Structured Reinforcement Learning

Heiko Hoppe[1], Léo Baty[2], Louis Bouvier[2], Axel Parmentier[2], Maximilian Schiffer[1]

[1] Technical University of Munich, Germany

[2] École des Ponts, France

XVII[th] International Conference on Stochastic Programming

Champs-sur-Marne, 01.08.2025

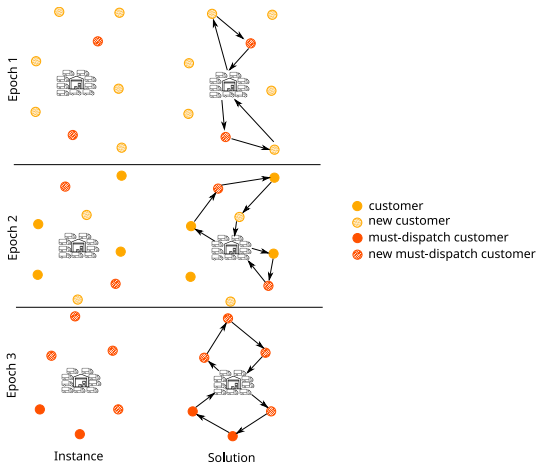
Background: stochastic multi-stage optimization problems

Stochastic multi-stage optimization problem settings

- **Common problems** in many operations research applications
- **Examples include:** vehicle routing, inventory planning, machine scheduling, assortment optimization
- **High importance** for real-world decision-making

Challenges of such problems

- **Multi-stage problems** require online decision-making
- **Contextual stochasticity** needs to be accounted for in decisions
- **Combinatorial constraints** necessitate use of appropriate algorithmic structure



Previous work: approaches to solve stochastic multi-stage optimization problems

(Smart) predict-then-optimize

Predict-then-optimize algorithms separate learning and optimization[1,2]

Smart predict-then-optimize algorithms require knowing true predictions[3,4]

Knowledge of **true prediction outcomes** required

Structured learning

Imitation learning algorithms require access to expert trajectories[5,6]

Empirical cost minimization algorithms only consider single/two-stage problems[7,8]

Offline expert solutions or single-stage problems required

Reinforcement learning

Neural CO struggles to ensure feasibility of actions in combinatorial problems[9,10]

Hybrid RL considers CO during inference, but not during training[11,12]

Gradient instability or problem-specific neural networks required

Lack of a **stable algorithm** to address stochastic multi-stage optimization problems **using collected experience only**

[1] Alonso-Mora et al. 2017, [2] Bertsimas and Kallus 2020, [3] Elmachtoub and Grigas 2022, [4] Mandi et al. 2020, [5] Parmentier 2022, [6] Baty et al. 2024, [7] Dalle et al. 2022, [8] Bouvier et al. 2025, [9] Bello et al. 2017, [10] Hottung and Tierney 2022, [11] Enders et al. 2023, [12] Hoppe et al. 2024

Aims and scope

In this talk, we...

- ...present **CO-augmented ML-pipelines** as a novel architecture for reinforcement learning
- ...motivate the use of **Fenchel-Young losses** for Structured Reinforcement Learning
- ...develop a **novel RL algorithm** to solve combinatorial problems
- ...test the performance of Structured RL on several **industrial problem settings**

A **well-performing** novel **RL algorithm** using structured learning for combinatorial problems

Problem setting: Combinatorial Markov Decision Processes

Combinatorial Markov Decision Processes:

- states $s \in S$
- actions $a \in \mathcal{A}(s) \subset \mathbb{R}^{d(s)}$
- $\mathcal{A}(s)$ **is the feasible solution space** of a combinatorial problem
- rewards r
- transitions $\mathbb{P}(s', r \mid s, a)$
- We have the same goal as most RL settings: **find a policy** π^* **maximizing** the expected **discounted return:**

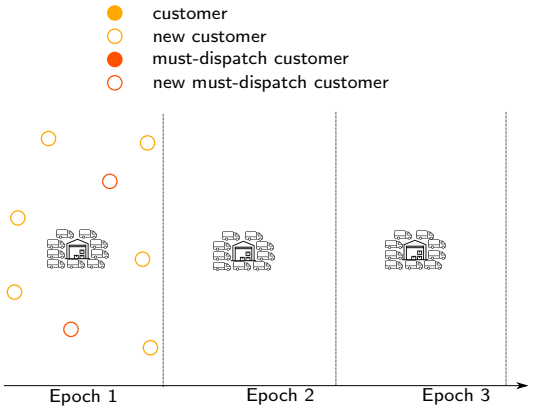
$$\pi^* \in \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

We consider Markov Decision Processes where the **action space** $\mathcal{A}(s)$ has **combinatorial structure**

Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

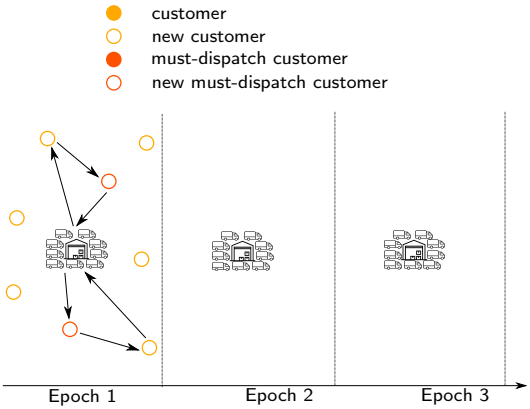
- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

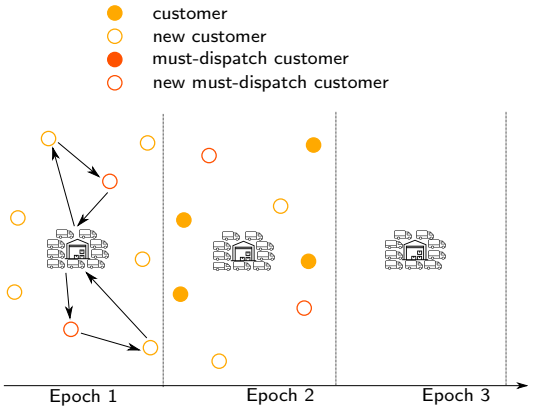
- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

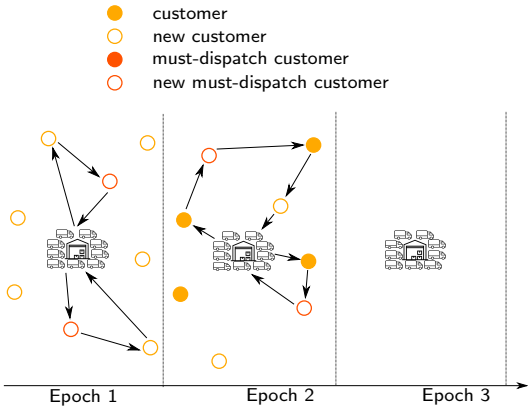
- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

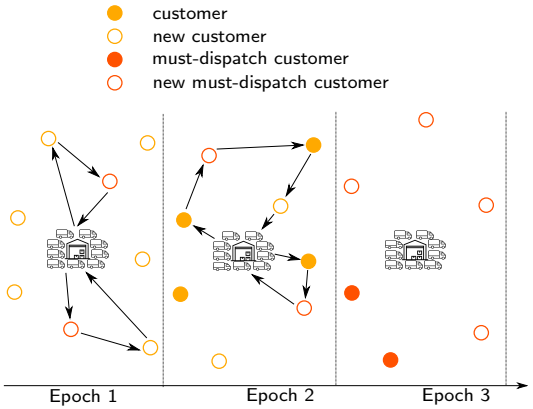
- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

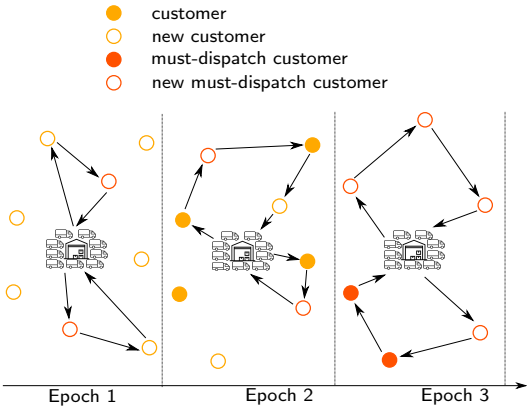
- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



Exemplary problem setting: dynamic vehicle routing with fixed delivery times

Dynamic vehicle scheduling problem:

- **Customers** appear dynamically over time
- Each customer has a **location** and a service **time point**
- We can **dispatch** a vehicle to serve the customer or **postpone** serving the customer
- Customers with service time points in the near future **have to be dispatched**
- **First decision:** Dispatch vs. postpone customers
- **Second decision:** Route vehicles to dispatch customers



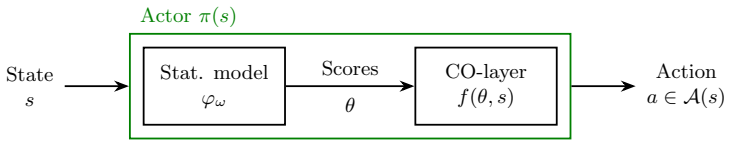
CO-augmented ML-pipelines

Model architectures for Combinatorial MDPs

- Combinatorial MDPs require special model architectures
- **Neural networks** can handle stochasticity, but struggle with combinatorial constraints
- **CO** can address combinatorial actions, but struggles to handle contextual stochasticity

CO-augmented ML-pipelines

- Stat. model estimates scores $\theta = \varphi_w(s)$, CO-layer generates actions $a = f(\theta, s) \in \operatorname{argmax}_{\tilde{a} \in \mathcal{A}(s)} : \theta^\top \tilde{a}$
- **Statistical model** φ_w allows generalization over states and anticipation of C-MDP dynamics
- **CO-layer** f enforces combinatorial feasibility and improves scalability



Training COAML-pipelines

Imitation learning

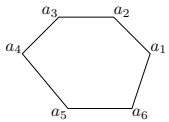
Requires access to expert policy $\bar{\pi}$ with actions $\bar{a}$

Reinforcement learning

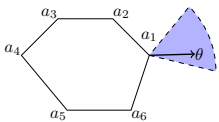
Only requires access to reward signal r given action a

Common problem: gradient instability

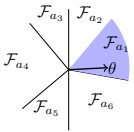
- Action space $\mathcal{A}(s)$ is **discrete and combinatorial,** its convex hull $\mathcal{C}(s) := \text{conv}(\mathcal{A}(s))$ forms a polytope
- CO-layer f maps continuous score vectors θ to discrete actions $a \in \mathcal{A}(s)$
- **Differentiating the actor is challenging:** the CO-layer is piecewise constant with respect to the action space
- Gradients are either zero if θ does not cross a cone boundary
- Or **the CO-layer causes jumps** between actions if θ crosses a cone boundary



Action polytope



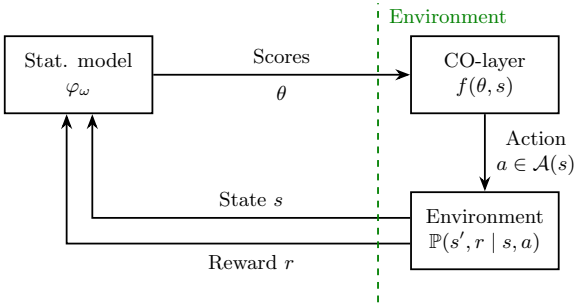
Normal cone



Cone in dual space

Deficiencies of ordinary RL gradients for COAML-pipelines

- In the reinforcement learning setting, we can treat θ as an action and the CO-layer f as part of the environment
- This enables **applying ordinary RL gradients** to the statistical model φ_w
- Since the CO-layer is piecewise constant, these gradients exhibit high variance
- **This hinders or prevents convergence**



Fenchel-Young losses as a suitable alternative in structured settings

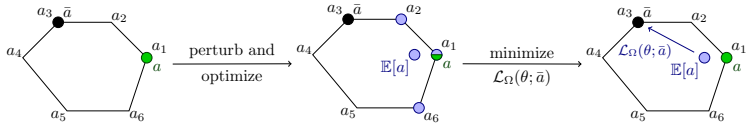
- **Fenchel-Young losses** offer a solution to non-smooth and non-convex gradient estimates
- Proposed by Blondel et al. 2020, they have been used successfully by Parmentier 2022, Baty et al. 2024, Jungel et al. 2024
- Given target action $\bar{a}$, they address the supervised learning problem of empirical risk minimization

$$\min_{\theta} \mathcal{L}(\theta; \bar{a}) = \min_{\theta} \sum_{j} \max_{\tilde{a} \in \mathcal{A}(s_j)} \theta^\top \tilde{a} - \theta^\top \bar{a}_j$$

- **By introducing a regularized CO-layer** using a Gaussian perturbation $Z \sim \mathbb{N}(0, \epsilon)$, we receive

$$\min_{\theta} \mathcal{L}_\Omega(\theta; \bar{a})^1 = \min_{\theta} \sum_{j} \mathbb{E}\Big[\max_{\tilde{a} \in \mathcal{A}(s_j)} (\theta + Z)^\top \tilde{a} \Big] - \theta^\top \bar{a}_j$$

- **The resulting gradient** is smooth and convex and enables stable backpropagation



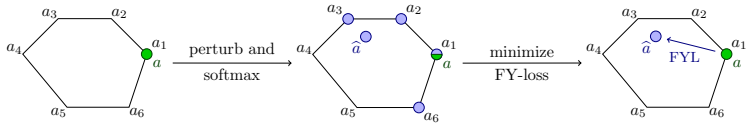
[1] Ω is based on a definition of Fenchel-Young losses using a regularization function $\Omega : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$

Algorithm training using online updates and Fenchel-Young losses

- Fenchel-Young losses require a target action $\bar{a}$, which is not given in an RL setting
- We **need to estimate a target action online:** we perturb θ and sample several η
- **We solve** $f(\eta, s)$ for each η to generate candidate actions a', which we evaluate using a critic as $Q_{\psi_\beta}(s, a')$
- We **estimate the target action** $\widehat{a}$ using a softmax:

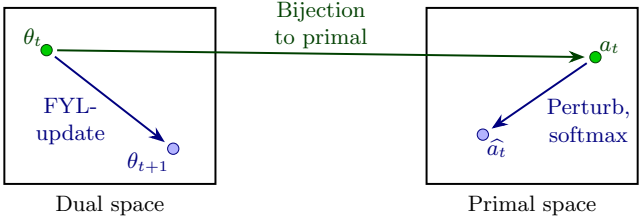
$$\widehat{a} = \underset{a'}{\text{softmax}}\left(\frac{1}{\tau}\, Q_{\psi_\beta}\right) = \sum_{a'} a' \frac{\exp\left(\frac{1}{\tau} \cdot Q_{\psi_\beta}(s, a')\right)}{\sum_{a'} \exp\left(\frac{1}{\tau} \cdot Q_{\psi_\beta}(s, a')\right)}$$

and update the actor by **minimizing the Fenchel-Young loss** between θ and $\widehat{a}$

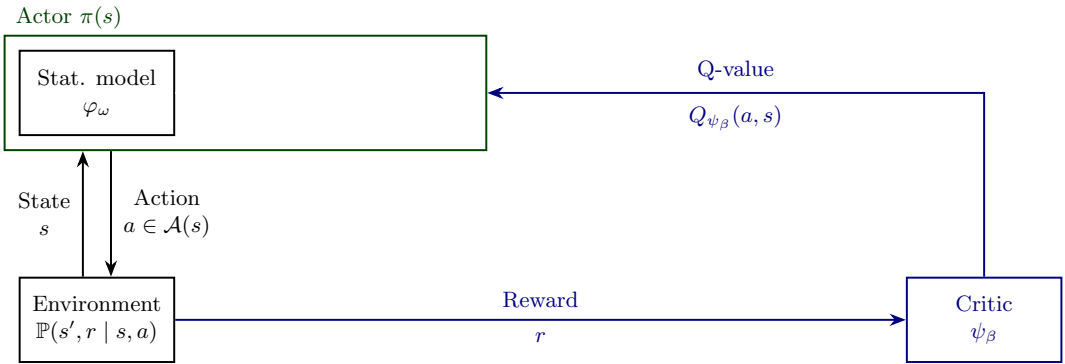


Background: primal-dual RL update step

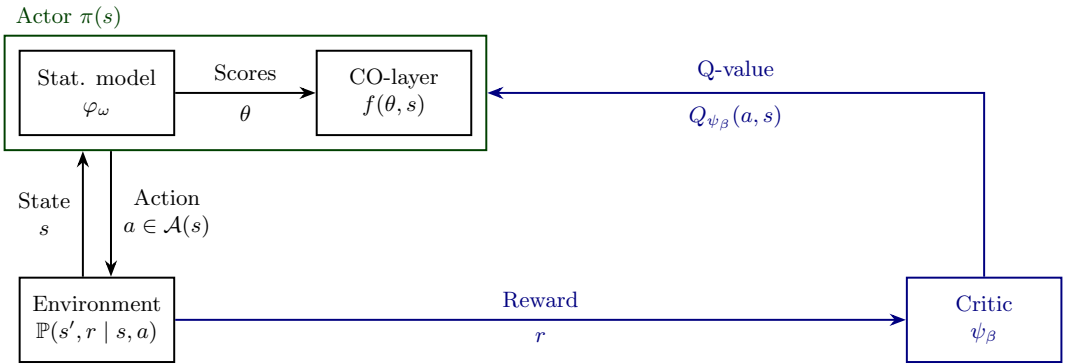
- The **dual space** is the space of scores θ, the **primal space** is the action space $\mathcal{A}(s)$
- We **use a bijection** between θ in the dual and a in the primal space
- We **search for a target action** $\widehat{a}$ in the primal space via perturbation, sampling, and the softmax
- **The dual update** is minimizing a Fenchel-Young loss between θ and $\widehat{a}$
- This is a **sampling-based extension of Mirror Descend** algorithms (Bouvier et al. 2025)



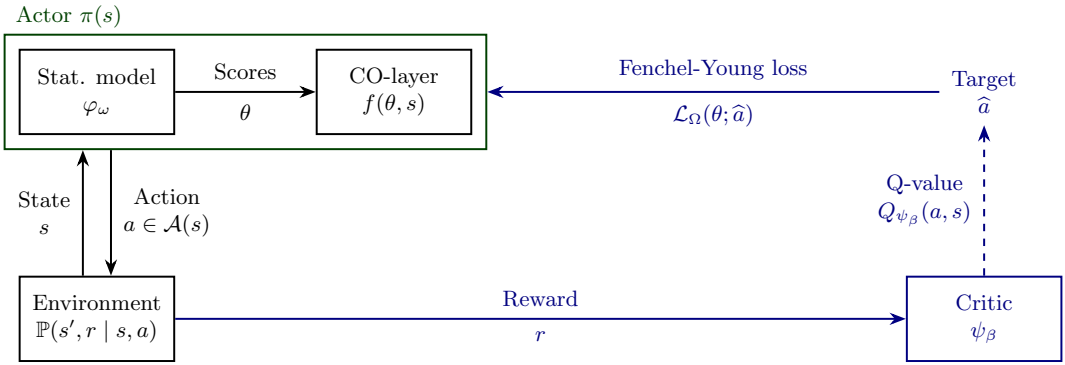
Overview over the Structured Reinforcement Learning algorithm



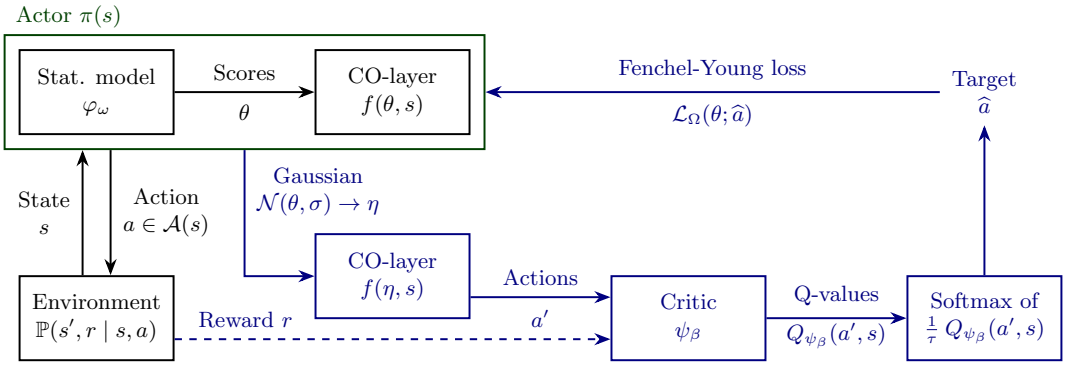
Overview over the Structured Reinforcement Learning algorithm



Overview over the Structured Reinforcement Learning algorithm



Overview over the Structured Reinforcement Learning algorithm



Experimental setup

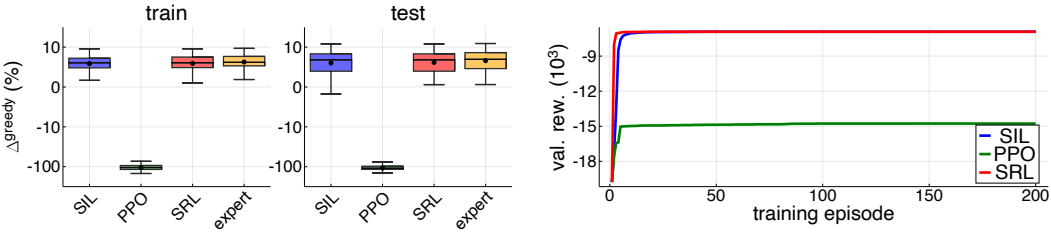
Studied environments

- **Six industrial problem settings** derived from related literature
- **Single-stage problems:** Warcraft Shortest Paths Problem, Single Machine Scheduling Problem, Stochastic Vehicle Scheduling Problem
- **Multi-stage problems:** Dynamic Vehicle Scheduling Problem, Dynamic Assortment Problem, Gridworld Shortest Paths Problem

Experimental setup

- **We compare SRL** to Structured Imitation Learning (SIL), and the unstructured RL algorithm Proximal Policy Optimization (PPO)
- All algorithms share identical COAML-pipelines
- We also estimate an expert and a greedy solution

Results: single-stage Stochastic Vehicle Scheduling Problem

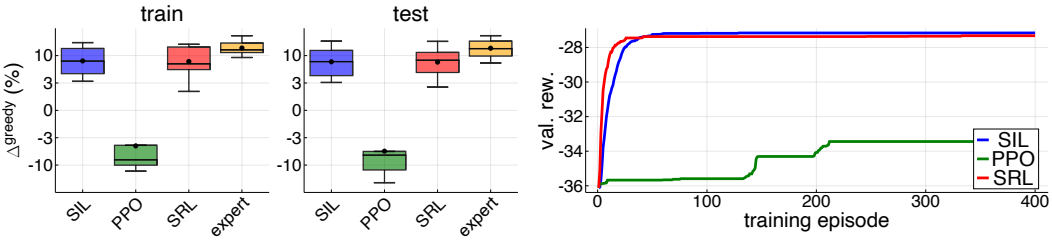


Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	0.2	10.0	0.1
Stdev. testing	0.0	10.0	0.0

SRL **outperforms Unstructured RL**, it performs **as good as Structured IL** and the expert policy

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Results: multi-stage Dynamic Vehicle Scheduling Problem

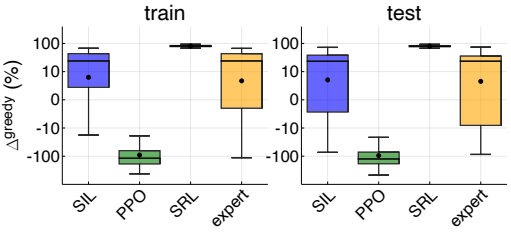


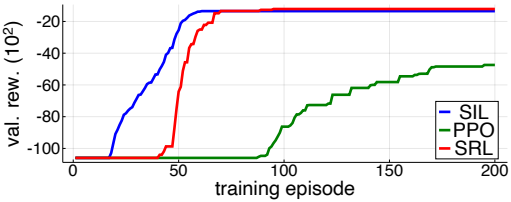
Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	0.3	5.8	0.3
Stdev. testing	0.4	5.6	0.3

SRL **outperforms Unstructured RL**, it performs **as good as Structured IL**

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Results: multi-stage Gridworld Shortest Paths Problem





Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	39.3	105.8	72.1
Stdev. testing	1.1	47.0	0.6

SRL **outperforms Unstructured RL and Structured IL** in this highly complicated problem setting

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Conclusion & future research

Summary of the presented work

We show...

- ...that **CO-augmented ML-pipelines** can generalize across states and ensure combinatorial feasibility well
- ...how to **update the actor model** of a COAML-pipeline using Fenchel-Young losses
- ...how to **find target actions** for the update by perturbation, sampling, and a Q-based softmax
- ...that the **proposed Structured RL algorithm** outperforms previous algorithms on six industrial benchmarks

Possible directions for future research

- **Test SRL** on real-world industrial problem settings
- Enhance computational efficiency of the SRL algorithm
- **Extend structured learning paradigms** to classical reinforcement learning algorithms

References I

Alonso-Mora, Javier, Alex Wallar, and Daniela Rus (Sept. 2017). "Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing". In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3583–3590. DOI: 10.1109/IROS.2017.8206203. (Visited on 05/08/2024).

Baty, Léo et al. (July 2024). "Combinatorial Optimization-Enriched Machine Learning to Solve the Dynamic Vehicle Routing Problem with Time Windows". In: Transportation Science 58.4, pp. 708–725. ISSN: 0041-1655, 1526-5447. DOI: 10.1287/trsc.2023.0107. (Visited on 04/22/2025).

Bello, Irwan et al. (2017). "Neural Combinatorial Optimization with Reinforcement Learning". In: International Conference on Learning Representations 2017 (ICLR) – Workshop Track.

Bertsimas, Dimitris and Nathan Kallus (Mar. 2020). "From Predictive to Prescriptive Analytics". In: Management Science 66.3, pp. 1025–1044. ISSN: 0025-1909, 1526-5501. DOI: 10.1287/mnsc.2018.3253. (Visited on 04/21/2025).

Blondel, Mathieu, André F.T. Martins, and Vlad Niculae (2020). "Learning with Fenchel-Young Losses". In: Journal of Machine Learning Research 21.35, pp. 1–69.

Bouvier, Louis et al. (May 2025). Primal-Dual Algorithm for Contextual Stochastic Combinatorial Optimization. DOI: 10.48550/arXiv.2505.04757. arXiv: 2505.04757 [cs]. (Visited on 05/12/2025).

Dalle, Guillaume et al. (Dec. 2022). Learning with Combinatorial Optimization Layers: A Probabilistic Approach. arXiv: 2207.13513 [cs, math, stat]. (Visited on 04/25/2024).

References II

Elmachtoub, Adam N. and Paul Grigas (Jan. 2022). "Smart "Predict, Then Optimize"". In: Management Science 68.1, pp. 9–26. ISSN: 0025-1909, 1526-5501. DOI: 10.1287/mnsc.2020.3922. (Visited on 04/21/2025).

Enders, Tobias et al. (June 2023). "Hybrid Multi-agent Deep Reinforcement Learning for Autonomous Mobility on Demand Systems". In: Proceedings of Machine Learning Research. Vol. 211. PMLR, pp. 1284–1296. (Visited on 12/15/2024).

Hoppe, Heiko et al. (July 2024). "Global Rewards in Multi-Agent Deep Reinforcement Learning for Autonomous Mobility on Demand Systems". In: Proceedings of Machine Learning Research. Vol. 242. PMLR, pp. 260–272. (Visited on 12/15/2024).

Hottung, André and Kevin Tierney (Dec. 2022). "Neural Large Neighborhood Search for Routing Problems". In: Artificial Intelligence 313, p. 103786. ISSN: 00043702. DOI: 10.1016/j.artint.2022.103786. (Visited on 05/09/2025).

Jungel, Kai et al. (Feb. 2024). Learning-Based Online Optimization for Autonomous Mobility-on-Demand Fleet Control. arXiv: 2302.03963 [cs, math]. (Visited on 04/25/2024).

Mandi, Jayanta et al. (Apr. 2020). "Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems". In: Proceedings of the AAAI Conference on Artificial Intelligence 34.02, pp. 1603–1610. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i02.5521. (Visited on 04/29/2024).

References III

Parmentier, Axel (Jan. 2022). "Learning to Approximate Industrial Problems by Operations Research Classic Problems". In: Operations Research 70.1, pp. 606–623. ISSN: 0030-364X. DOI: 10.1287/opre.2020.2094. (Visited on 04/25/2024).

Methodology: definition of Fenchel-Young losses via regularization

Given a regularization function $\Omega : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ and its Fenchel conjugate Ω^*, the Fenchel-Young loss $\mathcal{L}_\Omega(\theta; \bar{a})$ generated by Ω is defined over $\mathrm{dom}(\Omega^*) \times \mathrm{dom}(\Omega)$ as

$$\mathcal{L}_\Omega(\theta; \bar{a}) := \Omega^*(\theta) + \Omega(\bar{a}) - \langle \theta | \bar{a} \rangle = \sup_{a \in \mathrm{dom}(\Omega)} (\langle \theta | a \rangle - \Omega(a)) - (\langle \theta | \bar{a} \rangle - \Omega(\bar{a})).$$

For a given $\theta \in \mathrm{dom}(\Omega^*)$, we introduce the regularized prediction as $\sup_{a \in \mathrm{dom}(\Omega)} \langle \theta | a \rangle - \Omega(a)$.

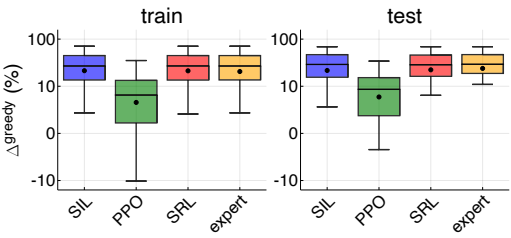
The Fenchel-Young loss measures the non-optimality of $\bar{a} \in \mathrm{dom}(\Omega)$ as a solution of the regularized prediction problem.

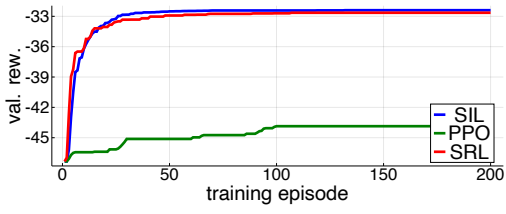
It is nonnegative and convex in θ.

If in addition Ω is proper, convex, and lower semi-continuous, $\mathcal{L}_\Omega$ reaches zero if and only if $\bar{a}$ is a solution of the regularized prediction problem.

Over the last years, Fenchel-Young losses have become the main approach for supervised training of structured policies.

Results: single-stage Warcraft Shortest Paths Problem

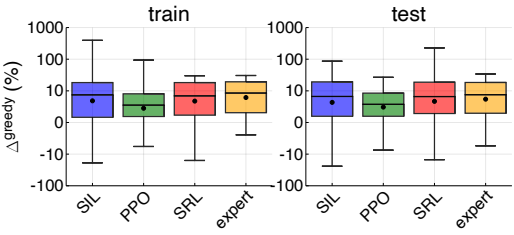


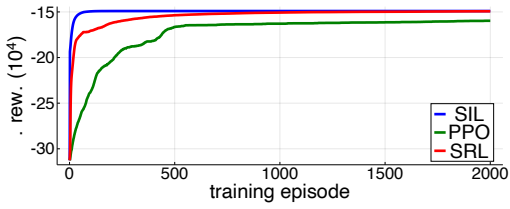


Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	0.2	3.8	0.5
Stdev. testing	0.6	5.6	1.0

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Results: single-stage Single Machine Scheduling Problem

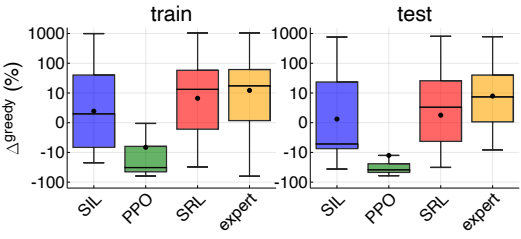


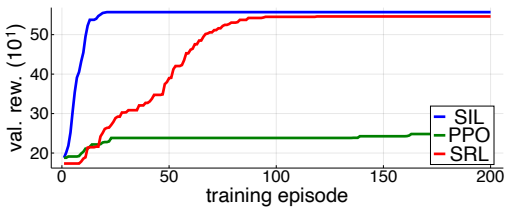


Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	0.0	1.9	0.1
Stdev. testing	0.0	0.3	0.0

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Results: multi-stage Dynamic Assortment Problem





Stability metric	Structured IL	Unstructured RL	Structured RL
Stdev. training	0.8	5.4	1.8
Stdev. testing	11.9	13.5	1.9

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

Results: training time of algorithms in all environments

Training time	Structured IL	Unstructured RL	Structured RL
WSPP	7m	9m	9m
SMSP	10m	15m	12m
SVSP	11m	3m	23m
DVSP	12m	3m	31m
DAP	3m	5m	31m
GSPP	11m	10m	34m

SIL: Structured Imitation Learning, PPO: Proximal Policy Optimization (Unstructured RL), SRL: Structured Reinforcement Learning

WSPP: Warcraft Shortest Paths Problem, SMSP: Single Machine Scheduling Problem, SVSP: Stochastic Vehicle Scheduling Problem, DVSP: Dynamic Vehicle Scheduling Problem, DAP: Dynamic Assortment Problem, GSPP: Gridworld Shortest Paths Problem