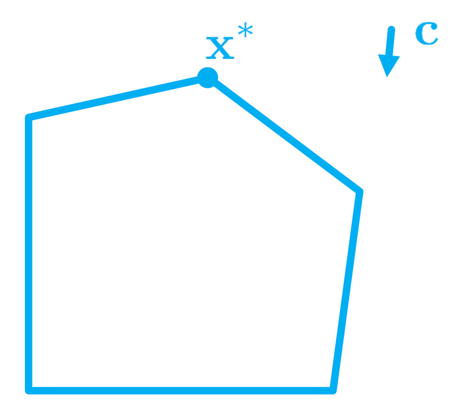
Learning Linear Programs from Optimal Decisions

Yingcong Tan[1] · Daria Terekhov[1] · Andrew Delong[2]

1. Mechanical, Industrial and Aerospace Engineering 2. Computer Science and Software Engineering
Gina Cody School of Engineering and Computer Science, Concordia University, Montréal Québec Canada

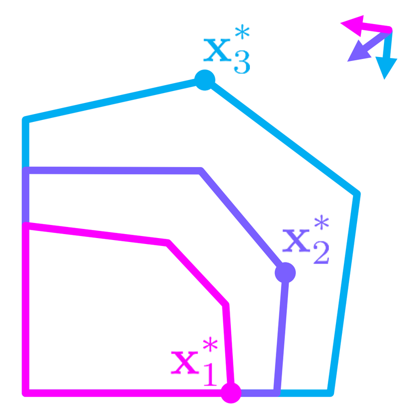
Linear Program

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \le \mathbf{b} \end{array}$$



Parametric Linear Program

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & \mathbf{c}(\mathbf{u})^T\mathbf{x} \\ \text{subject to} & \mathbf{A}(\mathbf{u})\mathbf{x} \le \mathbf{b}(\mathbf{u}) \end{array}$$



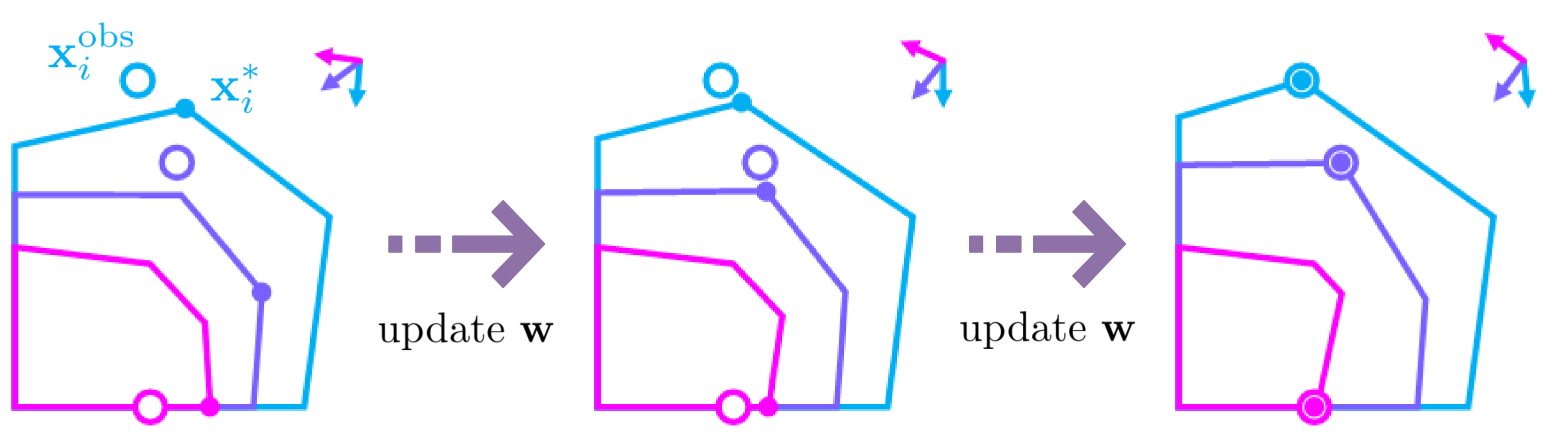
Learning a Parametric Linear Program

Hypothesis Define a suitable hypothesis space which is parameterized by $\mathbf{w} \in \mathbb{R}^K$.

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & \mathbf{c}(\mathbf{u},\mathbf{w})^T\mathbf{x} \\ \text{subject to} & \mathbf{A}(\mathbf{u},\mathbf{w})\mathbf{x} \le \mathbf{b}(\mathbf{u},\mathbf{w}) \end{array}$$

Input $(\mathbf{u}_i, \mathbf{x}_i^{\text{obs}}),\ i = 1, \dots, N$

Output Learn $\mathbf{w}$ to reduce the discrepancy between the prediction ($\mathbf{x}_i^*$) and observed solutions ($\mathbf{x}_i^{\text{obs}}$).



Novel Bilevel Formulation

Training Objective $\quad \min_{\mathbf{w} \in \mathcal{W}} \ \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{x}_i^*, \mathbf{x}_i^{\text{obs}}, \mathbf{u}_i, \mathbf{w}) + r(\mathbf{w})$

Outer Problem
$$\begin{array}{lll} \text{s.t.} & \mathbf{A}(\mathbf{u}_i,\mathbf{w})\mathbf{x}_i^{\text{obs}} \le \mathbf{b}(\mathbf{u}_i,\mathbf{w}), & i = 1,\dots,N \\ & \mathbf{G}(\mathbf{u}_i,\mathbf{w})\mathbf{x}_i^{\text{obs}} = \mathbf{h}(\mathbf{u}_i,\mathbf{w}), & i = 1,\dots,N \end{array}$$

Inner Problem
$$\mathbf{x}_i^* \in \operatorname*{argmin}_{\mathbf{x}} \left\{ \mathbf{c}(\mathbf{u}_i,\mathbf{w})^T\mathbf{x} \;\middle|\; \begin{array}{l} \mathbf{A}(\mathbf{u}_i,\mathbf{w})\mathbf{x} \le \mathbf{b}(\mathbf{u}_i,\mathbf{w}) \\ \mathbf{G}(\mathbf{u}_i,\mathbf{w})\mathbf{x} = \mathbf{h}(\mathbf{u}_i,\mathbf{w}) \end{array} \right\}, \quad i = 1,\dots,N$$

Example loss: *absolute objective error* (AOE), *squared decision error* (SDE)

$$\ell_{\text{AOE}} = |\,\mathbf{c}(\mathbf{u}_i,\ \mathbf{w})^T(\mathbf{x}_i^{\text{obs}} - \mathbf{x}_i^*)\,| \qquad \ell_{\text{SDE}} = \tfrac{1}{2}\|\mathbf{x}_i^{\text{obs}} - \mathbf{x}_i^*\|^2$$

Methodology

- Solve the bi-level formulation with *gradient-based non-linear programming* algorithms.
- In this paper, we use *sequential quadratic programming* **(SQP)** algorithms.
- Compute gradients by differentiating through an LP.

Methods for Differentiating through an LP

- **backprop**: backpropagate through the steps of a forward optimization algorithm [Tan et al., 2019].
- **implicit**: implicit differentiation procedure [Amos and Kolter, 2017] specialized to LP.
- **cvx**: use a *cvxpylayer* [Agrawal et al., 2019] for solving LP and for computing gradients.
- **direct**: closed-form expression of the gradients (applicable to AOE loss only).

Experiments

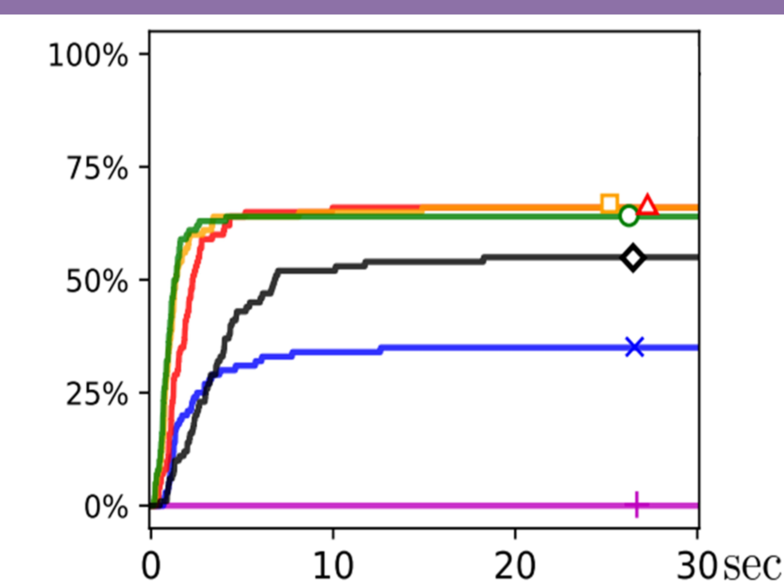
Experiment 1: Learn Parametric LPs

Learn $\mathbf{w}$ to minimize AOE loss.

Training Success @ AOE $\le 10^{-5}$.

gradient-free methods
+ Random Search × COBYLA

gradient-based methods
△ $\text{SQP}_{\text{bprop}}$ □ SQP_{impl}
◇ SQP_{cvx} ○ SQP_{dir}



Training result of experiment 1. This figure shows the percentage of instances reaching training loss of 10^{-5} over time.

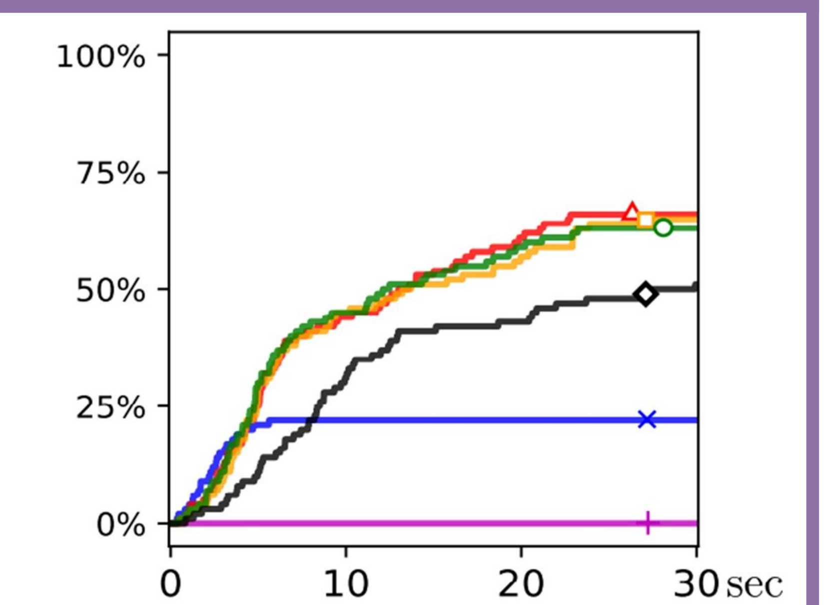
Experiment 2: Learn Multi-commodity Flow Problems

Learn $\mathbf{w}$ to minimize AOE loss.

Training Success @ AOE $\le 10^{-5}$.

gradient-free methods
+ Random Search × COBYLA

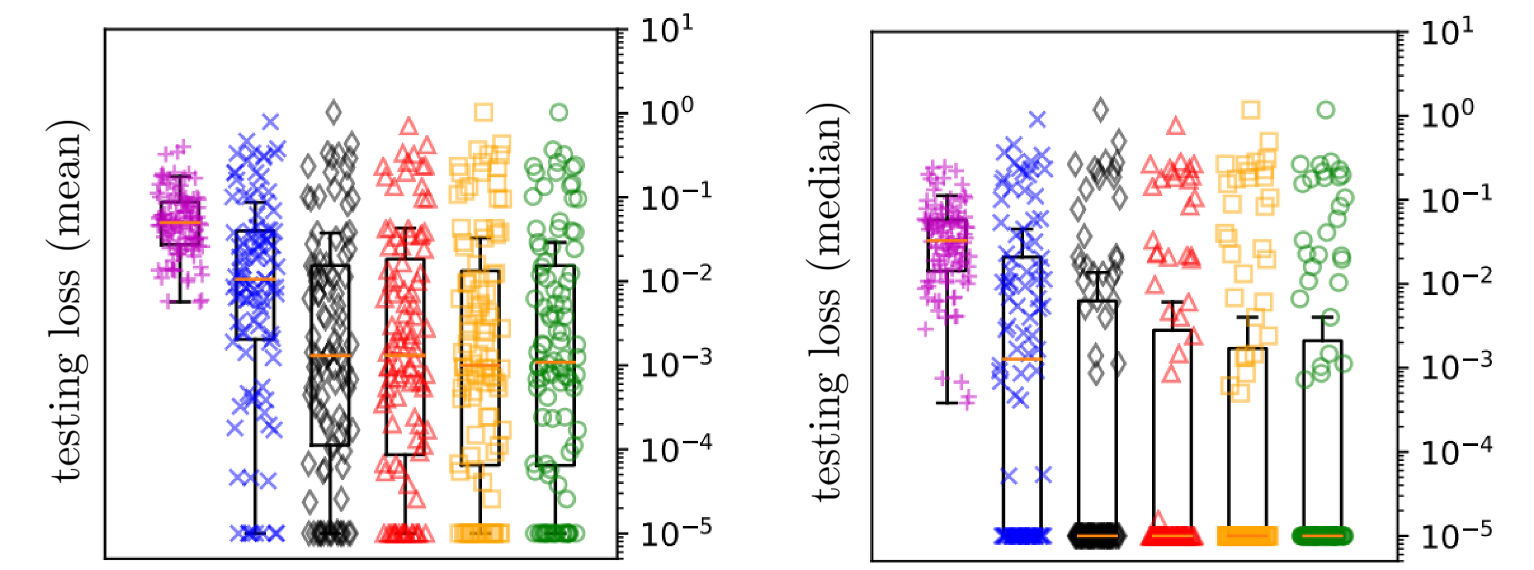
gradient-based methods
△ $\text{SQP}_{\text{bprop}}$ □ SQP_{impl}
◇ SQP_{cvx} ○ SQP_{dir}



Training result of experiment 1. This figure shows the percentage of instances reaching training loss of 10^{-5} over time.

Discussion

Challenge of Generalization

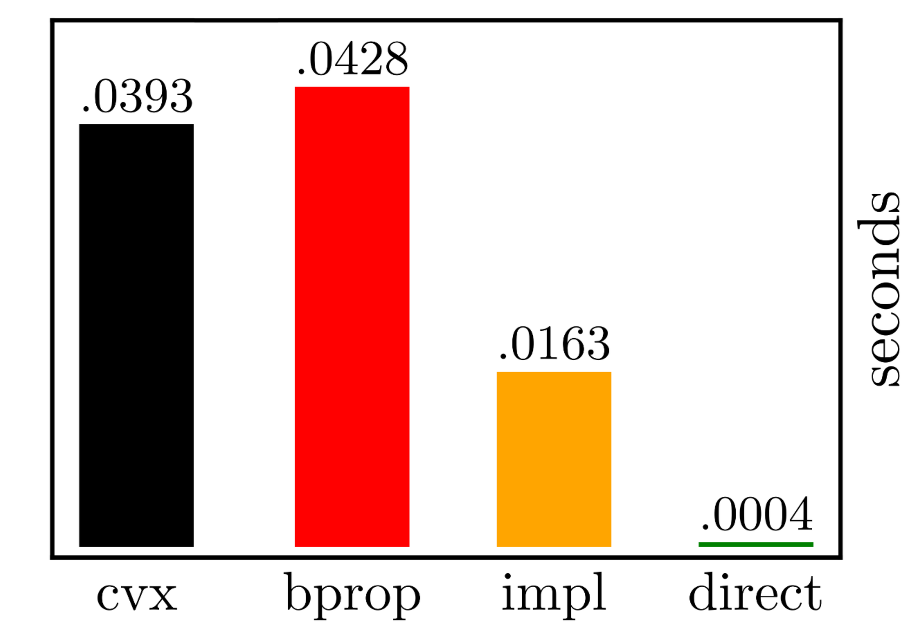


Testing loss on held-out data for experiment 2. Each marker represents the testing loss of the corresponding method over multiple observations for one instance.

gradient-free methods + Random Search × COBYLA

gradient-based methods ◇ SQP_{cvx} △ $\text{SQP}_{\text{bprop}}$ □ SQP_{impl} ○ SQP_{dir}

Runtime of Gradient Computations in Experiment 2



Runtime of gradient computations in experiment 2. Each bar represents the average runtime for one iteration of gradient computation over 100 random multi-commodity flow problem instances

Contributions

- ☑ Develop **the currently most general framework** for learning linear programs
  - Solve a **novel bi-level formulation** using **gradient-based non-linear programming algorithms**.
  - Provide a **closed-form expression for computing gradients** and demonstrates its **computation efficiency** over other existing methods.
- ☑ Show good performance on PLPs and multi-commodity flow problems.
- ☑ Code available on https://github.com/yingcongtan/ilop.

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In NeurIPS, pages 9562–9574, 2019.

Brandon Amos and J Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In ICML, PMLR 70, pages 136–145, 2017.

Yingcong Tan, Andrew Delong, and Daria Terekhov. Deep inverse optimization. In CPAIOR, pages 540–556, 2019.