NEURAL INFORMATION PROCESSING SYSTEMS

LANCER: Learning Decision Losses for Mathematical Optimization Under Partial Information






Arman Zharmagambetov Yuandong Tian Brandon Amos Aaron Ferber (USC) Taoan Huang (USC) Bistra Dilkina (USC)

Meta

USC

Background

Many real-world optimization problems consider settings that are nontrivial or extremely costly to solve.

Why? uncertainty in the problem formulation, intractability of the objective and/or constraints, etc.

Examples:

- Mixed-Integer Non-Linear Programming (MINLP).

  Non-linear (typically non-convex) objective with integer variables.
- Smart Predict+Optimize framework

  (a.k.a. decision-focused learning). Latent components of the optimization problem unknown.
- Model based Reinforcement Learning.

  optimal model design (OMD) to learn the dynamics model end-to-end with the policy objective
- etc...

A Unified Training Procedure

Consider opt. problem: $\min_{\mathbf{x}} f(\mathbf{x}; \mathbf{z}) \quad \text{s.t.} \quad \mathbf{x} \in \Omega$

Z - optimization problem descriptions (e.g., coefficients).

- In Smart Predict+Optimize: **Z** is unknown and must be derived from observable **Y**.
- In MINLP: **Z** is observable, but f is a general nonlinear objective and **X** is in discrete space.

A Unified Training Procedure

Instead, consider this: $$\min_{\boldsymbol{\theta}} \mathcal{L}(Y, Z) := \sum_{i=1}^{N} f\left(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{y}_i); \mathbf{z}_i\right) \qquad (1)$$

Over N training instances.

- In Smart Predict+Optimize : $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x}; \mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y}))$

where the goal is to learn mapping **C** (e.g., a neural net) so that a downstream solver outputs a high-quality solution (that minimizes f).

- In MINLP : $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\Omega} \mathbf{x}^\top \mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y})$

where the goal is to learn the coefficients **C** of the integer linear program so that the solution gives low objective f. **C** can be a vector or a model depending on the specific setting.

LANCER: Learning Landscape Surrogate Losses

- Compound function $f \circ g$ from Eqn. (1) is hard to optimize.
- Shortcomings of the existing methods:
  - Some methods are domain-specific (e.g., applies only to LP)
  - Requires $\nabla_x f(x)$ $\Longrightarrow$ Does not applicable with “black-box” functions;
  - Requires $\nabla_c g(c)$ $\Longrightarrow$ Tricky with discrete problems

- **Our proposal: learn surrogate loss $\mathcal{M}$ that approximates $f \circ \mathrm{g}$ and minimize $\mathcal{M}$ instead.**
- Although solver g can be hard to model, $f \circ \mathrm{g}$ is typically smooth and lies in continuous scalar space.

$$\min_{\boldsymbol{\theta}} \mathcal{L}(Y, Z) := \sum_{i=1}^{N} f\left(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{y}_i); \mathbf{z}_i\right)$$

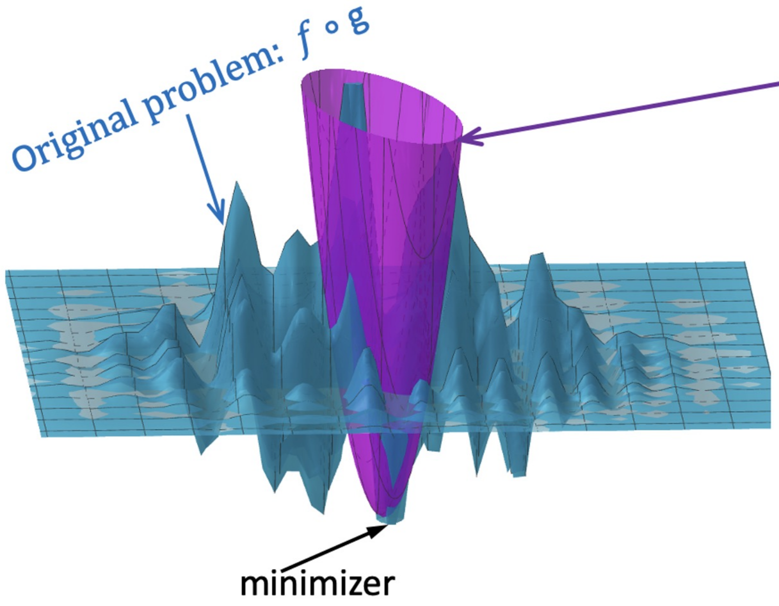
$$\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{y}) = \arg\min_{\mathbf{x} \in \Omega} \mathbf{x}^\top \mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y})$$

$$\longrightarrow \quad \min_{\boldsymbol{\theta}} \mathcal{M}(Y, Z) := \sum_{i=1}^{N} \mathcal{M}\left(\mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y}_i); \mathbf{z}_i\right).$$

How to learn surrogate loss *M*?

$$\min_{\mathbf{w}} \|\mathcal{M}_{\mathbf{w}}(\mathbf{c}_{\boldsymbol{\theta}^*}(\mathbf{y}_i), \mathbf{z}_i) - f\left(\mathbf{g}_{\boldsymbol{\theta}^*}(\mathbf{y}_i); \mathbf{z}_i\right)\|$$

$$\text{s.t.} \quad \boldsymbol{\theta}^* \in \text{argmin}_{\boldsymbol{\theta}} \mathcal{M}_{\mathbf{w}}(\mathbf{c}_{\theta}(\mathbf{y}_i), \mathbf{z}_i).$$



Landscape surrogate $\mathcal{M}$ →

learn c_θ:

$$\min_{\theta} \mathcal{M}(c_\theta(y), z)$$

← use (y, z, f) to refine $\mathcal{M}$

Input: $\mathcal{D}_{\text{train}} \leftarrow \{\mathbf{y}_i, \mathbf{z}_i\}_{i=1}^N$, solver $\mathbf{g}$, objective f, target model $\mathbf{c}_{\boldsymbol{\theta}}$

Initialize $\mathbf{c}_{\boldsymbol{\theta}}$ (e.g. random, warm start);

for $t = 1 \ldots T$ **do**

  • $\mathbf{w}$-step (fix $\boldsymbol{\theta}$ and optimize over $\mathbf{w}$):

    for $(\mathbf{y}_i, \mathbf{z}_i) \in \mathcal{D}_{\text{train}}$ **do**

      evaluate $\hat{\mathbf{c}}_i = \mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y}_i)$;

      evaluate $\hat{f}_i = f(\mathbf{g}(\hat{\mathbf{c}}_i); \mathbf{z}_i)$;

      add $(\hat{\mathbf{c}}_i, \mathbf{z}_i, \hat{f}_i)$ to $\mathcal{D}$;

    end for

    solve $\min_{\mathbf{w}} \sum_{i \in \mathcal{D}} \left\| \mathcal{M}_{\mathbf{w}}(\hat{\mathbf{c}}_i, \mathbf{z}_i) - \hat{f}_i \right\|$ via supervised lea

  • $\boldsymbol{\theta}$-step (fix $\mathbf{w}$ and optimize over $\boldsymbol{\theta}$):

    solve $\min_{\boldsymbol{\theta}} \sum_{i \in \mathcal{D}_{\text{train}}} \mathcal{M}_{\mathbf{w}}(\mathbf{c}_{\boldsymbol{\theta}}(\mathbf{y}_i), \mathbf{z}_i)$ via supervised lea

end for

Experiments: smart Predict+Optimize

Task: given input features ***y*** and model ***c***, predict optimization problem coefficients (e.g. edge weights for Shortest Path), solve the problem.

Learning problem: train predictor ***c*** such that the regret is minimized.

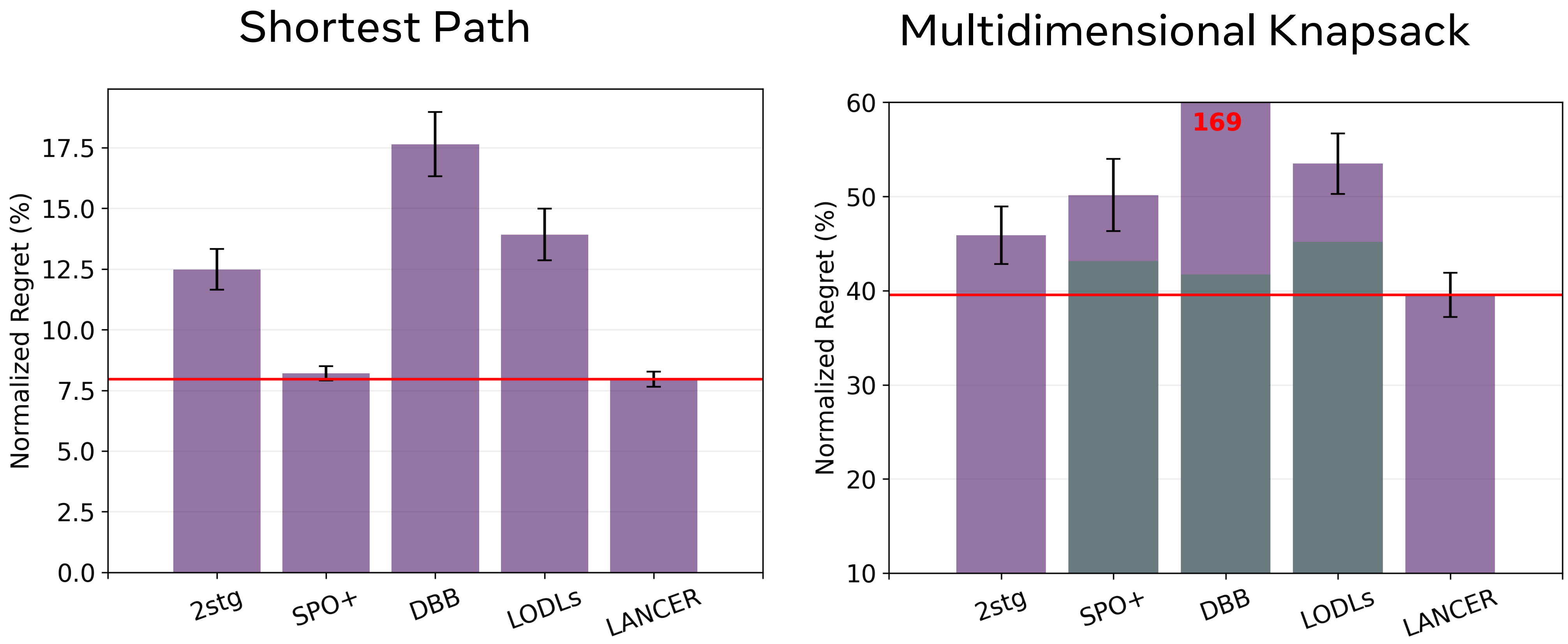
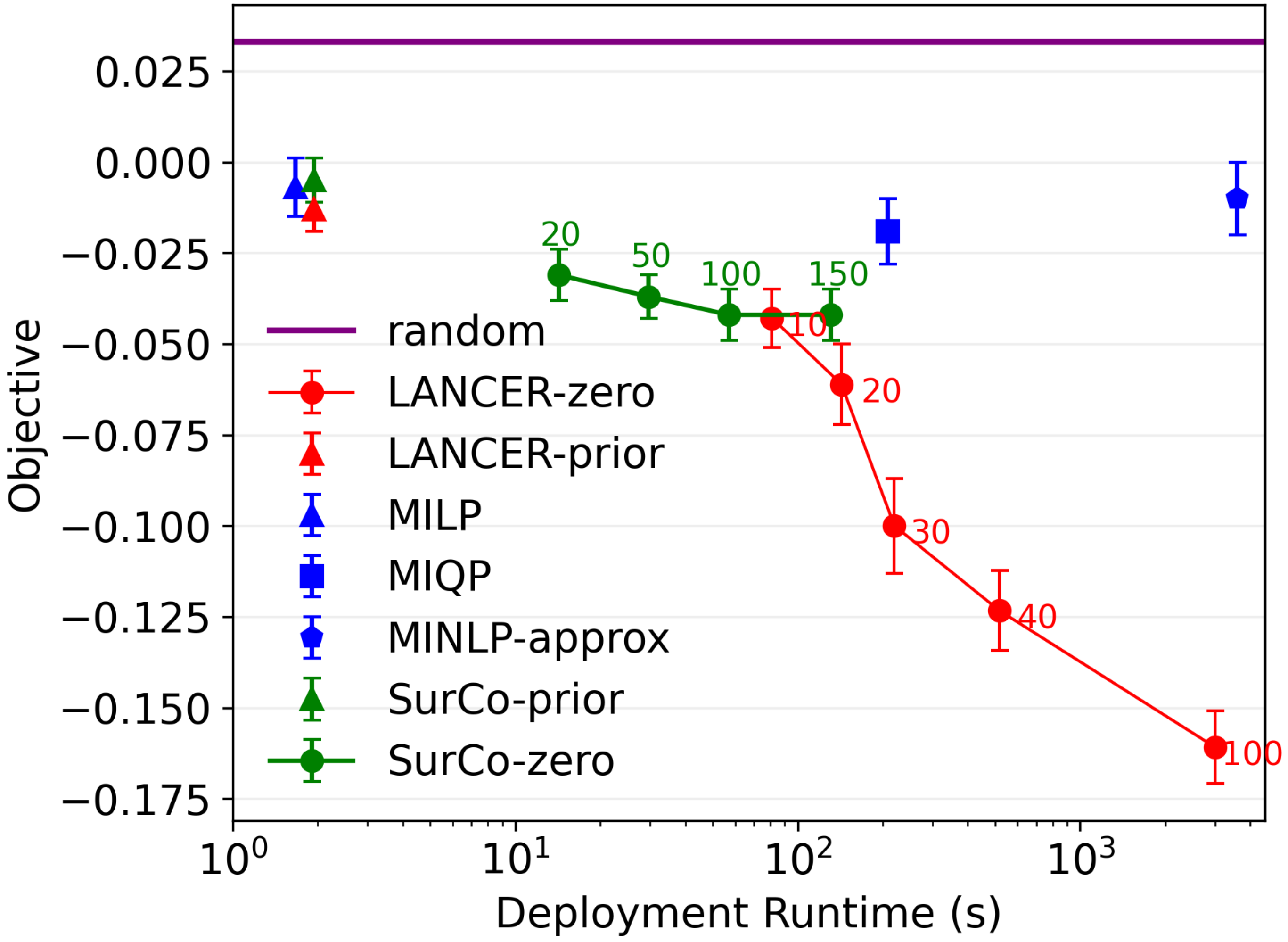


Figure 1. Normalized test regret (lower is better) for different P+O methods (x-axis). Overlaid dark green bars (right) indicate that the method warm started from the solution of 2stg.

Experiments: mixed-integer nonlinear program

- **Task:** Markowitz' portfolio selection problem but more complex objective and some variables are forced to be discrete. This is mixed-integer nonlinear program (MINLP).
- **Learning problem:** train surrogate MILP coefficients ***c*** such that the objective is minimized.
- **Dataset:** Historical data on market prices from QuandlWIKI.



Thank you!