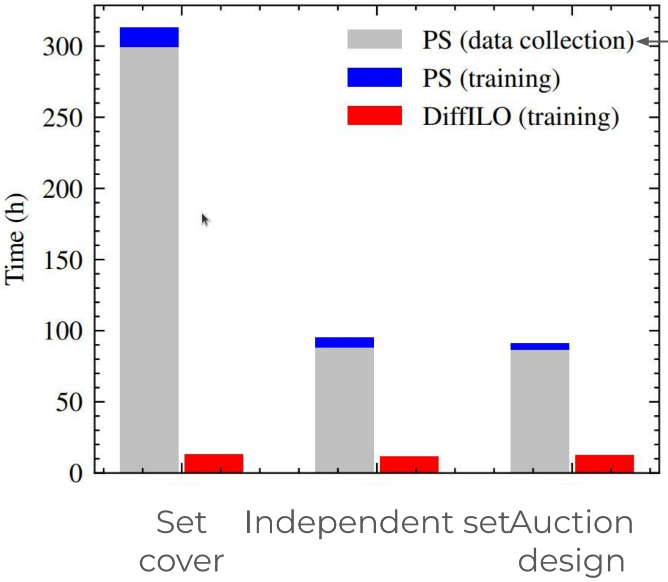
Sample from Bernoullis $\mathbf{x}' \sim p(\cdot \mid f_\theta(\mathcal{G}))$

Solve (e.g., with Gurobi):

$$
\begin{aligned}
\text{minimize} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\
& \sum_{i : x_i' = 0} x_i + \sum_{i : x_i' = 1} (1 - x_i) \leq \Delta \\
& \mathbf{x} \in \{0, 1\}^n
\end{aligned}
$$

Training time comparison

240 IPs for training, 60 for validation, 100 for testing



Similar to ConPaS (last class)
[Han et al., ICLR'23]

Objective values

Best known solution

	SC (min, BKS: 86.45)			IS (max, BKS:684.14)			CA (max, BKS:22272.55)		
	10s	100s	1000s	10s	100s	1000s	10s	100s	1000s
Gurobi	1031.39	87.09	86.52	682.02	684.12	684.13	22090.76	22242.58	22272.03
PS+Gurobi	131.87	125.26	125.26	**684.13**	**684.13**	684.13	22140.65	22243.12	22272.47
DiffILO+Gurobi	**95.65**	**86.78**	**86.48**	684.00	684.12	**684.14**	**22177.82**	**22260.48**	**22272.55**

Overview

- **Goal:** Learn to solve IPs without supervision or solver labels
  a. Reformulate discrete IP as continuous, probabilistic program
  b. Add exact penalty to remove constraints
  c. Apply relaxed Bernoulli for differentiable sampling
- Resulting objective differentiable almost everywhere
- Unsupervised: fast training
- Improves solver warm starts