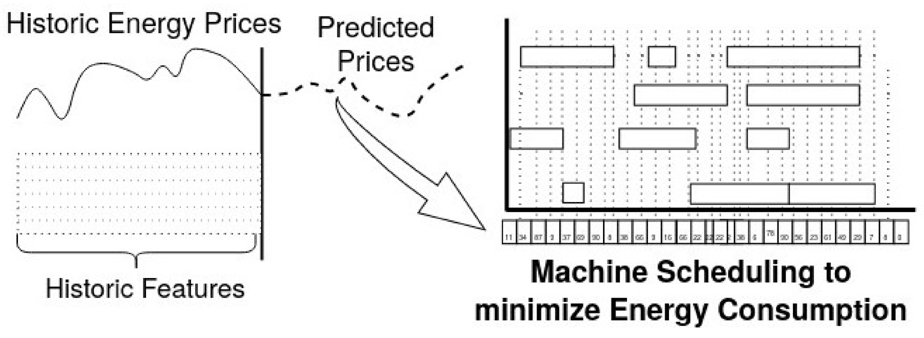
Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems

Jayanta Mandi,[1] Emir Demirović,[2] Peter. J Stuckey,[2] Tias Guns[1]

[1] Data Analytics Laboratory, Vrije Universiteit Brussel {jayanta.mandi,tias.guns}@vub.be

[2] University of Melbourne {emir.demirovic,pstuckey}@unimelb.edu.au

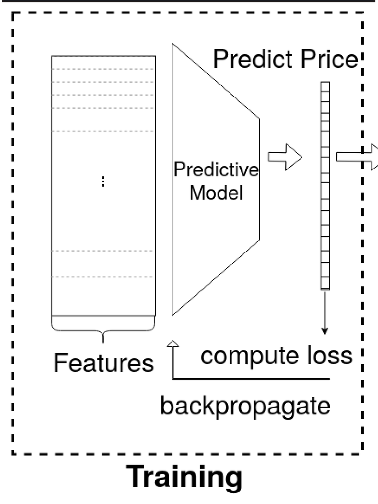
Predict-and-Optimize class of Problems



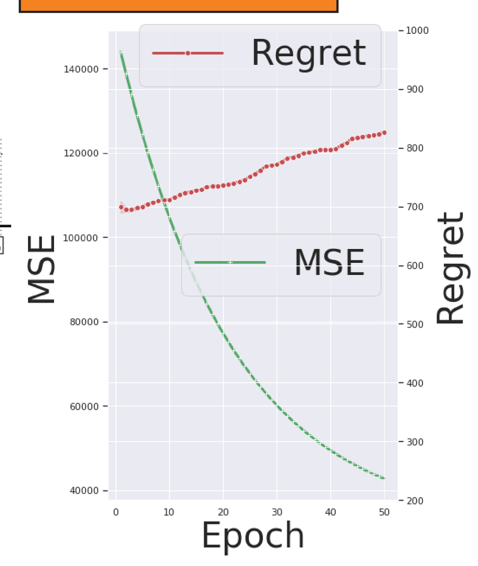
Parametric Optimization when some ***optimization parameters*** are ***not known*** at execution time.

Traditional Two-stage Approach

Implementation



Shortcomings



End-to-end Training

The downstream optimization is ***not*** in the model training

Ignores that the impact of prediction errors is ***not uniform throughout the underlying solution space***

Solution An *end-to-end training* minimizing a ***task-loss***- a measure of solution quality *after* the optimization task.

SPO[1]

For a Combinatorial Problem :

$$v^*(\theta) \equiv \arg\min_v f(v,\theta) \; s.t. \; C(v,\theta)$$

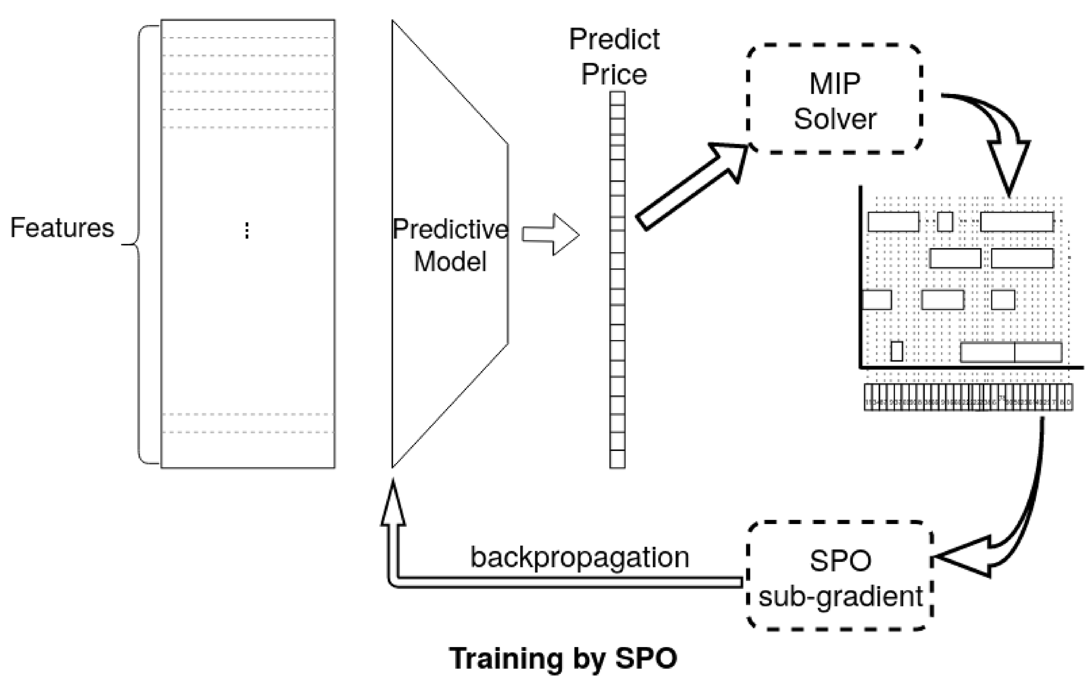
The task-loss is : $regret(\theta,\hat{\theta}) \equiv f(v^*(\hat{\theta}),\theta) - f(v^*(\theta),\theta)$.

If $\boldsymbol{regret(\theta,\hat{\theta})}$ is directly used as a task-loss, ***differentiate through argmin*** for backpropagation.

For a discrete output, the argmin is a ***piecewise constant*** function and ***non-differentiable***

SPO overcomes this by using a *convex upperbound of the regret.*

SPO framework



SPO Algorithm

repeat
- Sample N training datapoints
- **for** i *in* $1,...,N$ **do**
  - predict $\hat{\theta}_{u_i}$ using current ω
  - compute $v^*(2\hat{\theta}-\theta)$
  - $\nabla\mathcal{L}_i \leftarrow \boldsymbol{v^*(\theta) - v^*(2\hat{\theta}-\theta)}$ ***sub-gradient***
- **end**
- $\nabla\mathcal{L} = \frac{\sum_{i=1}^{N} \nabla\mathcal{L}_i}{N}$
- $\omega \leftarrow \omega - \alpha * \nabla\mathcal{L} * \frac{\partial\hat{\theta}_u}{\partial\omega}$;

until convergence;

Algorithm: SGD implementation of SPO

Challenge

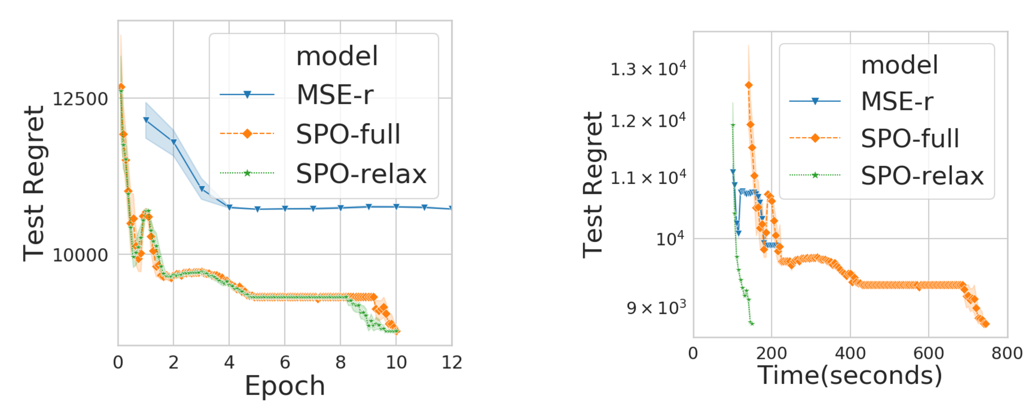
To compute the subgradient, $v^*(2\hat{\theta}-\theta)$ must be solved repeatedly for each training instance

High training time & computation-expensive

Relaxed Oracle

Solution A ***weak*** but fast yet accurate oracle

For MIP, the *relaxed oracle* is a weak oracle



(a) Epoch **(b)** Time

Relaxed Oracle helps in reducing training time without compromising quality

Warmstart using Earlier basis

Instance	SPO-relax	SPO-relax with Warmstart
1	6.5 (1.5) sec	**1.5** (0.2) sec
2	7 (1.5) sec	**1** (0.2) sec
3	10 (0.5) sec	**2.5** (0.1) sec

Table 1: Average and SD of per epoch runtime with and without warmstarting

Warmstarting the solver from basis is an effective strategy to speed up training

Large Scale Problem Instances

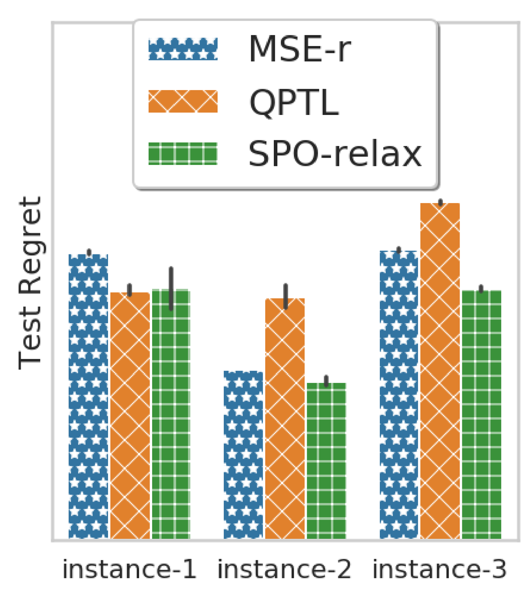
Hard Instances (200 tasks on 10 machines)	Two-stage Approach				SPO-relax		
	2 epochs	4 epochs	6 epochs	8 epochs	2 hour	4 hour	6 hour
instance I	90,769	88,952	**86,059**	86,464	**72,662**	74,572	79,990
instance II	128,067	124,450	124,280	**123,738**	120,800	**110,944**	114,800
instance III	129,761	128,400	122,956	**119,000**	108,748	**102,203**	112,970
instance IV	135,398	132,366	132,167	**126,755**	109,694	99,657	**97,351**
instance V	122,310	**120,949**	122,116	123,443	118,946	**116,960**	118,460

Table 2: Relaxed regret on hard ICON challenge[3] instances

SPO outperforms the two-stage approach on hard combinatorial problem instances even if it runs for limited epochs

Comparison with Decision-Focused Learning[2]

Decision-Focused Learning computes the gradient using a differentiable QP solver



SPO provides solution equal to or better than the Decision Focused QP

Contribution

- We propose an *end-to-end training and optimize* approach applicable to *large-scale combinatorial problem instances.*
- We show a *relaxed oracle* is good enough for computing SPO subgradient.
- We show *warmstarting using the basis of earlier solutions* is effective to speedup training.

References

[1] Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize".

[2] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI-19*.

[3] CSPLib problem 059: Energy-cost aware scheduling. `http://www.csplib.org/Problems/prob059`.

[4] Emir Demirović, Peter J. Stuckey, James Bailey, Jeffrey Chan, Chris Leckie, Kotagiri Ramamohanarao, and Tias Guns. An investigation into prediction + optimisation for the knapsack problem. In *CPAIOR-19*.