POLYSTEPOR: LEARNING TO DECIDE WITHOUT OPTIMAL DECISIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision-focused learning (DFL) trains predictors for downstream decision quality, but often relies on optimal reference decisions that are expensive to obtain. We present PolyStepOR, which trains directly from realized decision costs without pre-computed optima and extends to in-constraint predictions through repair or infeasibility penalties. To handle piecewise-constant losses, PolyStepOR perturbs predictor parameters, evaluates the resulting decisions, and uses optimal transport to favor lower-cost directions, requiring no derivatives. Without task-specific tuning, PolyStepOR performs strongly on classical optimization benchmarks and competitively on predicted-constraint and real-world problems. Theoretically, we characterize decision-preserving perturbations and boundary detection, bound sensitivity to cost errors, and establish stationarity guarantees for a smoothed objective. PolyStepOR thus replaces optimal reference decisions and derivatives with forward evaluations.

1 INTRODUCTION

In many analytical applications, operational decisions are made under uncertainty. In vehicle routing, for instance, routes are determined before travel times are observed (Elmachtoub & Grigas, 2022). Decisions in these applications are commonly made through a two-stage approach called *predict-then-optimize* (Elmachtoub & Grigas, 2022; Mandi et al., 2024). In the first stage, a machine learning (ML) model predicts the unknown parameters from observed features. The second stage then solves a constrained optimization problem using the predicted parameters. ML models are often trained to minimize prediction errors using statistical losses such as mean-squared error. However, lower prediction error does not necessarily improve decision quality: large errors may leave the decision unchanged, while small errors near a decision boundary can substantially increase its cost. To bridge this gap, *decision-focused learning* (DFL) addresses this mismatch by incorporating downstream decision quality into the training of ML models. This is challenging for many combinatorial problems, as the selected decision is piecewise constant in the predicted parameters, resulting in zero gradients almost everywhere when the loss depends on predictions only through that decision. To obtain useful training signals, existing methods use continuous relaxations (Wilder et al., 2019), perturbations or approximate solver derivatives (Berthet et al., 2020; Pogancic et al., 2020), or surrogate losses (Blondel et al., 2020; Elmachtoub & Grigas, 2022). When predicted parameters enter the constraints, decisions may become infeasible under the true parameters, requiring methods that account for feasibility or repair costs (Silvestri et al., 2026; Mandi et al., 2026).

Another challenge is obtaining optimal reference decisions for training, which can be expensive for large instances (Parmentier, 2025; Ahmed et al., 2024). For a fixed feasible set, their costs do not affect the minimizers of expected unnormalized regret because they are independent of the predictor, although the reference decisions themselves can still provide useful training signals. Several DFL methods have proposed learning without optimal targets (Amos & Kolter, 2017; Gupta & Huang, 2024; Silvestri et al., 2026), motivating training from candidate evaluations when optimal decisions are costly to obtain. In this work, we propose PolyStepOR, a DFL framework that applies the forward-only PolyStep optimizer (Le, 2026) to train predictors directly from realized decision costs. It perturbs predictor parameters along structured directions, solves the resulting problems exactly or heuristically, and updates the predictor using cost-weighted perturbation directions. Evaluations can include repair costs or finite infeasibility penalties for in-constraint predictions. Training requires neither optimal reference decisions nor derivatives of the predictor, solver, or task loss. Finite

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

perturbations can expose decision changes where local gradients vanish, while a restricted search subspace controls the evaluation budget. In summary, our contributions are threefold:

1. We formulate DFL using forward cost evaluations, including for predicted constraints, and establish when removing optimal reference costs preserves both the training objective's minimizers and PolyStepOR's updates.
2. We characterize decision-preserving perturbations for linear optimization over finite feasible sets, and when a probe detects and crosses an isolated boundary toward a cheaper decision. We also bound the update's sensitivity to evaluation errors and show that the smoothed objective reaches stationarity at rate $O(T^{-1/2})$ after T iterations (Le, 2026), up to a residual set by the update bias.
3. We evaluate decision quality on classical optimization benchmarks (Mandi et al., 2024), in-constraint predictions (Silvestri et al., 2026), and extend to practical applications, including districting (Ahmed et al., 2024), and brass alloy production (Mandi et al., 2026).

2 Related work

Decision-focused learning. Existing DFL methods mainly differ in how they obtain useful training signals through optimization problems. Optimization layers differentiate convex problems or continuous relaxations (Amos & Kolter, 2017; Agrawal et al., 2019; Wilder et al., 2019), while perturbed optimizers and black-box differentiation smooth or approximate discrete solver maps (Berthet et al., 2020; Pogancic et al., 2020; Niepert et al., 2021). Surrogate losses, including SPO+, Fenchel-Young, contrastive, and ranking losses, instead use reference solutions to construct informative gradients (Elmachtoub & Grigas, 2022; Blondel et al., 2020; Mulamba et al., 2021; Mandi et al., 2022). Reference-free alternatives include perturbation-gradient losses (Gupta & Huang, 2024) and LANCER (Zharmagambetov et al., 2023), which learn a differentiable surrogate from decision-cost evaluations. These approaches differ in their supervision and solver requirements but retain gradient-based predictor training.

In-constraint predictions. When predicted parameters enter the constraints, decisions may become infeasible under the true parameters. Existing methods account for this through post-hoc correction (Hu et al., 2023b;a), score-function gradient estimation (SFGE) (Silvestri et al., 2026), or infeasibility penalties combined with supervision from optimal solutions (Mandi et al., 2026). SFGE avoids differentiating the solver or task loss but differentiates the prediction distribution; PolyStepOR instead perturbs predictor parameters directly. Unlike on-policy DFL, which learns from observed action costs (Benslimane et al., 2026), our setting requires evaluating every queried candidate using recorded data or simulation.

Derivative-free DFL. Derivative-free DFL has been explored through decision trees (Elmachtoub et al., 2020) and structured approximations trained without optimal targets (Parmentier, 2025). For neural predictors, evolution strategies offer a general approach to training from function evaluations (Salimans et al., 2017). PolyStep likewise uses function evaluations, but constructs updates from rotated polytope probes and entropy-regularized cost weights (Le et al., 2023; Le, 2026). We apply this optimizer to DFL and analyze how decision regions and evaluation errors affect its updates. The resulting PolyStepOR framework accommodates in-constraint predictions without optimal training targets or derivatives of the predictor, solver, or task loss.

3 Problem setup

Let $\boldsymbol{c} \in \mathbb{R}^p$ denote the unknown parameters of a constrained optimization (CO) problem:

$$\boldsymbol{x}^\star(\boldsymbol{c}) \in \arg\min_{\boldsymbol{x} \in \mathcal{X}(\boldsymbol{c})} f(\boldsymbol{x}, \boldsymbol{c}), \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is the objective and $\mathcal{X}(\boldsymbol{c}) \subseteq \mathbb{R}^n$ is the feasible set. The feasible set may contain discrete decisions and depend on $\boldsymbol{c}$. Evaluating our method requires neither convexity nor differentiability. For this formulation, we assume an optimum exists, with a fixed tie-breaking rule

selecting $\boldsymbol{x}^\star(\boldsymbol{c})$. Section 4.1 extends the setting to heuristic solvers. At decision time, $\boldsymbol{c}$ is unknown and predicted from observed correlated features $\boldsymbol{z}$:

$$\hat{\boldsymbol{c}} = \phi_{\boldsymbol{\theta}}(\boldsymbol{z}), \tag{2}$$

where $\phi_{\boldsymbol{\theta}}$ is an ML model parametrized by $\boldsymbol{\theta}$. The predicted parameters are passed to the solver to obtain the decision $\boldsymbol{x}^\star(\hat{\boldsymbol{c}})$. For a fixed feasible set, its quality is measured by the *regret*:

$$\texttt{Regret}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = f(\boldsymbol{x}^\star(\hat{\boldsymbol{c}}), \boldsymbol{c}) - f(\boldsymbol{x}^\star(\boldsymbol{c}), \boldsymbol{c}), \tag{3}$$

which quantifies the additional cost relative to the optimal *reference solution* under the true parameters. Let $(\boldsymbol{z}, \boldsymbol{c}) \sim \mathcal{P}$ denote a random pair of observed features and problem parameters with a joint distribution $\mathcal{P}$. Assuming finite expectations, minimizing expected regret is equivalent to minimizing expected realized decision cost:

$$\begin{aligned} \arg\min_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{z},\boldsymbol{c})\sim\mathcal{P}}[\texttt{Regret}(\phi_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{c})] \\ &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{z},\boldsymbol{c})\sim\mathcal{P}}[f(\boldsymbol{x}^\star(\phi_{\boldsymbol{\theta}}(\boldsymbol{z})), \boldsymbol{c})]. \end{aligned} \tag{4}$$

The reference cost (second term of Eq 3) is omitted because it is independent of $\boldsymbol{\theta}$. Proposition 1 establishes when this omission also preserves the parameter updates of our algorithm.

In-constraint prediction arises when predicted parameters enter the CO's constraints. For example, stochastic-weight knapsack (Silvestri et al., 2026) selects items with known values and capacity $\boldsymbol{b}$ but unknown weights $\boldsymbol{c}$, giving $\mathcal{X}(\boldsymbol{c}) = \{\boldsymbol{x} \in \{0,1\}^p : \boldsymbol{c}^\top \boldsymbol{x} \leq \boldsymbol{b}\}$. Here, decisions based on the predicted weights may violate the true capacity. A way to tackle this is to perform a *recourse action* to correct the decision to $\boldsymbol{x}^{\text{corr}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) \in \mathcal{X}(\boldsymbol{c})$, incurring a problem-specific penalty pen_ρ (Birge & Louveaux, 1997; Hu et al., 2023a;b). The resulting *post-hoc regret* then reads:

$$\texttt{PRegret}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = f(\boldsymbol{x}^{\text{corr}}(\hat{\boldsymbol{c}}, \boldsymbol{c}), \boldsymbol{c}) - f(\boldsymbol{x}^\star(\boldsymbol{c}), \boldsymbol{c}) + \text{pen}_\rho\big(\boldsymbol{x}^\star(\hat{\boldsymbol{c}}), \boldsymbol{x}^{\text{corr}}(\hat{\boldsymbol{c}}, \boldsymbol{c})\big). \tag{5}$$

The reference term is again independent of $\boldsymbol{\theta}$, so Eq. 4 also applies with the corrected cost and recourse penalty. For our algorithm, `Regret` and `PRegret` are used only for test-time evaluation.

4 METHOD

PolyStepOR applies PolyStep[1] (Le, 2026; Le et al., 2023) to the prediction–solver pipeline. It perturbs the predictor, evaluates the resulting decisions, and combines the perturbation directions using their realized costs. Finite probes can reveal decision changes even where local derivatives vanish.

4.1 LEARNING FROM REALIZED DECISION COSTS

Let $S(\hat{\boldsymbol{c}})$ return a decision for predicted parameters $\hat{\boldsymbol{c}}$. Our ideal oracle uses a deterministic exact solver or heuristic with fixed tie-breaking; time limits alone need not make a solver deterministic. Appendix A.6 discusses deviations from this assumption due to evaluation errors. On training data $\mathcal{D} = \{(\boldsymbol{z}_i, \boldsymbol{c}_i)\}_{i=1}^N$, the learning objective is:

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \ell\big(S(\phi_{\boldsymbol{\theta}}(\boldsymbol{z}_i)), \boldsymbol{c}_i\big), \tag{6}$$

where $\ell(\boldsymbol{x}, \boldsymbol{c})$ is the task loss of decision $\boldsymbol{x}$ under the true parameters $\boldsymbol{c}$. Evaluating L takes N solver calls per candidate predictor. We instead use a minibatch $\mathcal{B}$ of B training indices drawn uniformly at random,

$$\widehat{L}_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{B} \sum_{i \in \mathcal{B}} \ell\big(S(\phi_{\boldsymbol{\theta}}(\boldsymbol{z}_i)), \boldsymbol{c}_i\big). \tag{7}$$

For fixed $\boldsymbol{\theta}$, uniform sampling gives $\mathbb{E}_{\mathcal{B}}[\widehat{L}_{\mathcal{B}}(\boldsymbol{\theta})] = L(\boldsymbol{\theta})$ using B solver calls per candidate. This *cost oracle* requires no gradients or optimal reference decisions. It only requires true instance parameters or a simulation process (Ahmed et al., 2024) to score candidate decisions that may never have been taken in the historical data.

[1] A visualization of the algorithm: https://anonymous.4open.science/w/polystep-vis-anon-8D15/

Task loss. We use $\ell(\boldsymbol{x}, \boldsymbol{c}) = f(\boldsymbol{x}, \boldsymbol{c})$ for problems where predictions are only in the objective. If they are in the constraints, the decision $\boldsymbol{x}^\star = S(\hat{\boldsymbol{c}})$ may be infeasible (Silvestri et al., 2026). As in Eq. (5), we then use the cost of the corrected decision and the recourse penalty:

$$\ell(\boldsymbol{x}^\star, \boldsymbol{c}) = f\big(\boldsymbol{x}^{\mathrm{corr}}(\boldsymbol{x}^\star, \boldsymbol{c}), \boldsymbol{c}\big) + \mathrm{pen}_\rho\big(\boldsymbol{x}^\star, \boldsymbol{x}^{\mathrm{corr}}(\boldsymbol{x}^\star, \boldsymbol{c})\big), \tag{8}$$

where $\boldsymbol{x}^{\mathrm{corr}}(\boldsymbol{x}^\star, \boldsymbol{c}) \in \mathcal{X}(\boldsymbol{c})$ corrects $\boldsymbol{x}^\star$ under the true parameters. For an exact solver, this is the post-hoc regret of Eq. (5) without its reference term. If the solver returns no decision, a fixed finite failure cost or fallback decision is used.

4.2 Structured parameter perturbations

Each iteration evaluates nearby predictors, called *probes*, and favors directions with lower costs. There are 3 components in the algorithm: *search coordinates* control the evaluation budget, *symmetric probes* compare directions, and *cost weights* determine the update. Section 5 separates detecting a decision change from taking a step that improves the loss.

Search coordinates. We dentote $\boldsymbol{\theta}(\boldsymbol{y}) = \boldsymbol{\theta}_0 + A\boldsymbol{y}$, with fixed $A \in \mathbb{R}^{D_\theta \times d}$ and initial search coordinates $\boldsymbol{y} = \mathbf{0}$. Setting $A = I$ searches the full parameter space; fewer columns reduce the number of probes. Partition $d = Pq$ coordinates into P blocks of equal size q. Each probe changes one block, with the other blocks held fixed. Distances below are in these search coordinates.

Probes. Choose unit vertices $\boldsymbol{v}_1, \dots, \boldsymbol{v}_V \in \mathbb{R}^q$ with $\sum_v \boldsymbol{v}_v = 0$: either a regular simplex ($V = q + 1$) or the orthoplex $\{\pm \boldsymbol{e}_r\}_{r=1}^q$ ($V = 2q$). For each block, draw a Haar rotation $R_j \in \mathrm{SO}(q)$. When $q \geq 2$, every rotated vertex is uniform on the unit sphere. Probe each direction at radii $\rho_1, \dots, \rho_K > 0$, typically $\rho_k = rk/(K+1)$ for an outer radius $r > 0$, and average the costs:

$$\boldsymbol{y}_{jvk} = \boldsymbol{y} + \rho_k E_j R_j \boldsymbol{v}_v, \qquad C_{jv} = \frac{1}{K} \sum_{k=1}^{K} \widehat{L}_{\mathcal{B}}(\boldsymbol{\theta}(\boldsymbol{y}_{jvk})), \tag{9}$$

where E_j embeds a vector in block j. All probes use the same minibatch, so their costs compare predictors on the same instances. A shared batch still need not represent the full-dataset preference.

Update. Within each block, temperature $\varepsilon > 0$ controls how strongly the update favors cheaper directions:

$$w_{jv} = \frac{\exp(-C_{jv}/\varepsilon)}{\sum_{u=1}^{V} \exp(-C_{ju}/\varepsilon)}, \qquad \boldsymbol{D}_j(C) = \sum_{v=1}^{V} w_{jv} R_j \boldsymbol{v}_v, \tag{10}$$

and a step length $s > 0$ scales their simultaneous update:

$$\boldsymbol{y}^+ = \boldsymbol{y} + s \sum_{j=1}^{P} E_j \boldsymbol{D}_j(C), \qquad \boldsymbol{\theta}^+ = \boldsymbol{\theta}(\boldsymbol{y}^+). \tag{11}$$

Lower temperature concentrates weight on the cheapest directions, while higher temperature makes the weights more uniform and the resulting displacement smaller. The weights uniquely minimize $\sum_v C_{jv} w_v + \varepsilon \sum_v w_v \log w_v$ on the probability simplex, which gives the one-sided transport interpretation in Appendix A.1 (Le, 2026). Because the softmax is nonlinear, even unbiased probe-cost estimates do not generally produce unbiased weights; sampling noise can therefore amplify or weaken the preference for a direction. Since each $\boldsymbol{D}_j$ is a convex combination of unit vectors, Eq. (11) implies the step-size bound $|\boldsymbol{y}^+ - \boldsymbol{y}|_2 \leq s\sqrt{P}$. Since all blocks update simultaneously, the result generally differs from any single probe.

Momentum. The optional heavy-ball update accumulates barycentric displacements. With $D_t = \sum_j E_j D_{j,t}$, it is

$$v_{t+1} = \mu_t v_t + s_t D_t, \qquad y_{t+1} = y_t + \eta v_{t+1}, \qquad v_0 = 0, \tag{12}$$

where $\eta > 0$ scales the velocity and $0 \leq \mu_t < 1$ controls momentum. We linearly interpolate μ_t between its initial and final values during training. Setting $\mu_t = 0$ and $\eta = 1$ recovers Eq. (11). Momentum can carry the predictor across plateaus, but may also oppose a new signal. Hence, $s_t\sqrt{P}$ bounds only the fresh displacement, not the full momentum step. The implementation resets the velocity when the search basis changes, while the convergence analysis assumes a fixed basis.

Algorithm 1 One iteration of PolyStepOR

Require: State (y_t, v_t), with $v_0 = 0$; reconstruction $\theta(\cdot)$; shared-batch oracle $\widehat{L}_{\mathcal{B}}$; block embeddings $E_{1:P}$; centered unit vertices $\boldsymbol{v}_{1:V}$; radii $\rho_{1:K} > 0$, temperature $\varepsilon > 0$, and step $s_t > 0$.
Require: Momentum $0 \le \mu_t < 1$ and multiplier $\eta > 0$; use $(\mu_t, \eta) = (0, 1)$ without momentum.
1: **for** $j = 1, \dots, P$ **do**
2: Draw an independent Haar rotation $R_j \in \mathrm{SO}(q)$.
3: $C_{jv} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \widehat{L}_{\mathcal{B}}\big(\theta(y_t + \rho_k E_j R_j \boldsymbol{v}_v)\big)$ for $v = 1, \dots, V$.
4: $w_{j:} \leftarrow \mathrm{softmax}(-C_{j:}/\varepsilon)$; $D_{j,t} \leftarrow \sum_{v=1}^{V} w_{jv} R_j \boldsymbol{v}_v$.
5: **end for**
6: $D_t \leftarrow \sum_{j=1}^{P} E_j D_{j,t}$.
7: $v_{t+1} \leftarrow \mu_t v_t + s_t D_t$; $y_{t+1} \leftarrow y_t + \eta v_{t+1}$.
8: **return** state (y_{t+1}, v_{t+1}) and predictor parameters $\theta(y_{t+1})$.

Evaluation budget. Each iteration evaluates $Q = PVK$ candidate predictors and makes BQ solver calls. Simplex and orthoplex probes use $K(q+1)d/q$ and $2Kd$ evaluations, respectively. Recourse, validation, and restarts add to this count. Appendix A.9 gives arithmetic and storage bounds; runtime comparisons should also include tuning and baseline reference generation.

4.3 Invariance to reference costs

We show that reference subtraction preserves both the objective minimizers (Section 3) and the PolyStepOR updates, because the weights depend only on within-row cost differences.

Proposition 1 (Reference-cost invariance)**.** *With a shared minibatch across probes and identical temperature and cost scaling, replacing each instance loss by $\ell(\boldsymbol{x}, \boldsymbol{c}_i) - r_i$ for parameter-independent r_i leaves the weights unchanged. Runs with the same initial parameters and velocity, random draws, prescribed schedules, and stopping horizon have identical parameter trajectories, including under Eq.* (12)*.*

This means that computing $r_i = f(\boldsymbol{x}^\star(\boldsymbol{c}_i), \boldsymbol{c}_i)$ gives the same updates from costs and unnormalized regret without computing the references. Their rankings also agree on a fixed validation set. Scaling instance costs by fixed positive factors preserves reference subtraction but changes their relative weighting. Loss-dependent normalization and feasibility-conditional evaluation are handled separately in Appendix A.1.

5 When do probes yield a decision signal?

Finite probes can reveal cheaper decisions where local derivatives vanish. We ask when they detect a change, how instance costs combine, and whether the resulting motion reduces loss. For the geometric analysis, we assume exact linear optimization over fixed finite feasible sets and decision-only losses, and denote the batch loss in search coordinates by setting $\widehat{\mathcal{L}}_{\mathcal{B}}(y) = \widehat{L}_{\mathcal{B}}(\theta(y))$. Proofs and extensions are in Appendix A.

The first result measures how far a probe must reach to change a decision.

Proposition 2 (Decision margin)**.** *Let $\boldsymbol{x}_0$ uniquely minimize $\hat{\boldsymbol{c}}(\boldsymbol{y})^\top \boldsymbol{x}$ over a finite set $\mathcal{F}$, and suppose $\hat{\boldsymbol{c}}(\boldsymbol{y}) = \boldsymbol{b} + M\boldsymbol{y}$. The distance to loss of unique optimality in search coordinates is*

$$r_{\boldsymbol{y}} = \min_{\substack{\boldsymbol{x} \in \mathcal{F} \setminus \{\boldsymbol{x}_0\} \\ M^\top(\boldsymbol{x} - \boldsymbol{x}_0) \neq \boldsymbol{0}}} \frac{\hat{\boldsymbol{c}}(\boldsymbol{y})^\top (\boldsymbol{x} - \boldsymbol{x}_0)}{\|M^\top (\boldsymbol{x} - \boldsymbol{x}_0)\|_2}, \qquad \min \varnothing = +\infty. \tag{13}$$

Every perturbation $\|\boldsymbol{h}\|_2 < r_{\boldsymbol{y}}$ preserves the decision. With exact evaluations and a loss depending on parameters only through the returned decision, unchanged decisions at every batch probe imply $\boldsymbol{D}_j = \boldsymbol{0}$ for every block.

The ratio compares the predicted-cost gap between the current decision and an alternative decision with the maximum rate at which a perturbation can close that gap. If the denominator is zero, no

perturbation in the search space can close it. The smallest such margin across a finite batch gives the radius below which all current decisions remain unchanged. Under exact decision-only evaluations, probes within this radius therefore yield no new update direction, although momentum from earlier iterations may still move the predictor. These margins are used only for analysis; Appendix A.2 extends the result to individual parameter blocks and nonlinear predictors.

When a nearby boundary separates the current decision from a cheaper one, the probes may detect the cheaper region even if the subsequent update is too small to cross the boundary.

Proposition 3 (Detection and crossing)**.** *Fix a batch and a block j. Suppose the loss in a neighborhood is $\widehat{\mathcal{L}}_{\mathcal{B}}(\boldsymbol{y} + E_j\boldsymbol{h}) = L_- + J\mathbf{1}\{\delta + \boldsymbol{n}^\top\boldsymbol{h} > 0\}$, where $\|\boldsymbol{n}\|_2 = 1$, $J > 0$, and either value is allowed at a tie. Use orthoplex probes with exact costs, all inside the neighborhood and none on the boundary. Then*

$$\boldsymbol{n}^\top\boldsymbol{D}_j < 0 \quad\Longleftrightarrow\quad \rho_{\max}\|R_j^\top\boldsymbol{n}\|_\infty > |\delta|, \qquad \rho_{\max} = \max_k \rho_k. \tag{14}$$

Otherwise $\boldsymbol{D}_j = \mathbf{0}$. If $m = -\boldsymbol{n}^\top\boldsymbol{D}_j > 0$, a block-only update remaining in the neighborhood reaches the lower-cost side when $sm > \delta$.

Here, δ is the signed distance to the boundary, positive on the higher-cost side, while $|R_j^\top\boldsymbol{n}|_\infty$ measures probe alignment with the boundary normal. The probe radius determines whether the cheaper side is detected, and the step length whether the update crosses into it. This extends PolyStep's centered planar-jump analysis (Le, 2026). In a batch, however, different instances may favor different directions or cross different boundaries, so these signals can interact before the update is formed.

What averaging changes. The boundary result above considers a single local boundary, whereas PolyStepOR computes probe costs by averaging over the batch before forming the weights. For orthoplex pairs $\pm u_r$, let $C_{r,\pm}$ be their costs averaged over instances and radii, $\Delta_r = C_{r,+} - C_{r,-}$, and $\pi_r = w_{r,+} + w_{r,-}$. Then

$$D_j = -\sum_{r=1}^{q} \pi_r \tanh\left(\frac{\Delta_r}{2\varepsilon}\right) u_r. \tag{15}$$

Each pair points toward its cheaper side, with strength set by its weight π_r and cost difference relative to temperature. Cost differences from different instances can cancel before weighting. A lone changing instance contributes only its jump divided by batch size, weakening the signal at fixed temperature. Proposition 4 covers batches and populations, recovering Proposition 3 when changing instances share a boundary and favor the same side.

The role of momentum. Stored velocity can carry a moving predictor through a plateau; equal probe costs alone cannot start motion from rest. For a block-only step in the same two-value neighborhood, the crossing condition becomes $\delta + \eta n^\top v_{j,t+1} < 0$, where $v_{j,t+1} = E_j^\top v_{t+1}$. Earlier directions can help cross or oppose the current preference (Appendix A.5).

What repeated steps can guarantee. Following PolyStep's smoothing analysis (Le, 2026), we study a smoothed objective F obtained by averaging the training loss over nearby search coordinates. A small ∇F means that nearby perturbations change the average loss only slightly. Under fixed coordinates, temperature, and probe distribution, suitable smoothness, and conditionally unbiased batches, Theorems 6 and 8 bound the expected squared gradient by $O(T^{-1/2})$ plus a bias residual from nonlinear weights, block interactions, and cost errors. Proposition 5 controls the effect of cost errors relative to temperature (Cohen & Fausti, 2023). Thus, we obtain stationarity guarantees for the smoothed objective, including with momentum, but not acceleration or optimal decisions.

6 Experiments

We evaluate PolyStepOR on classical optimization benchmarks (Mandi et al., 2024), in-constraint prediction problems (Silvestri et al., 2026), and two real-world applications: network districting (Ahmed et al., 2024) and brass alloy production (Mandi et al., 2026). We build directly on original benchmark implementations, maintaining their predictor architectures and optimization formulations. For the first two benchmarks, we employ fixed hyperparameter configurations for PolyStepOR

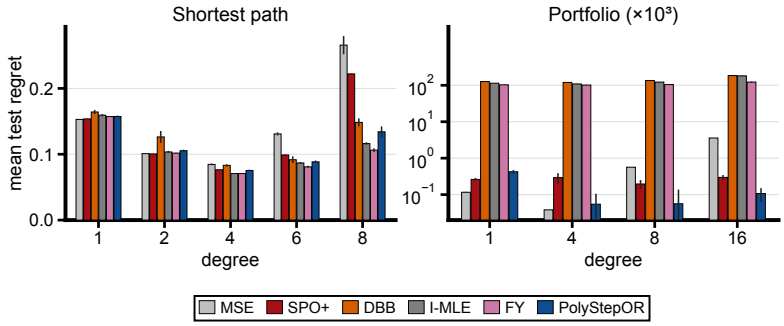

Figure 1: Mean test regret on synthetic shortest-path and portfolio tasks as misspecification degree varies; lower is better. The portfolio panel uses a logarithmic axis and the displayed $\times 10^3$ scale.

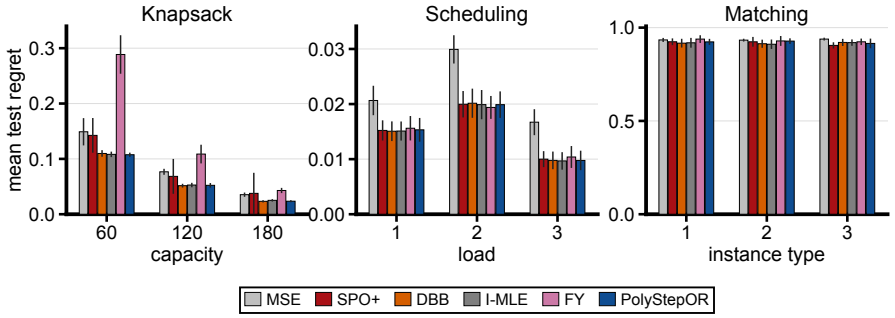

Figure 2: Mean test regret for knapsack capacities and scheduling loads on energy-price data (Ifrim et al., 2012), and matching instance types on citation networks (Sen et al., 2008); lower is better.

without task-specific tuning and observe consistently strong performance out of the box. The ML models are trained directly on realized decision costs, and optionally including recourse costs for in-constraint predictions. The optimal reference solutions are used only to *compare test regret against baselines*. Appendix B.5 provides full hyperparameter configurations and extended results.

6.1 Classical optimization benchmarks

We evaluate PolyStepOR on five classical DFL benchmarks from Mandi et al. (2024): two synthetic tasks (Figure 1) and three real-world tasks (Figure 2), comparing against MSE, SPO+ (Elmachtoub & Grigas, 2022), DBB (Pogancic et al., 2020), I-MLE (Niepert et al., 2021), and FY (Blondel et al., 2020) over 10 seeds. Baseline results are taken from Mandi et al. (2024). PolyStepOR is competitive across tasks and levels of model misspecification. On portfolio, it achieves the lowest mean regret across polynomial degrees, with $(0.160 \pm 0.031) \times 10^{-3}$ versus $(0.262 \pm 0.032) \times 10^{-3}$ for SPO+ ($p = 0.002$). On knapsack and scheduling, it is statistically indistinguishable from the baseline with the lowest mean regret in all six settings ($p \geq 0.19$). Performance is weaker on shortest path, especially at degree 8, where nine of ten runs exhaust the step budget without early stopping. Overall, these results show that realized-cost training can match strong DFL baselines without optimal reference targets, although performance remains problem-dependent.

6.2 Predicted constraint parameters

We evaluate three in-constraint problems from Silvestri et al. (2026), such as predicting knapsack capacity, item weights, and weighted set multi-cover requirements. We compare against MSE, CombOptNet (Paulus et al., 2021) where available, and SFGE (Silvestri et al., 2026) over 15 runs per setting. PolyStepOR uses one fixed configuration across all settings without task-specific tuning; full training details are provided in Appendix B. Here, PolyStepOR achieves the lowest mean rela-

Table 1: Mean relative post-hoc regret over 15 runs of 3 in-constraint prediction problems. Bold underlined marks the lowest mean in each row and bold the second lowest.

Problem	ρ	MSE	CombOptNet	SFGE	PolyStepOR
KP (capacity)	5	0.5556 ± 0.3529	–	$\mathbf{0.3577 \pm 0.2152}$	$\underline{\mathbf{0.2975 \pm 0.1666}}$
	10	1.2126 ± 0.7797	–	$\mathbf{0.5081 \pm 0.2784}$	$\underline{\mathbf{0.3802 \pm 0.1958}}$
	20	2.5205 ± 1.6313	–	$\mathbf{0.8216 \pm 0.5031}$	$\underline{\mathbf{0.4772 \pm 0.2415}}$
KP (weight)	5	0.1676 ± 0.0357	0.1901 ± 0.0289	$\underline{\mathbf{0.1297 \pm 0.0114}}$	$\mathbf{0.1501 \pm 0.0210}$
	10	0.3190 ± 0.0814	0.4015 ± 0.0659	$\underline{\mathbf{0.1760 \pm 0.0221}}$	$\mathbf{0.2293 \pm 0.0372}$
	20	0.6146 ± 0.1745	0.8233 ± 0.1401	$\underline{\mathbf{0.2205 \pm 0.0276}}$	$\mathbf{0.2884 \pm 0.0558}$
WSMC 10×50	1	1.2960 ± 0.5676	5.2044 ± 2.1887	$\mathbf{1.1373 \pm 0.4368}$	$\underline{\mathbf{1.1205 \pm 0.5002}}$
	5	5.7115 ± 2.7809	110.7887 ± 30.2424	$\mathbf{2.5460 \pm 0.9283}$	$\underline{\mathbf{2.5253 \pm 1.0119}}$
	10	11.2307 ± 5.5511	440.7395 ± 118.7445	$\underline{\mathbf{3.3108 \pm 1.1913}}$	$\mathbf{3.3701 \pm 1.5660}$

tive post-hoc regret at all three penalty levels for predicted capacity ($p \leq 0.002$ versus SFGE). For predicted item weights, SFGE performs better at every penalty level, while PolyStepOR improves over MSE and CombOptNet. On multi-cover (WSMC), PolyStepOR and SFGE do not differ significantly ($p \geq 0.25$). Appendix B reports prediction error and infeasibility for further analysis.

6.3 Case Study: Network Districting

DistrictNet (Ahmed et al., 2024) aims to partition a geographical area into connected districts of bounded size to minimize expected routing cost (Ahmed et al., 2024). For city i, the cost of a feasible districting $\lambda \in \Lambda_i$ is estimated over demand scenarios $\boldsymbol{\xi}^\omega$ as:

$$C_i(\lambda) = \sum_{d \in \lambda} \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \mathrm{TSP}(d, \boldsymbol{\xi}^\omega), \tag{16}$$

where $\mathrm{TSP}(d, \boldsymbol{\xi}^\omega)$ is the tour length serving district d under scenario ω. Obtaining exact solutions is expensive in this application, which can take roughly 400 CPU-core days for one instance with 60 geographical units and 10 districts (Ahmed et al., 2024). DistrictNet addresses this by solving a capacitated minimum spanning tree (CMST) surrogate. A neural network first predicts edge weights, and solving the surrogate using the weights then produces a district partition. The model is trained on optimal districtings of small cities, encoded as targets with the FY loss (Blondel et al., 2020). We adapt PolyStpPOR on the same network and surrogate but train on the estimated routing costs:

$$\min_{\boldsymbol{\theta}} \ \frac{1}{n} \sum_{i=1}^{n} C_i\big(S(\phi_{\boldsymbol{\theta}}(\boldsymbol{z}_i))\big), \tag{17}$$

where S maps predicted edge weights to a districting. We use the authors' precomputed district costs to evaluate candidate partitions, without using optimal districtings.

Table 2 reports results on 35 benchmark problems and the 2,000-unit Ile-de-France instance. Across ten seeds, PolyStepOR is competitive with DistrictNet on the test data. The relative cost is the mean over the 35 problems of each method's cost difference to PolyStepOR divided by the PolyStepOR cost. On the large instance Ile-de-France, the best of ten PolyStepOR runs costs 2156.3, compared with the published DistrictNet cost of 2205.7. These results demonstrate competitive districting quality from routing-cost evaluations without optimal training partitions, and can potentially scale up to larger instances for training. We further visualize the real districting solutions in Appendix B.3.

6.4 Case Study: Brass alloy production

We apply PolyStepOR to brass alloy production, where a factory purchases ore from 10 suppliers to meet copper and zinc requirements using predicted metal contents (Mandi et al., 2026). PolyStepOR trains on purchase costs including recourse penalties for shortages, without optimal reference purchases or an additional feasibility-specific loss. We also evaluate SFGE (Silvestri et al., 2026) using post-hoc regret. ODECE (Mandi et al., 2026) instead combines infeasibility and optimality-preserving losses, using reference solutions and a weight α to control their balance (Appendix B.4).

Table 2: Districting costs on the 35 test problems (left), which consists of 7 real cities at 120 units, 5 target sizes and on the large instance Ile-de-France (right) with 2000 units.

	35 test problems			Ile-de-France	
Method	Cost	Rel. (%)	p	Cost	Rel. (%)
BD	594.1	9.46	1×10^{-8}	2379.0	10.33
FIG	594.3	9.60	1×10^{-8}	2388.8	10.78
PredGNN	603.0	10.95	1×10^{-10}	2295.2	6.44
AvgTSP	583.3	3.86	2×10^{-5}	2262.7	4.93
DistrictNet	**557.7**	-0.05	0.14	**2205.7**	2.29
PolyStepOR	**554.0**	0.00	–	**2156.3**	0.00

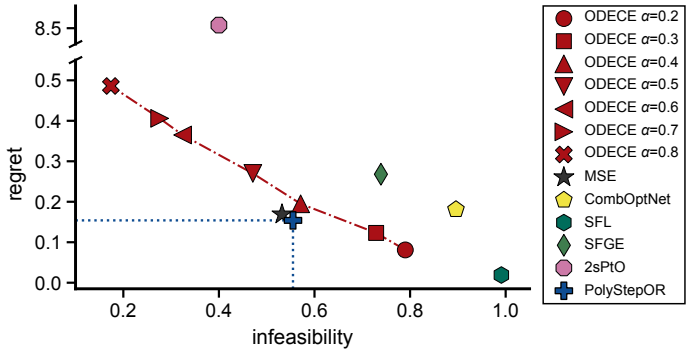

Figure 3: Infeasibility ratio and regret of feasible solutions on test instances (adapted from Mandi et al., 2026). PolyStepOR is competitive with ODECE's strongest baseline (MSE).

Figure 3 reports infeasibility before correction and regret among feasible purchases. PolyStepOR achieves 55.5% infeasibility and regret of 0.154, close to MSE at 53.2% and 0.169, and below SFGE's reported values on both metrics. At similar infeasibility, ODECE with $\alpha = 0.4$ has regret of 0.195; larger α reduces infeasibility at the expense of regret. PolyStepOR therefore achieves competitive results from corrected-cost evaluations alone, close to their strongest baseline (MSE).

7 Discussion and Conclusion

PolyStepOR trains predictors from realized decision costs without optimal training targets or derivatives of the predictor, solver, or task loss. It accommodates exact and heuristic solvers, including in-constraint predictions evaluated through recourse. Experiments demonstrate competitive decision quality across several benchmarks and practical applications, although performance varies across problems and cost-based training does not ensure first-stage feasibility. Our analysis separates boundary detection from crossing, explains how batch averaging and momentum affect the signal, and bounds the expected squared gradient of a smoothed objective with an explicit bias residual. The bounds cover fixed search coordinates, including scheduled momentum, under sampling and regularity assumptions; they do not guarantee improvement at every step or cover all practical configurations. The main computational cost remains repeated solver calls, making the approach particularly relevant when candidate decisions can be evaluated much more cheaply than optimal reference decisions can be obtained.

Future work includes more efficient parallel evaluations, adaptive probe radii, and analysis of practical optimizer variants. Incorporating an explicit feasibility objective could also provide greater control over the trade-off between decision cost and feasibility.

AI USE STATEMENT

LLMs assisted with writing, editorial checks, and literature review. The LLM also helped to implement and debug the experiment code. The authors directed the research, verified all results, and take full responsibility for all methods, proofs, experiments, and claims.

ETHICS STATEMENT

This work uses only publicly available benchmark data (synthetic instances, electricity prices, the CORA citation network, public geographic and population data for city districting, and alloy supplier data) and involves no human subjects or personal information.

REPRODUCIBILITY STATEMENT

To aid complete reproducibility, Appendix B provides sets of hyperparameters per experiment conducted in this paper that the audience can set up and reproduce. Our algorithm is available in Julia and Python programming languages. The full code implementation is provided via this link: https://anonymous.4open.science/r/polystep-or-52D7.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Cheikh Ahmed, Alexandre Forel, Axel Parmentier, and Thibaut Vidal. DistrictNet: Decision-aware learning for geographical districting. *Advances in Neural Information Processing Systems*, 37: 128574–128602, 2024.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Wyame Benslimane, Tinghan Ye, Pascal Van Hentenryck, and Paul Grigas. Decision-focused on-policy learning for contextual linear optimization with partial feedback. *arXiv preprint arXiv:2606.01081*, 2026.

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer, 1997.

Mathieu Blondel, André FT Martins, and Vlad Niculae. Learning with fenchel-young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

Samuel N Cohen and Eliana Fausti. Hyperbolic contractivity and the hilbert metric on probability measures. *arXiv preprint arXiv:2309.02413*, 2023.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management science*, 68(1): 9–26, 2022.

Adam N Elmachtoub, Jason Cheuk Nam Liang, and Ryan McNellis. Decision trees for decision-making under the predict-then-optimize framework. In *International conference on machine learning*, pp. 2858–2867. PMLR, 2020.

Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 385–394, 2005.

Peyré Gabriel and Cuturi Marco. Computational optimal transport with applications to data sciences. *Foundations and trends® in machine learning*, 11(5-6):355–607, 2019.

Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

Vishal Gupta and Michael Huang. Decision-focused learning with directional gradients. *Advances in Neural Information Processing Systems*, 37:79194–79220, 2024.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

Xinyi Hu, Jasper Lee, and Jimmy Lee. Two-stage predict+ optimize for milps with unknown parameters in constraints. *Advances in neural information processing systems*, 36:14247–14272, 2023a.

Xinyi Hu, Jasper CH Lee, and Jimmy HM Lee. Predict+ optimize for packing and covering lps with unknown parameters in constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 3987–3995, 2023b.

Georgiana Ifrim, Barry O'Sullivan, and Helmut Simonis. Properties of energy-price forecasts for scheduling. In *International Conference on Principles and Practice of Constraint Programming*, pp. 957–972. Springer, 2012.

An T. Le. Training non-differentiable networks via optimal transport. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=8mlcqTTMuU. To appear.

An T Le, Georgia Chalvatzaki, Armin Biess, and Jan R Peters. Accelerating motion planning via optimal transport. *Advances in Neural Information Processing Systems*, 36:78453–78482, 2023.

Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning*, pp. 14935–14947. PMLR, 2022.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701, 2024.

Jayanta Mandi, Marianne Defresne, Senne Berden, and Tias Guns. Feasibility-aware decision-focused learning for predicting parameters in the constraints. *Advances in Neural Information Processing Systems*, 38:163285–163309, 2026.

Maxime Mulamba, Jayanta Mandi, Michelangelo Diligenti, Michele Lombardi, Victor Bucarey, and Tias Guns. Contrastive losses and solution caching for predict-and-optimize. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2833–2840. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/390. URL https://doi.org/10.24963/ijcai.2021/390. Main Track.

Yatin Nandwani, Rishabh Ranjan, Parag Singla, et al. A solver-free framework for scalable learning in neural ilp architectures. *Advances in Neural Information Processing Systems*, 35:7972–7986, 2022.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit mle: backpropagating through discrete exponential family distributions. *Advances in Neural Information Processing Systems*, 34:14567–14579, 2021.

Axel Parmentier. Learning structured approximations of combinatorial optimization problems. *Open Journal of Mathematical Optimization*, 6:1–27, 2025.

Anselm Paulus, Michal Rolínek, Vít Musil, Brandon Amos, and Georg Martius. Comboptnet: Fit the right np-hard problem by learning integer programming constraints. In *International Conference on Machine Learning*, pp. 8443–8453. PMLR, 2021.

Marin Vlastelica Pogancic, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *ICLR*, 2020.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–106, 2008.

Mattia Silvestri, Senne Berden, Gaetano Signorelli, Ali İrfan Mahmutoğulları, Jayanta Mandi, Brandon Amos, Tias Guns, and Michele Lombardi. Score function gradient estimation to widen the applicability of decision-focused learning. *Journal of Artificial Intelligence Research*, 85, 2026.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1658–1665, 2019.

Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. *arXiv preprint arXiv:1808.10396*, 2018.

Arman Zharmagambetov, Brandon Amos, Aaron Ferber, Taoan Huang, Bistra Dilkina, and Yuandong Tian. Landscape surrogate: Learning decision losses for mathematical optimization under partial information. *Advances in Neural Information Processing Systems*, 36:27332–27350, 2023.

A FULL ANALYSIS AND PROOFS

We first establish what a single update can learn from its probes, then study repeated updates. The local results explain when a decision change produces a direction and how averaging or momentum changes its effect. The multi-step analysis concerns a smoothed mean loss and states its sampling and regularity assumptions separately. Norms on search coordinates are Euclidean unless stated otherwise.

A.1 TRANSPORT ROWS AND REFERENCE COSTS

Optimal transport assigns mass from sources to targets according to a cost matrix $C \in \mathbb{R}^{P\times V}$. A plan $T_{jv} \geq 0$ records the mass assigned from source j to target v, and $\langle C, T\rangle = \sum_{jv} C_{jv}T_{jv}$ is its total cost. Entropic regularization favors distributed assignments when costs are similar (Cuturi, 2013; Gabriel & Marco, 2019). Given positive source masses a_j with $\sum_j a_j = 1$, the one-sided problem is

$$\min_{T\geq 0,\, T\mathbf{1}=a} \langle C, T\rangle + \varepsilon H(T), \qquad H(T) = \sum_{jv} T_{jv}(\log T_{jv} - 1), \quad \varepsilon > 0, \tag{18}$$

with $0\log 0 = 0$. Only the source masses are fixed, so each row can be optimized independently. Its unique solution is $T_{jv} = a_j w_{jv}$ with w_{jv} from Eq. (10). The source masses cancel upon row normalization. In PolyStepOR, $a_j = 1/P$ and each target label denotes a block-local direction; it is not a shared point in parameter space.

For a fixed row mass $a_j > 0$, write $T_{jv} = a_j w_v$ with w in the probability simplex Δ_V. Up to an additive constant and the positive factor a_j, the row objective is $J(w) = \sum_v C_{jv}w_v + \varepsilon \sum_v w_v \log w_v$. Put $Z = \sum_v \exp(-C_{jv}/\varepsilon)$ and $p_v = \exp(-C_{jv}/\varepsilon)/Z$. Direct substitution gives

$$J(w) - J(p) = \varepsilon \sum_v w_v \log \frac{w_v}{p_v} \geq 0.$$

The inequality follows from $\log t \leq t - 1$: summing $w_v \log(p_v/w_v) \leq p_v - w_v$ over positive w_v gives a nonpositive sum. Since all $p_v > 0$, omitting any index with $w_v = 0$ makes this bound strict. With full support, equality in $\log t \leq t - 1$ requires $w_v = p_v$ for every v. Thus p is the unique minimizer.

Proof of Proposition 1. Write $r_{\mathcal{B}} = B^{-1}\sum_{i\in\mathcal{B}} r_i$. Replacing C_{jv} by $C_{jv} - r_{\mathcal{B}}$ multiplies every exponential in a row by $e^{r_{\mathcal{B}}/\varepsilon}$, which cancels in Eq. (10). If two runs have the same parameters and velocity at step t, their identical queries therefore give the same D_t. Equation (12) then gives the same velocity and parameters at step $t + 1$. Induction from their common initialization proves trajectory invariance. This uses identical prescribed schedules and stopping horizons; an unadjusted rule based on absolute loss values need not be invariant. □

Reference shifts and scaling. The same argument allows a separate constant shift for each block. At fixed probes and step length, multiplying every cost and the temperature by the same positive factor also preserves the weights. Temperature therefore has the same units as the cost being weighted.

Why normalization changes the objective. Consider two instances with optimal costs $(1, 100)$. Suppose two predictors induce cost vectors $(1, 110)$ and $(2, 100)$, respectively. The first predictor has mean regret 5, versus $1/2$ for the second, but mean relative regret $1/20$, versus $1/2$. Thus per-instance division by the reference cost reverses their ordering. For fixed positive divisors d_i, subtracting r_i/d_i still cancels from each aggregated probe row: normalized regret and normalized cost give the same update. It is the division by d_i, not reference subtraction, that changes the objective relative to raw cost. The example uses two feasible decisions per instance, with costs $(1, 2)$ and $(100, 110)$.

Reference subtraction need not preserve instance-wise batch standardization. Costs $(2, 4)$ and references $(1, 4)$ give regrets $(1, 0)$. The within-batch variances are 1 and $1/4$, respectively, and the centered vectors even have opposite signs. Proposition 1 applies when the same reference is subtracted from each *aggregated probe cost*. It also applies to row centering or row-spread scaling that

is itself invariant to common shifts. It does not justify replacing a loss-dependent normalization computed across instances by a different one.

Regret conditional on feasibility poses a different issue: its averaging set depends on the predictor. Even a parameter-independent reference cost can then have a parameter-dependent conditional mean. The common-shift argument applies to a fixed population of instances, including when every instance is scored after repair.

A.2 Decision margins

For one instance, let $\mathcal{F}$ be a fixed finite feasible set with at least two decisions and let x_0 uniquely minimize $\hat{c}^\top x$. In a prediction norm $\|\cdot\|$ with dual norm $\|u\|_* = \sup_{\|v\|\le 1} u^\top v$, define

$$r_c = \min_{x \in \mathcal{F}\setminus\{x_0\}} \frac{\hat{c}^\top (x - x_0)}{\|x - x_0\|_*} > 0. \tag{19}$$

The region where x_0 is optimal is the polyhedral cone $\mathcal{C}_{x_0} = \{c : c^\top (x - x_0) \ge 0 \text{ for every } x \in \mathcal{F}\}$. The exact distance to loss of unique optimality in this norm is r_c. If the predictor is locally H-Lipschitz, $H > 0$, a search perturbation $\|h\|_2 < r_c/H$ within that neighborhood preserves the decision. These statements extend Proposition 2 to general prediction norms and nonlinear predictors. The affine formula in that proposition gives the exact distance in search coordinates. For block j, replace $M^\top$ with $E_j^\top M^\top$ in Eq. (13) to obtain $r_{y,j}$.

Proof of Proposition 2 and its extensions. A point x_0 minimizes $c^\top x$ if and only if $c^\top (x - x_0) \ge 0$ for all $x \in \mathcal{F}$, proving the polyhedral description. At the given unique optimum all gaps $m_x = \hat{c}^\top (x - x_0)$ for $x \ne x_0$ are positive. Hölder's inequality gives

$$(\hat{c} + \Delta c)^\top (x - x_0) \ge m_x - \|\Delta c\| \, \|x - x_0\|_* > 0 \quad \text{when } \|\Delta c\| < r_c.$$

Local Lipschitz continuity bounds $\|\Delta c\|$ by $H\|h\|_2$ whenever the perturbation remains in the stated neighborhood. If $H = 0$, the prediction is constant on that neighborhood and the same conclusion holds there without dividing by H.

The prediction-space radius is also exact. Choose a decision x attaining the minimum in Eq. (19). The unit ball is compact, so there is a unit vector v with $v^\top (x - x_0) = \|x - x_0\|_*$. At $\Delta c = -r_c v$, this gap is zero and all other gaps are nonnegative by the same inequality. Hence x_0 loses unique optimality at distance r_c.

For the affine map, let $a_x = M^\top (x - x_0)$. The perturbed gap is $m_x + a_x^\top h$. If $a_x = 0$, it stays strictly positive for every h. Otherwise, the nearest point on its zero hyperplane is

$$h_x = -\frac{m_x}{\|a_x\|_2^2} a_x, \qquad \|h_x\|_2 = \frac{m_x}{\|a_x\|_2}.$$

Every h of norm less than the minimum of these distances has all gaps positive. At a minimizing h_x, that gap is zero and every other gap is nonnegative by the same norm bound. Thus unique optimality is lost at exactly r_y. If no a_x is nonzero, all gaps stay positive everywhere and the distance is infinite. At the distance itself, the selected decision depends on tie-breaking. If the prediction map is affine only locally, the distance formula is valid only while the relevant ball and witness remain in that affine region.

Replacing the affine map M with ME_j gives the block-specific radius by the same argument. Finally, unchanged decisions and a decision-only task loss make every probe in a row have the same cost. Its softmax weights equal $1/V$, so $D_j = V^{-1} R_j \sum_v v_v = 0$. The argument assumes exact evaluations, or common evaluation noise that preserves these equalities; independent noisy evaluations can induce motion even on a plateau. □

The Lipschitz bound gives a sufficient radius, while the affine formula gives the exact distance to a tie in the search coordinates. For a finite batch, let $r_{i,j}$ be instance i's block-j radius. If $\rho_{\max} < \min_{i \in \mathcal{B}} r_{i,j}$, every instance keeps its decision at every block-j probe and $D_j = 0$. The batch mean may stay constant beyond this radius because distinct decisions can have equal costs or different instances' cost changes can cancel. Zero D_j describes the fresh signal; momentum may still move the parameters.

A changed cost need not give a fresh direction. For a regular unit simplex with $V = q + 1$, PolyStep's identity (Le, 2026) gives

$$\left\| \sum_v w_v R v_v \right\|_2^2 = \frac{V}{V-1} \sum_v (w_v - 1/V)^2.$$

To check the identity, use $v_v^\top v_u = -1/(V-1)$ for $v \neq u$ and $\sum_v w_v = 1$:

$$\left\| \sum_v w_v R v_v \right\|_2^2 = \sum_v w_v^2 - \frac{1 - \sum_v w_v^2}{V-1} = \frac{V}{V-1} \left(\sum_v w_v^2 - \frac{1}{V} \right).$$

Thus the simplex direction is zero exactly when the finite-temperature cost row is constant. For an orthoplex, the row $(0, 0, 1, 1)$ on $(e_1, -e_1, e_2, -e_2)$ has unequal costs but zero direction, because each antipodal pair has equal weights. A decision change, a cost change, a fresh direction, parameter motion, and a loss decrease are distinct events. These directions live in search coordinates; a nonzero displacement can also vanish under reconstruction if it lies in the null space of A.

A.3 Updates at decision boundaries

To determine whether an update approaches a boundary, only its component along the normal matters. Orient each antipodal pair as $\pm u_r$ with $a_r = n^\top u_r \geq 0$. Let $w_{r,+}, w_{r,-}$ be the weights from Eq. (10), $\pi_r = w_{r,+} + w_{r,-}$ their sum, and $\alpha_r = K^{-1} \#\{k : \rho_k a_r > |\delta|\}$ the fraction of radii for which the pair straddles the boundary. Then

$$n^\top D_j = -\sum_{r=1}^{q} \pi_r a_r \tanh\left(\frac{J\alpha_r}{2\varepsilon} \right) \leq 0. \tag{20}$$

Each term combines the pair's weight, its alignment with the normal, and its cost contrast. At $\delta = 0$, the no-tie assumption gives $\alpha_r = 1$ and $\pi_r = 1/q$, recovering PolyStep's planar-jump formula (Le, 2026). Away from the boundary, only straddling radii contribute to the cost difference within a pair. An orientation-independent sufficient condition for detection is $\rho_{\max} > \sqrt{q}\,|\delta|$, provided all probes remain in the two-decision neighborhood.

Proof of Proposition 3. Orient $u_r = \pm R_j e_r$ so that $a_r = n^\top u_r \geq 0$, choosing either sign if $a_r = 0$. Let $C_{r,+}$ and $C_{r,-}$ denote the mean probe costs on u_r and $-u_r$. At radius ρ_k, the pair is on opposite sides exactly when $\rho_k a_r > |\delta|$; otherwise the two costs agree. No equality case occurs because no probe lies on the boundary. Since the positive member always has at least the negative member's cost,

$$C_{r,+} - C_{r,-} = J\alpha_r.$$

Let $Z = \sum_r [\exp(-C_{r,+}/\varepsilon) + \exp(-C_{r,-}/\varepsilon)]$. The pair's contribution to the normal component is

$$a_r \frac{e^{-C_{r,+}/\varepsilon} - e^{-C_{r,-}/\varepsilon}}{Z} = -\pi_r a_r \frac{1 - e^{-J\alpha_r/\varepsilon}}{1 + e^{-J\alpha_r/\varepsilon}} = -\pi_r a_r \tanh\left(\frac{J\alpha_r}{2\varepsilon} \right).$$

Summing proves Eq. (20). Every π_r is positive. A summand is strictly negative exactly when $\alpha_r > 0$, which also implies $a_r > 0$. Thus strict drift is equivalent to $\rho_{\max} \max_r a_r > |\delta|$. If no pair straddles, its two weights agree, so every antipodal contribution cancels and $D_j = 0$.

The signed distance after a block-only update is $\delta + s n^\top D_j = \delta - sm$. If this is negative and the update stays in the stated neighborhood, its loss is L_-. For $\delta > 0$, this decreases the loss by J; for $\delta < 0$, it preserves the lower loss. At $\delta = 0$, strict decrease depends on the incumbent tie-breaking rule. Every pair then straddles, giving $\alpha_r = 1$ and $\pi_r = 1/q$. Haar rotations are non-tangent with probability one for $q \geq 2$.

Finally, $\sum_r a_r^2 = 1$ implies $\max_r a_r \geq 1/\sqrt{q}$, proving the sufficient radius. This worst-orientation threshold is sharp: an orthogonal orientation can give $a_r = 1/\sqrt{q}$ for all r. The lower threshold $\rho_{\max} \leq |\delta|$ prevents strict straddling for every orientation, apart from excluded ties. Intermediate radii depend on alignment with the boundary normal. □

Detection without an immediate decision change. Take $n = (3/5, 4/5)$, $R = I$, radii $(1/2, 1)$, $\delta = 9/20$, $J = 2\log 3$, and $\varepsilon = 1$. Only the larger radius straddles the boundary in each pair, so $\alpha_1 = \alpha_2 = 1/2$. After subtracting L_-, each positive–negative pair has costs $(2\log 3, \log 3)$ and unnormalized weights $(1/9, 1/3)$. Normalizing all four weights gives $(1/8, 3/8)$ in each pair. Therefore $D = (-1/4, -1/4)$ and $m = 7/20$. A unit step leaves the predictor on the higher-cost side because $\delta - m = 1/10$. A step $s > 9/7$ crosses, provided it stays in the two-decision neighborhood.

The boundary model arises when two distinct feasible decisions exchange optimality under an affine predictor, other decisions are strictly worse throughout the neighborhood, and their true task costs differ. Orienting the normal toward the higher-cost decision gives the stated loss. A batch also has this form if just one instance changes, or if several instances share a boundary with a nonzero aggregate jump. The next result handles different boundaries and conflicting costs.

A counterexample with three decisions. An update can increase the task loss when its probes cross more than one decision boundary. Let $\mathcal{F} = \{e_1, e_2, e_3\}$, minimize predicted coefficients, and evaluate the selected decision under true costs. Take

$$\hat{c}(y) = (0, -y_1 + 4/5, -y_1/2 + 31/100), \qquad c = (1, 0, 2), \qquad y = (0, 0).$$

Use orthoplex directions $(e_1, -e_1, e_2, -e_2)$, $R = I$, $K = 1$, and probe radius 1. At $y_1 = 0$, e_1 is uniquely optimal and costs 1; at the positive unit probe e_2 is uniquely optimal and costs 0; the other three probes select e_1 and cost 1. With $\varepsilon = 1/\log 3$, the weights are $(1/2, 1/6, 1/6, 1/6)$ and $D = (1/3, 0)$. Taking $s = 21/10$ gives $y_1^+ = 7/10$, where e_3 is uniquely optimal and costs 2. Thus all probe costs are at most the incumbent cost, yet the barycentric step increases the true cost. The lower-cost decision region at $y_1 = 1$ is separated from the incumbent by a worse region.

Why the crossing guarantee is block-only. Two individually improving block moves can interact badly. Take $P = q = 2$, $\mathcal{F} = \{e_1, e_2, e_3, e_4\}$, predicted costs $(0, 1/4 - y_1, 1/4 - y_3, 1/2 - y_1 - y_3)$, and true costs $(1, 0, 0, 2)$. At $y = 0$ the loss is 1. Unit orthoplex probes in either block have costs $(0, 1, 1, 1)$. With $\varepsilon = 1/\log 3$, each block direction is $(1/3, 0)$. Either block-only unit step reaches cost 0, but their simultaneous step reaches $y = (1/3, 0, 1/3, 0)$ and selects e_4, with cost 2. The single-boundary argument for a fixed block therefore cannot be applied to all blocks at once.

A.4 How instance costs combine

An instance may favor a direction that is poor for the rest of the batch. The relevant quantity is therefore its contribution to the mean cost difference across each probe pair. Let $\mathcal{L}_i$ denote instance i's loss in search coordinates. The next result expresses the update in terms of these differences, then gives a sufficient condition for the instances to agree.

Proposition 4 (Mean-cost signal)**.** *Fix a block, orthonormal directions $u_r = R_j e_r$, positive radii, and $\varepsilon > 0$. Let $\mathbb{E}_i$ denote either a finite weighted average over instances or a population expectation. Assume all queried losses are measurable and integrable, and define*

$$C_{r,\pm} = \mathbb{E}_i \frac{1}{K} \sum_k \mathcal{L}_i(y \pm \rho_k E_j u_r), \qquad \Delta_r = C_{r,+} - C_{r,-}.$$

With $\pi_r = w_{r,+} + w_{r,-} > 0$, Eq. (15) holds. In particular, $D_j = 0$ if and only if every $\Delta_r = 0$.

Suppose, additionally, that each instance has a local two-value loss $\mathcal{L}_i(y + E_j h) = a_i + J_i \mathbf{1}\{\delta_i + n_i^\top h > 0\}$, where the instance parameters are measurable, $J_i \geq 0$, $\|n_i\|_2 = 1$, all probes lie in the stated neighborhoods, and probe ties occur with zero instance probability. If $\mathbb{E}_i J_i < \infty$, then

$$\Delta_r = \mathbb{E}_i\big[J_i \operatorname{sgn}(n_i^\top u_r)\alpha_{ir}\big], \qquad \alpha_{ir} = \frac{1}{K}\sum_k \mathbf{1}\{\rho_k |n_i^\top u_r| > |\delta_i|\}. \tag{21}$$

If $(n_i^\top u_r)\Delta_r \geq 0$ for every r and almost every instance with $J_i > 0$, then $n_i^\top D_j \leq 0$ for those instances. A block-only step along D_j cannot increase their mean loss if its segment remains in the neighborhoods and its endpoints have zero probability of ties.

Proof. Within each antipodal pair, divide the difference of its weights by their sum:

$$w_{r,+} - w_{r,-} = \pi_r \frac{e^{-C_{r,+}/\varepsilon} - e^{-C_{r,-}/\varepsilon}}{e^{-C_{r,+}/\varepsilon} + e^{-C_{r,-}/\varepsilon}} = -\pi_r \tanh\left(\frac{\Delta_r}{2\varepsilon}\right).$$

Multiplying by u_r and summing proves Eq. (15). Since the u_r form an orthonormal basis, the sum vanishes exactly when each coefficient vanishes, equivalently when every $\Delta_r = 0$.

For an instance and radius, the difference of the two indicators is $\operatorname{sgn}(n_i^\top u_r)\mathbf{1}\{\rho_k |n_i^\top u_r| > |\delta_i|\}$: it is zero when both probes are on the same side and has the stated sign when they straddle. Averaging proves Eq. (21); integrability justifies exchanging the finite sum and expectation. Finally,

$$n_i^\top D_j = -\sum_r \pi_r (n_i^\top u_r) \tanh\left(\frac{\Delta_r}{2\varepsilon}\right) \le 0$$

under the sign condition. Along $h = sD_j$, $s \ge 0$, each signed distance is nonincreasing. Hence each indicator, and then its mean cost, cannot increase between the specified endpoints. □

Agreement and cancellation. The sign condition in Proposition 4 requires each pair's aggregate preference to agree with every affected instance's orientation. It guarantees that almost every instance's cost is nonincreasing. A decrease in the mean can also occur through tradeoffs between instances, so this sufficient condition is stronger than mean-cost improvement alone.

When all $n_i = n$, orient each u_r so $n^\top u_r \ge 0$. Then $\Delta_r = \mathbb{E}_i[J_i \alpha_{ir}] \ge 0$, and $n^\top D_j < 0$ exactly when $\Delta_r > 0$ for some r. If the offsets are also equal, Proposition 3 follows with jump $J = \mathbb{E}_i J_i > 0$. In a batch of size B, a lone changing instance contributes J_i/B to the aggregate jump. Its weight contrast is therefore governed by $J_i/(B\varepsilon)$, which can be small even for a substantial individual cost change.

For conflicting instances, take losses $\mathcal{L}_1(h) = \mathbf{1}\{n^\top h > 0\}$ and $\mathcal{L}_2(h) = a\mathbf{1}\{n^\top h < 0\}$, away from ties. Their equally weighted mean is $a/2 + (1-a)\mathbf{1}\{n^\top h > 0\}/2$. At $a = 1$ all mean probe costs agree even though both decisions change; at $a = 3$ the mean favors the opposite side to instance 1. These are finite linear decision problems: predicted costs $(-n^\top h, n^\top h)$ choose e_1 on the positive side, and true costs $(1, 0)$ and $(0, a)$ give the stated losses. A minibatch containing only instance 1 can thus point against the full-data direction. Even without cancellation, the three-decision example above rules out a general loss-decrease claim.

Population means need not have plateaus. Finite-batch margins do not imply a positive population margin. For $Z \sim \mathrm{Unif}[-1, 1]$, predicted costs $(Z - y, 0)$ and true costs $(1, 0)$ give loss $\mathbf{1}\{y > Z\}$ away from ties. At $y = 0$, each instance has positive margin $|Z|$ almost surely, yet the population mean is $(1+y)/2$ for $|y| < 1$. A positive essential lower bound on the margins suffices to preserve almost every decision throughout a common ball; positive individual margins alone do not. For bounded measurable losses, population averaging also commutes with the smooth convolution used below. This yields a smoothed population target when batches are sampled freshly from that population; empirical training alone gives no population or generalization guarantee.

A.5 Momentum at a boundary or on a plateau

Momentum reuses directions computed at earlier points. This can sustain motion when the current probes give equal costs, but those earlier directions may no longer favor the current decision. Equation (12) stores a displacement, with no factor $1 - \mu_t$ on the new input. Unrolling it shows exactly how much of each past displacement remains:

$$v_{t+1} = \sum_{k=0}^{t} s_k D_k \prod_{l=k+1}^{t} \mu_l, \qquad \|v_{t+1}\|_2 \le \sqrt{P} \sum_{k=0}^{t} s_k \prod_{l=k+1}^{t} \mu_l, \tag{22}$$

where an empty product is one. Both statements follow by substitution and $\|D_k\|_2 \le \sqrt{P}$. If $s_k \le s_{\max}$ and $\mu_k \le \bar{\mu} < 1$, the geometric series bounds the actual step by $\eta s_{\max}\sqrt{P}/(1-\bar{\mu})$. Thus the scale of the current step alone does not bound motion with momentum.

For the isolated boundary in Proposition 3, write $v_{j,t} = E_j^\top v_t$, $m_t = -n^\top D_{j,t}$, and $a_t = -n^\top v_{j,t}$. A block-only momentum step has new signed distance

$$\delta_{t+1} = \delta_t - \eta(\mu_t a_t + s_t m_t). \tag{23}$$

It reaches the lower side exactly when $\eta(\mu_t a_t + s_t m_t) > \delta_t$, provided the update remains in the two-value neighborhood. A detected improvement gives $m_t > 0$, but a sufficiently negative a_t can reverse the move. For example, $\mu_t = 1/2$, $a_t = -1$, $s_t = 1$, and $m_t = 1/4$ give a displacement toward the higher-cost side. Conversely, stored velocity toward the lower-cost side can carry the update across even if $m_t = 0$.

If $D_t, \dots, D_{t+H-1} = 0$ and $\mu_t = \mu \in [0, 1)$ throughout these steps, direct summation gives

$$y_{t+H} - y_t = \eta \frac{\mu(1 - \mu^H)}{1 - \mu} v_t. \tag{24}$$

The distance traveled on a plateau is thus limited by the incoming velocity. With $v_t = 0$ the trajectory remains fixed. Resetting velocity when the basis changes ends this accumulation. In particular, a reset after every step reduces the next step to $y_{t+1} = y_t + \eta s_t D_t$ in its current coordinates. The finite-horizon theorem below assumes fixed coordinates; refreshed bases need a separate analysis even when velocities are reset.

A.6 Cost errors and sampling

We ask how accurately candidate costs must be evaluated to preserve a directional signal. A common error across a row cancels, so the relevant error is measured after removing such an offset. The following bound is the finite softmax form of the sharp total-variation inequality of Cohen & Fausti (2023); we give an elementary proof below.

Proposition 5 (Cost-error stability)**.** *For $\varepsilon > 0$, $\beta \geq 0$, and unit directions u_v, set $D(C) = \sum_v \mathrm{softmax}(-C/\varepsilon)_v u_v$. If $\|\widehat{C} - C - b\mathbf{1}\|_\infty \leq \beta$ for some scalar b, then*

$$\|D(\widehat{C}) - D(C)\|_2 \leq 2 \tanh\left(\frac{\beta}{2\varepsilon}\right) \leq \min\{2, \beta/\varepsilon\}. \tag{25}$$

If $\|n\|_2 = 1$ and an exact direction has $n^\top D(C) = -m < 0$, its noisy counterpart still points toward the lower side whenever $2\tanh(\beta/(2\varepsilon)) < m$; the simpler condition $\beta < \varepsilon m$ suffices. The first bound is attained by two antipodal directions and suitable two-entry cost rows.

Lower temperature strengthens a reliable cost preference and also amplifies errors in that preference. The ratio β/ε quantifies this tradeoff. Here β concerns realized task costs. A solver's gap under predicted coefficients measures a different quantity, as the counterexample below shows.

Proof of Proposition 5. Put $e = \widehat{C} - C - b\mathbf{1}$, $p = \mathrm{softmax}(-C/\varepsilon)$, and $\widehat{p} = \mathrm{softmax}(-\widehat{C}/\varepsilon)$. The common factor from b cancels, so with $X_v = \exp(-e_v/\varepsilon)$ and $m = \sum_v p_v X_v$ we have $\widehat{p}_v = p_v X_v / m$. Thus the problem is to bound the change in a probability vector after multiplying each entry by a factor in a fixed interval.

If $\beta = 0$, the weights agree. Otherwise let $a = \exp(-\beta/\varepsilon)$ and $b_+ = \exp(\beta/\varepsilon)$, so $a \leq X_v \leq b_+$ and $a \leq m \leq b_+$. For $x \in [a, b_+]$, convexity gives the explicit chord bound

$$|x - m| \leq \frac{b_+ - x}{b_+ - a}(m - a) + \frac{x - a}{b_+ - a}(b_+ - m).$$

Substitute $x = X_v$, multiply by p_v, and sum. Since $\sum_v p_v X_v = m$, this yields

$$\sum_v p_v |X_v - m| \leq \frac{2(b_+ - m)(m - a)}{b_+ - a}.$$

Dividing by m and using $m + ab_+/m \geq 2\sqrt{ab_+}$ yields

$$\|\widehat{p} - p\|_1 \leq \frac{2(a + b_+ - m - ab_+/m)}{b_+ - a} \leq 2\frac{\sqrt{b_+} - \sqrt{a}}{\sqrt{b_+} + \sqrt{a}} = 2\tanh\left(\frac{\beta}{2\varepsilon}\right).$$

The triangle inequality and $\|u_v\|_2 = 1$ bound the direction error by this ℓ_1 distance. Since $\tanh t \le \min\{1, t\}$ for $t \ge 0$, the linear and diameter bounds follow. Taking the inner product with a unit normal proves the drift claim. Sharpness follows with $u_1 = -u_2$ unit, $C = (-\beta/2, \beta/2)$, and $\widehat{C} = (\beta/2, -\beta/2)$: the two directions differ by exactly $2\tanh(\beta/(2\varepsilon))$. This is the finite softmax case of Cohen & Fausti (2023). □

Minibatches versus exact mean costs. Condition on the current parameters, rotations, and radii, chosen independently of the current batch. Let $C^\star$ contain the exact mean costs, either over the training set or over a population, and let $\widehat{C}$ use B i.i.d. instances shared across all probes. Suppose each queried instance loss lies in a common interval of width W. Then, for $0 < \alpha < 1$, with conditional probability at least $1 - \alpha$,

$$\max_{j,v} |\widehat{C}_{jv} - C^\star_{jv}| \le \beta_B := W\sqrt{\frac{\log(2PV/\alpha)}{2B}}, \qquad \|D_j(\widehat{C}) - D_j(C^\star)\|_2 \le 2\tanh\left(\frac{\beta_B}{2\varepsilon}\right). \tag{26}$$

To see this, average the K radii within each instance first. Each resulting variable still lies in an interval of width W, and the B variables for a fixed entry are independent. Hoeffding's inequality (Hoeffding, 1963) gives entrywise failure probability at most $2\exp(-2B\beta_B^2/W^2)$; a union bound over the PV entries proves the first bound. Proposition 5 proves the second. Shared instances create dependence between entries, which the union bound does not require us to remove. For $W = 0$, all errors are zero.

The conditional expected direction error in each block is consequently at most

$$\chi_B := \min\left\{2,\ 2\tanh\left(\frac{\beta_B}{2\varepsilon}\right) + 2\alpha\right\}. \tag{27}$$

On the success event use Eq. (26); on failure the distance between two convex combinations of unit vectors is at most two. A full-mean normal drift $-m < 0$ is therefore preserved on the success event whenever $2\tanh(\beta_B/(2\varepsilon)) < m$. This quantifies when a batch represents the mean signal. It does not imply unbiased softmax weights, or that noise always shortens the step.

For a concrete comparison, take directions ± 1, $B = 1$, and $\varepsilon = 1/\log 3$, so $D(C) = -\tanh((C_+ - C_-)/(2\varepsilon))$. If the instance cost row is equally likely to be $(0, 0)$ or $(2, 0)$, then $D(\mathbb{E}C) = -1/2$ but $\mathbb{E}D(C) = -2/5$: sampling weakens the preference. If instead the row is $(0, 8)$ with probability $1/10$ and $(2, 0)$ with probability $9/10$, then $D(\mathbb{E}C) = -1/2$ again, whereas $\mathbb{E}D(C) = \tanh(4\log 3)/10 - 18/25 < -31/50$. Sampling now strengthens it. Both examples have unbiased probe costs.

Simulation estimates. Suppose each of Q probe evaluations averages M independent, conditionally identically distributed simulation costs in an interval of width W, with means equal to their target losses. Conditional on the candidate queries, Hoeffding's inequality and a union bound give simultaneous error at most

$$\beta = W\sqrt{\frac{\log(2Q/\alpha)}{2M}} \tag{28}$$

with probability at least $1 - \alpha$, for $0 < \alpha < 1$. Independence across candidates is unnecessary; independence within each estimate suffices. Averaging these evaluations along a direction does not enlarge their maximum error. The bound assumes that the simulation estimates are unbiased for the target costs.

A solver-gap counterexample. For $\mathcal{F} = \{e_1, e_2\}$, let the predicted costs be $(0, \eta)$ with $\eta > 0$, and the true costs $(0, M)$ with any $M > 0$. Returning e_2 instead of the exact predicted optimum e_1 incurs predicted additive gap η but changes the realized cost by M. Since M is arbitrary, the predicted optimality gap alone does not bound downstream cost error. Such a bound requires additional assumptions linking predicted and true costs.

A.7 Smoothing and the explicit bias

The boundary results explain individual updates, but their isolation assumptions need not hold along a training run. For repeated steps, we measure progress on a spatial average of the mean loss.

This gives a notion of stationarity even when the unsmoothed decision loss is discontinuous. Write $\mathcal{L}(y) = L(\theta(y))$ for the empirical objective. The same argument applies to a bounded population mean with fresh population samples. Throughout, A is fixed and gradients are with respect to y, so the guarantee concerns the chosen search space.

The target and bound. The smoothing scale determines which nearby decision changes the target averages together. Define

$$F(y) = \mathbb{E}_\xi[\mathcal{L}(y + \xi)], \qquad \xi = (\xi_1, \dots, \xi_P), \tag{29}$$

where each block ξ_j is drawn independently from a mixture of uniform distributions on balls centered at zero. For fixed probe radii, the ball of radius ρ_k has mixture weight $\rho_k / \sum_u \rho_u$. This radius weighting follows from the relation between spherical probes and derivatives of ball averages. We derive this relation and define the random-radius case below.

The analysis compares the actual softmax direction with its linear approximation; the algorithm continues to use the exact weights. For each block, averaging the linear term over rotations gives a scaled negative gradient of the loss smoothed in that block. Nonlinear weighting, block interactions, and cost errors account for the remaining bias. Under the sampling and regularity conditions below, the complete direction $D_t = (D_{1,t}, \dots, D_{P,t})$ satisfies

$$\mathbb{E}[D_t \mid \mathcal{H}_t] = -\gamma \nabla F(y_t) + e_t, \qquad \|e_t\|_2 \le b, \qquad \gamma = \frac{\bar{\rho}}{q\varepsilon}, \tag{30}$$

where $\mathcal{H}_t$ is the history before iteration t and $\bar{\rho}$ is the mean probe radius. For deterministic radii, $\bar{\rho} = K^{-1} \sum_k \rho_k$. Let G_j smooth only block j. If σ_j^2 bounds the conditional second moment of the ideal cost-row spread, κ_j bounds $\|\nabla_j G_j - \nabla_j F\|_2$, and β_j bounds the cost error after a row shift, then

$$b^2 = \sum_{j=1}^{P} \left(\underbrace{\frac{\sigma_j^2}{4\varepsilon^2}}_{\text{softmax}} + \underbrace{\gamma \kappa_j}_{\text{block discrepancy}} + \underbrace{2 \tanh(\beta_j/(2\varepsilon))}_{\text{cost error}} \right)^2 . \tag{31}$$

The block term appears because a probe changes one block while F averages all blocks. The other terms bound the departure from linear cost weights and the error in evaluated costs. When $\|\nabla F(y_t)\|_2 > b/\gamma$, the gradient contribution dominates these errors and the expected direction is a descent direction for F. Smooth descent then gives the following bound (Ghadimi & Lan, 2013).

Theorem 6 (Finite-horizon bound with bias)**.** *Assume a deterministic initialization y_0, $F \ge F_{\inf}$, F is differentiable with L_F-Lipschitz gradient on a convex set containing the iterates and update segments, and Eq.* (30) *holds almost surely with fixed $\gamma > 0$ and $b \ge 0$. Let D_t be the concatenated block directions from Eq.* (10)*, so $\|D_t\|_2 \le \sqrt{P}$. Run $y_{t+1} = y_t + sD_t$ for T steps with $s = a/\sqrt{T}$, $a > 0$. For an independently uniform index $\tau \in \{0, \dots, T-1\}$,*

$$\mathbb{E}\|\nabla F(y_\tau)\|_2^2 \le \frac{2(F(y_0) - F_{\inf})}{\gamma a \sqrt{T}} + \frac{L_F P a}{\gamma \sqrt{T}} + \frac{b^2}{\gamma^2}. \tag{32}$$

With constants independent of T, the first two terms decay as $T^{-1/2}$; persistent direction error leaves the residual b^2/γ^2. This residual limits what the bound certifies; actual gradients may be smaller. A small gradient describes the sensitivity of the spatial average and does not certify optimal decisions. We derive the bias next, then extend the bound to momentum in Theorem 8. Both results use a fixed search map, temperature, and probe law, conditionally unbiased batches, and independent rotations. The smoothness assumption is substantive: the radius randomization below suffices, while fixed radii alone need not. Annealing, refreshed subspaces, and fixed-order epochs require separate arguments.

Sampling assumptions. Assume $q \ge 2$ and centered unit vertices. At each iteration, draw the batch, Haar rotations, and any randomized radii independently of one another and of the past, using fixed sampling laws. Rotations are independent across blocks; radii may share a common random multiplier. Require $\mathbb{E}[\widehat{\mathcal{L}}_{\mathcal{B}}(y) \mid \mathcal{H}_t] = \mathcal{L}(y)$ for every y. Uniform sampling with replacement from the training data suffices. The same batch is used across candidates. Assume these random functions are jointly measurable and uniformly bounded on all required neighborhoods. Sampling without replacement within an epoch requires a separate argument for conditional unbiasedness.

The smoothing distribution. Let B_q and $\mathbb{S}^{q-1}$ denote the unit ball and unit sphere in $\mathbb{R}^q$. Let ν be the law of a probe radius after choosing k uniformly from $1, \dots, K$, independently of rotations. Assume its support lies in $[\rho_{\min}, \rho_{\max}]$ with $0 < \rho_{\min} \le \rho_{\max} < \infty$. For fixed radii, $\nu = K^{-1} \sum_k \delta_{\rho_k}$ and $\bar{\rho} = \mathbb{E}_\nu \rho$. Define a block smoothing operator

$$(\mathcal{K}_j h)(y) = \int \frac{\rho}{\bar{\rho}} \, \mathbb{E}_{U \sim \mathrm{Unif}(B_q)}[h(y + \rho E_j U)] \, \nu(d\rho), \qquad G_j = \mathcal{K}_j \mathcal{L}, \quad F = \mathcal{K}_1 \cdots \mathcal{K}_P \mathcal{L}. \tag{33}$$

The mixture weights integrate to one. With fixed radii, they are $\rho_k/(K\bar{\rho})$, giving the joint average in Eq. (29). The use of ball averages follows from the spherical differentiation identity (Flaxman et al., 2005), derived below.

For a bounded measurable loss and fixed radii, the following differentiation identity holds for almost every y:

$$\mathbb{E}_{U \sim \mathrm{Unif}(\mathbb{S}^{q-1}), \, \rho \sim \nu}[\mathcal{L}(y + \rho E_j U) U] = \frac{\bar{\rho}}{q} \nabla_j G_j(y). \tag{34}$$

To use this identity along an optimization trajectory, it must hold at the iterates. A sufficient condition for pointwise validity is smooth radius randomization: multiply every ρ_k in a step by a common independent $1 + \zeta$, where ζ has an even C^∞ density compactly supported in $(-h, h)$, $0 < h < 1$. The mean radii are unchanged. If p_ν is the resulting radius density and ω_q is the unit-ball volume, the block kernel has density

$$k(u) = \frac{1}{\omega_q \bar{\rho}} \int_{\|u\|_2}^{\infty} p_\nu(r) r^{1-q} \, dr.$$

It is smooth, compactly supported, and constant near the origin. Thus Eq. (34) holds pointwise, and the product kernel defining F has bounded derivatives for a bounded loss. For other radius distributions, the theorem requires pointwise validity at the iterates and the stated smoothness of F. A shared probe-radius multiplier is sufficient because the expected linear term uses only marginal expectations; F is still defined using independent block kernels.

Proof of Eq. (34). First take a smooth h. The divergence theorem applied to the q-ball gives

$$\nabla_j \mathbb{E}_{U \sim \mathrm{Unif}(B_q)} h(y + \rho E_j U) = \frac{q}{\rho} \mathbb{E}_{U \sim \mathrm{Unif}(\mathbb{S}^{q-1})} h(y + \rho E_j U) U.$$

Multiplication by $\rho/\bar{\rho}$ and integration over ν give the identity. For bounded measurable h, approximation by smooth functions extends it as a weak derivative, hence almost everywhere.

The factor q/ρ is the surface-to-volume ratio: a ball of radius ρ has volume $\omega_q \rho^q$ and surface area $q\omega_q \rho^{q-1}$. This also explains the radius weights in Eq. (33). Weighting the ball averages by $\rho/\bar{\rho}$ cancels the $1/\rho$ factor and recovers the uniform average over the K probe radii.

For smooth radius randomization, a direct calculation proves pointwise validity even for a discontinuous loss. The kernel density above satisfies

$$\nabla k(u) = -\frac{p_\nu(\|u\|_2)}{\omega_q \bar{\rho} \, \|u\|_2^q} \, u \quad (u \neq 0).$$

Differentiating the convolution gives $\nabla_j G_j(y) = -\int \mathcal{L}(y + E_j u) \nabla k(u) \, du$. Polar integration, with sphere area $q\omega_q$, yields the right coefficient $q/\bar{\rho}$ in Eq. (34). Boundedness of the loss and the smooth, compactly supported kernel justify differentiation under the integral. The product kernel has the same properties, so its second derivatives are integrable and F has a bounded Hessian on the required neighborhoods. □

The ball identity involves costs multiplied by directions, whereas the algorithm uses exponential weights. The next lemma bounds the difference. It explains why cost spread relative to temperature controls the softmax contribution to the bias.

Lemma 7 (Softmax linearization with a global remainder)**.** *For $\varepsilon > 0$ and centered unit directions u_v, let $D(C) = \sum_v \mathrm{softmax}(-C/\varepsilon)_v u_v$ and $\Delta = \max_v C_v - \min_v C_v$. Then*

$$\left\| D(C) + \frac{1}{V\varepsilon} \sum_v C_v u_v \right\|_2 \le \frac{\Delta^2}{4\varepsilon^2}. \tag{35}$$

The bound holds at every spread; it can be large for low temperatures.

Proof. Put $z_v = -C_v/\varepsilon$, $p_v(t) = \mathrm{softmax}(tz)_v$, and $d(t) = \sum_v p_v(t)u_v$, for $0 \le t \le 1$. This interpolates between uniform weights and the actual softmax weights. Write $\mu(t) = \sum_v p_v(t)z_v$. Differentiating gives

$$p_v'(t) = p_v(t)(z_v - \mu(t)), \qquad \mu'(t) = \sum_v p_v(t)(z_v - \mu(t))^2 = \mathrm{Var}_{p(t)}\, z.$$

Hence $d(0) = 0$ by centering, $d'(0) = V^{-1}\sum_v z_v u_v$, and

$$p_v''(t) = p_v(t)\big[(z_v - \mathbb{E}_{p(t)} z)^2 - \mathrm{Var}_{p(t)}\, z\big].$$

The unit-vector bound and the triangle inequality give

$$\|d''(t)\|_2 \le \sum_v p_v(t)\big[(z_v - \mu(t))^2 + \mathrm{Var}_{p(t)}\, z\big] = 2\,\mathrm{Var}_{p(t)}\, z.$$

For $z_{\mathrm{mid}} = (\max z + \min z)/2$, the variance is at most $\mathbb{E}_{p(t)}(z - z_{\mathrm{mid}})^2 \le (\max z - \min z)^2/4$. Taylor's formula with integral remainder now yields

$$\|d(1) - d'(0)\|_2 \le \int_0^1 (1-t)\|d''(t)\|_2\, dt \le \frac{\Delta^2}{4\varepsilon^2}.$$

□

Assume Eq. (34) is pointwise valid at the iterates, and let $C^0_{j:}$ denote ideal batch probe costs before any additional evaluation error. Suppose, uniformly in t, almost surely,

$$\begin{aligned} \mathbb{E}[(\max_v C^0_{jv} - \min_v C^0_{jv})^2 \mid \mathcal{H}_t] &\le \sigma_j^2, \\ \|\nabla_j G_j(y_t) - \nabla_j F(y_t)\|_2 &\le \kappa_j, \\ \inf_{b_j \in \mathbb{R}} \|\widehat{C}_{j:} - C^0_{j:} - b_j \mathbf{1}\|_\infty &\le \beta_j. \end{aligned} \tag{36}$$

These are bounds on the ideal cost spread, the block discrepancy, and the evaluation error after a common shift. They imply the bias model with the constant in Eq. (31).

Derivation of Eqs. (30) *and* (31). Each rotated vertex is uniform on the sphere. Conditional unbiasedness of the batch loss and Eq. (34) therefore give

$$\mathbb{E}\left[-\frac{1}{V\varepsilon}\sum_v C^0_{jv} R_j v_v \,\middle|\, \mathcal{H}_t\right] = -\frac{1}{\varepsilon}\mathbb{E}_{\rho,U}[\mathcal{L}(y_t + \rho E_j U)U] = -\gamma \nabla_j G_j(y_t).$$

Lemma 7 bounds the norm of the expected remainder by $\sigma_j^2/(4\varepsilon^2)$. Replacing $\nabla_j G_j$ by $\nabla_j F$ adds at most $\gamma\kappa_j$. Proposition 5 bounds the effect of using $\widehat{C}$ by $2\tanh(\beta_j/(2\varepsilon))$. Adding these errors within each block and summing their squares across the orthogonal blocks gives Eqs. (30) and (31). □

For $g_t = \nabla F(y_t)$, the bias bound has the direct interpretation

$$\langle g_t, \mathbb{E}[D_t \mid \mathcal{H}_t]\rangle \le -\gamma\|g_t\|_2^2 + b\|g_t\|_2 = -\gamma\|g_t\|_2\left(\|g_t\|_2 - \frac{b}{\gamma}\right).$$

This proves the descent threshold stated above. The next proof controls the additional error from taking a finite step.

Proof of Theorem 6. Each block direction is a convex combination of unit vectors, so $\|D_t\|_2^2 \le P$. Smoothness along the update segment and Eq. (30) give, with $g_t = \nabla F(y_t)$,

$$\begin{aligned} \mathbb{E}[F(y_{t+1}) \mid \mathcal{H}_t] &\le F(y_t) - s\gamma\|g_t\|_2^2 + s\langle g_t, e_t\rangle + \tfrac{1}{2}L_F s^2 P \\ &\le F(y_t) - \tfrac{1}{2}s\gamma\|g_t\|_2^2 + \frac{sb^2}{2\gamma} + \tfrac{1}{2}L_F s^2 P. \end{aligned}$$

For the second line, Cauchy–Schwarz gives $\langle g_t, e_t \rangle \le b\|g_t\|_2$, and the required scalar bound follows from a square:

$$\frac{\gamma}{2}\|g_t\|_2^2 + \frac{b^2}{2\gamma} - b\|g_t\|_2 = \frac{(\gamma\|g_t\|_2 - b)^2}{2\gamma} \ge 0.$$

Summing expectations over $t < T$ makes the objective differences telescope:

$$\frac{s\gamma}{2}\sum_{t=0}^{T-1}\mathbb{E}\|g_t\|_2^2 \le F(y_0) - F_{\inf} + \frac{Tsb^2}{2\gamma} + \frac{TL_F s^2 P}{2}.$$

Divide by $s\gamma T/2$ and substitute $s = a/\sqrt{T}$. The independent uniform choice of τ turns the average on the left into $\mathbb{E}\|\nabla F(y_\tau)\|_2^2$, proving Eq. (32). For a random initialization with finite expected loss, replace $F(y_0)$ by its expectation. □

The bound separates the finite-iteration term from the persistent direction error. Reducing the smoothing radius may increase L_F near a discontinuity, while lowering the temperature increases the error bounds in Eq. (31). A convergence analysis for decreasing radii or temperatures must control these changes jointly. The derivation uses the cost rows in Eq. (9): a fixed positive cost scale can be absorbed into ε, whereas data-dependent normalization requires its own conditional bias bound.

Interpreting the bias. For one block, $G_1 = F$ and $\kappa_1 = 0$. With several blocks, G_j averages only the active block, whereas F averages all blocks. Their gradients can therefore differ. Minibatch randomness also contributes to σ_j^2, even when the batch loss is unbiased. Alternatively, under Eq. (26), linearize the exact mean row $C^\star$ instead: use its conditional spread bound for σ_j^2 and add χ_B from Eq. (27) to each block's bias bound. This separates the softmax approximation from sampling error. These are alternative bounds; the sampling error need not be counted twice.

If $\mathcal{L}(y) = \sum_j h_j(y_j)$ is additively separable across blocks, all κ_j can also be zero: smoothing the other blocks changes only terms that are constant with respect to y_j. For a coupled loss, separate block averages need not have the partial derivatives of a common objective. The discrepancy term accounts for this difference.

With one block and exact evaluations, the normalized bias bound is $b/\gamma = q\sigma_1^2/(4\varepsilon\bar{\rho})$. Increasing the temperature reduces this bound but also reduces the gradient coefficient γ. With several blocks, the discrepancy term need not vanish with temperature or iteration count.

For simulation estimates, Eq. (28) gives a high-probability error bound rather than the almost-sure bound in Eq. (36). If its failure probability is at most α conditional on the history and queries, the expected direction error is at most $\min\{2, 2\tanh(\beta_j/(2\varepsilon)) + 2\alpha\}$. Use this quantity in place of the cost-error term in Eq. (31). On the success event, Proposition 5 applies; on failure, the direction difference is at most two.

Block discrepancy and decision changes. Under smooth radius randomization, let ξ_{-j} have the product smoothing law in all blocks except j, with a zero jth block. Commuting the other block averages with the active-block derivative gives

$$\|\nabla_j G_j(y) - \nabla_j F(y)\|_2 \le \frac{q}{\bar{\rho}}\,\mathbb{E}\big|\mathcal{L}(y + \rho E_j U + \xi_{-j}) - \mathcal{L}(y + \rho E_j U)\big|,$$

where U is uniform on the sphere and all variables in this expectation are independent. Apply Eq. (34) before and after the other block averages, then use $\|U\|_2 = 1$. For the finite empirical decision-only loss with range width W, the expectation is at most W times the probability that ξ_{-j} changes at least one instance's returned decision. Rare changes therefore suffice for a small discrepancy. Frequent changes can also cancel, so this probability bound need not be tight.

Why fixed radii require a regularity assumption. Consider a two-decision OR problem with $\mathcal{F} = \{e_1, e_2\}$, predicted costs $(\|y\|_2^2 - 1, 0)$, true costs $(1, 0)$, and ties assigned to e_1. Its loss is $\mathcal{L}(y) = \mathbf{1}\{\|y\|_2 \le 1\}$. For $P = 1$, $q = 2$, and radius 1, the ball average is the normalized overlap of two unit disks. This overlap consists of two equal circular segments. Writing $r = \|y\|_2 \le 2$ and integrating their cross-sections gives

$$F(y) = \frac{4}{\pi}\int_{r/2}^{1}\sqrt{1 - t^2}\,dt = \frac{2}{\pi}\arccos(r/2) - \frac{r}{2\pi}\sqrt{4 - r^2} = 1 - \frac{2}{\pi}r + O(r^3).$$

Hence F is not differentiable at $y = 0$, although the predictor is smooth and the decision set is finite.

A.8 Stationarity with scheduled momentum

The bias model controls the current probe direction, while momentum also contains earlier directions. We account for this memory by adding to each iterate the displacement its stored velocity would produce over the remaining steps if no new directions were added. The resulting auxiliary sequence moves only in the current probe direction, so the same descent argument applies. This change of variables follows the approach used for stochastic momentum (Yan et al., 2018); the construction below allows a deterministic coefficient schedule.

Theorem 8 (Finite-horizon bound with momentum)**.** *Run Eq.* (12) *for T steps from deterministic y_0 and $v_0 = 0$, with $s_t = s = a/\sqrt{T}$, $a > 0$, $\eta > 0$, and a deterministic schedule $0 \le \mu_t \le \bar{\mu} < 1$. Assume Eq.* (30) *holds conditional on the history including v_t, and $\|D_t\|_2 \le \sqrt{P}$. Define the deterministic coefficients and auxiliary points*

$$c_T = 0, \qquad c_t = \mu_t(1 + c_{t+1}), \qquad h_t = \eta s(1 + c_{t+1}), \qquad z_t = y_t + \eta c_t v_t. \tag{37}$$

Assume $F \ge F_{\inf}$ and F has L_F-Lipschitz gradient on a convex set containing all y_t, z_t almost surely. Choose τ independently of the run with $\Pr(\tau = t) = h_t / \sum_{k<T} h_k$. Then

$$\mathbb{E}\|\nabla F(y_\tau)\|_2^2 \le \frac{2(F(y_0) - F_{\inf})}{\gamma\eta a\sqrt{T}} + \frac{L_F P \eta a(1+\bar{\mu})}{\gamma(1-\bar{\mu})^2\sqrt{T}} + \frac{b^2}{\gamma^2}. \tag{38}$$

A linear momentum schedule is covered when its endpoints lie in $[0, \bar{\mu}]$ and the remaining assumptions hold. Momentum retains the $T^{-1/2}$ rate and the same bias residual for this smoothed objective. Its longer memory also enlarges the finite-step term, so this bound establishes no acceleration. Taking $\bar{\mu} = 0$ and $\eta = 1$ recovers Theorem 6, including its uniform output index. The auxiliary points and the random index are used only in the analysis.

Proof. The backward recurrence and a geometric sum give

$$0 \le c_t \le \frac{\bar{\mu}}{1-\bar{\mu}}, \qquad \eta s \le h_t \le \frac{\eta s}{1-\bar{\mu}}, \qquad \|v_t\|_2 \le \frac{s\sqrt{P}}{1-\bar{\mu}}.$$

Consequently $\|z_t - y_t\|_2 \le R_s := \eta s \bar{\mu}\sqrt{P}/(1-\bar{\mu})^2$. Substituting Eq. (12) into the definition of z_{t+1} cancels the stored velocity:

$$z_{t+1} = y_t + \eta(1 + c_{t+1})(\mu_t v_t + s D_t) = z_t + h_t D_t.$$

Also $z_0 = y_0$ and $z_T = y_T$. Put $g_t = \nabla F(y_t)$. Smoothness and the bound on D_t imply

$$\begin{aligned}\mathbb{E}[F(z_{t+1}) \mid \mathcal{H}_t] &\le F(z_t) + h_t\langle g_t, \mathbb{E}[D_t \mid \mathcal{H}_t]\rangle + h_t L_F R_s\sqrt{P} + \tfrac{1}{2}L_F h_t^2 P \\ &\le F(z_t) - \tfrac{1}{2}h_t\gamma\|g_t\|_2^2 + \frac{h_t b^2}{2\gamma} + h_t L_F R_s\sqrt{P} + \tfrac{1}{2}L_F h_t^2 P.\end{aligned}$$

The first line bounds the gradient difference between z_t and y_t by $L_F R_s$. The second uses the bias model and the same scalar square inequality as the proof of Theorem 6. Let $H_T = \sum_{t<T} h_t$. Summing expectations and dividing by $\gamma H_T/2$ yields

$$\sum_{t<T} \frac{h_t}{H_T}\mathbb{E}\|g_t\|_2^2 \le \frac{2(F(y_0) - F_{\inf})}{\gamma H_T} + \frac{b^2}{\gamma^2} + \frac{2L_F R_s\sqrt{P}}{\gamma} + \frac{L_F P\sum_{t<T} h_t^2}{\gamma H_T}.$$

Use $H_T \ge \eta s T$ and $\sum_t h_t^2/H_T \le \eta s/(1-\bar{\mu})$. The last two terms involving smoothness are at most

$$\frac{L_F P \eta s}{\gamma}\left(\frac{2\bar{\mu}}{(1-\bar{\mu})^2} + \frac{1}{1-\bar{\mu}}\right) = \frac{L_F P \eta s(1+\bar{\mu})}{\gamma(1-\bar{\mu})^2}.$$

Substitute $s = a/\sqrt{T}$ and the specified distribution of τ to obtain Eq. (38). □

The constants must be independent of T for the stated rate. The lower bound and smoothness are needed on the auxiliary points as well as the actual iterates; smooth radius randomization and a globally bounded loss suffice. The result controls a weighted average of squared gradient norms, and hence their minimum in expectation, but not the last iterate or the cost of its decisions. Historical directions do not remove the softmax, block, or evaluation bias. Data-dependent momentum schedules and changing search bases are not covered by this proof.

A.9 ARITHMETIC AND STORAGE COSTS

We use physical lengths in search coordinates: PolyStep's multiplier convention r_p, r_s corresponds to $r = r_p\varepsilon$ and $s = r_s\varepsilon$ (Le, 2026). Let $C_{\mathrm{pred}}(B)$ be the cost of a batch prediction, C_S the average per-instance solve-and-score cost, and C_A the cost of reconstructing one candidate. For simplex or orthoplex vertices, serial work per step is

$$O\big(Q[C_A + C_{\mathrm{pred}}(B) + BC_S] + Pq^3 + PVq\big)\,. \tag{39}$$

The last terms account for dense QR rotations and weighted directions; row softmax costs $O(PV)$. Momentum adds $O(d)$ work and storage, with no additional solver calls. A dense $D_\theta \times d$ subspace map can cost $O(D_\theta d)$ per reconstruction, while structured maps permit cheaper implementations. The cost matrix, accumulated update, and stored rotations need $O(PV + d + Pq^2)$ storage; candidate models, predictor activations, solver state, and the subspace map are additional costs. Streaming candidates avoids storing Q complete models. Batching and parallel solver calls can reduce wall-clock time. Total computational comparisons should include both training evaluations and any reference-solution preprocessing required by a baseline.

B ADDITIONAL EXPERIMENTAL MATERIAL

This appendix provides detailed information on the optimization problems, extended experimental designs and results. Finally, we provide a list of hyperparameters configurations for reproducibility of the reported experiments.

B.1 EXTENDED OR METRICS AND RESULTS

The technical details of the problems are taken from (Mandi et al., 2024).

B.1.1 PROBLEM SETUPS

We evaluate 5 problems in the benchmark of Mandi et al. (2024), following the formulations in their Section 5.1. Table 3 summarizes the settings, solvers, predictors, and data splits.

Shortest path on a 5×5 grid. Following Mandi et al. (2024) and Elmachtoub & Grigas (2022), the problem finds a minimum-cost path from the southwest corner to the northeast corner of a 5×5 grid. The grid contains 25 nodes and 40 directed edges, each pointing north or east. The optimization problem is

$$\min_{\boldsymbol{x}} \ \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad A\boldsymbol{x} = \boldsymbol{b}, \qquad \boldsymbol{x} \geq \mathbf{0},$$

where $A \in \mathbb{R}^{|V| \times |E|}$ is the node-edge incidence matrix, with $+1$ at an edge's tail and -1 at its head. The vector $\boldsymbol{b} \in \mathbb{R}^{|V|}$ has value 1 at the source, -1 at the sink, and 0 elsewhere. The LP admits an integral optimal solution, in which $x_e = 1$ indicates that edge e is traversed. The cost vector $\boldsymbol{c}_i \in \mathbb{R}^{40}$ is generated from features $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, I_p)$ with $p = 5$. For a fixed randomly generated matrix $B \in \mathbb{R}^{40 \times p}$, each cost is computed as:

$$c_{ij} = \left[\left(\frac{1}{\sqrt{p}} (B\boldsymbol{z}_i)_j + 3 \right)^{\mathrm{Deg}} + 1 \right] \xi_{ij},$$

where ξ_{ij} is drawn uniformly from $[1-\vartheta, 1+\vartheta]$, with $\vartheta = 0.5$. A linear predictor maps the features to the edge costs. The degree $\mathrm{Deg} \in \{1, 2, 4, 6, 8\}$ controls the nonlinearity of the underlying mapping and hence the misspecification of this predictor. Each setting uses 1,000 training, 250 validation, and 10,000 test instances.

Portfolio optimization. Following Elmachtoub & Grigas (2022), the problem allocates investments across m assets to maximize return subject to a risk constraint,

$$\max_{\boldsymbol{x}} \ \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{x}^\top \Sigma \boldsymbol{x} \leq \gamma, \qquad \mathbf{1}^\top \boldsymbol{x} \leq 1, \qquad \boldsymbol{x} \geq \mathbf{0}.$$

Here, x_j is the fraction invested in asset j, Σ is the covariance matrix used to measure risk, and γ is the risk limit. The features satisfy $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, I_p)$. A fixed matrix $B \in \{0, 1\}^{m \times p}$ has independent Bernoulli entries with probability 0.5. The conditional mean return is defined as:

$$\bar{c}_{ij} = \left(\frac{0.05}{\sqrt{p}} (B\boldsymbol{z}_i)_j + 0.1^{1/\mathrm{Deg}} \right)^{\mathrm{Deg}}.$$

For noise magnitude ϑ, a fixed matrix $L \in \mathbb{R}^{m \times 4}$ has entries drawn uniformly from $[-0.0025\vartheta, 0.0025\vartheta]$. The observed returns are then computed:

$$\boldsymbol{c}_i = \bar{\boldsymbol{c}}_i + L\boldsymbol{f}_i + 0.01\vartheta\boldsymbol{\xi}_i,$$

where $\boldsymbol{f}_i \sim \mathcal{N}(\mathbf{0}, I_4)$ and $\boldsymbol{\xi}_i \sim \mathcal{N}(\mathbf{0}, I_m)$ are independent of each other and of the features. Conditional on the fixed generator matrices, the return covariance given the features is $\mathrm{Cov}(\boldsymbol{c}_i \mid \boldsymbol{z}_i) = \Sigma = LL^\top + (0.01\vartheta)^2 I_m$. The risk limit is $\gamma = 2.25\, \boldsymbol{e}^\top \Sigma \boldsymbol{e}$, where $\boldsymbol{e} = \mathbf{1}/m$ is the equal-allocation portfolio. Both Σ and γ remain fixed within each setting. Our experiments use $m = 50$ assets, $p = 5$ features, $\mathrm{Deg} \in \{1, 4, 8, 16\}$, and $\vartheta = 1$. A linear predictor estimates the returns. Each setting uses 1,000 training, 250 validation, and 10,000 test instances.

Energy-cost aware scheduling. This problem schedules tasks on machines one day ahead using predicted electricity prices. Each task $j \in J$ has duration ζ_j, earliest start $\zeta_j^{(1)}$, latest end $\zeta_j^{(2)}$, power consumption ϕ_j, and resource requirements u_{jw} for $w \in W$. Machine $i \in I$ has capacity Q_{iw} for resource w. Tasks cannot be interrupted or transferred between machines. Let $T = \{0, \dots, 47\}$ index the half-hour slots, with task durations expressed in slots and latest completion times no greater than 48. The binary variable x_{jit} indicates whether task j starts on machine i at time t. The schedule solves

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & \sum_{j\in J}\sum_{i\in I}\sum_{t\in T} x_{jit}\left(\sum_{\substack{t'\in T\\ t\le t'<t+\zeta_j}} \phi_j c_{t'}\right) && \\
\text{s.t.} \quad & \sum_{i\in I}\sum_{t\in T} x_{jit} = 1 && \forall j \in J, \\
& x_{jit} = 0 && \forall j \in J,\ i \in I,\ t \in T \text{ with } t < \zeta_j^{(1)}, \\
& x_{jit} = 0 && \forall j \in J,\ i \in I,\ t \in T \text{ with } t + \zeta_j > \zeta_j^{(2)}, \\
& \sum_{j\in J}\sum_{\substack{t'\in T\\ t-\zeta_j<t'\le t}} x_{jit'} u_{jw} \le Q_{iw} && \forall i \in I,\ w \in W,\ t \in T, \\
& x_{jit} \in \{0,1\} && \forall j \in J,\ i \in I,\ t \in T.
\end{aligned}
$$

These constraints assign each task exactly once, respect its time window, and limit the resources used by simultaneously running tasks. The electricity prices come from the Irish Single Electricity Market Operator dataset (Ifrim et al., 2012), recorded at 30-minute intervals from 1 November 2011 to 31 December 2013. Each day forms one instance with 48 prices, each predicted from eight features by a shared linear model. Our three settings use three machines and 10, 15, or 20 tasks, denoted by loads $L = 1$, 2, and 3. The 789 daily instances are reshuffled for each seed and split into 550 training, 100 validation, and 139 test instances.

Knapsack. The problem selects items to maximize their total value subject to a known capacity:

$$
\max_{\boldsymbol{x}\in\{0,1\}^{48}} \ \boldsymbol{c}^\top \boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{w}^\top \boldsymbol{x} \le C,
$$

where the item weights $\boldsymbol{w}$ and capacity C are known, while the item values $\boldsymbol{c}$ are predicted. Each day of the electricity-price dataset forms one instance, with one item per half-hour slot. A fixed weight vector has entries in $\{3, 5, 7\}$ and total weight 240, and is shared across daily instances. Item values are constructed by multiplying each slot's electricity price by its weight and adding Gaussian noise $\xi \sim \mathcal{N}(0, 25)$. A shared linear model predicts each value from the eight features associated with its slot. We evaluate capacities $C \in \{60, 120, 180\}$ and use the same 550/100/139 training, validation, and test split procedure as for scheduling.

Diverse bipartite matching. The problem matches articles between two disjoint sets S_1 and S_2, each containing 50 articles from the CORA citation network (Sen et al., 2008). Let $c_{ij} \in \{0, 1\}$ indicate whether a citation exists between articles $i \in S_1$ and $j \in S_2$, and let $\phi_{ij} = 1$ if they belong

to the same field of study and 0 otherwise. For objective coefficients $\boldsymbol{c}$, the matching solves

$$
\begin{aligned}
\max_{\boldsymbol{x}} \quad & \sum_{i \in S_1} \sum_{j \in S_2} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j \in S_2} x_{ij} \leq 1 && \forall i \in S_1, \\
& \sum_{i \in S_1} x_{ij} \leq 1 && \forall j \in S_2, \\
& \sum_{i \in S_1} \sum_{j \in S_2} \phi_{ij} x_{ij} \geq \rho_1 \sum_{i \in S_1} \sum_{j \in S_2} x_{ij}, \\
& \sum_{i \in S_1} \sum_{j \in S_2} (1 - \phi_{ij}) x_{ij} \geq \rho_2 \sum_{i \in S_1} \sum_{j \in S_2} x_{ij}, \\
& x_{ij} \in \{0, 1\} && \forall i \in S_1,\ j \in S_2.
\end{aligned}
$$

At decision time, the predicted citation probabilities $\hat{c}_{ij}$ replace the unknown indicators c_{ij} in the objective. The diversity constraints require minimum fractions ρ_1 and ρ_2 of selected pairs to belong to the same and different fields, respectively. The network is divided into 27 disjoint instances of 100 articles. Each article has 1,433 bag-of-words features. Concatenating the features of a pair gives a 2,866-dimensional input to a predictor with one hidden layer and a sigmoid output. We evaluate $(\rho_1, \rho_2) \in \{(0.1, 0.1), (0.25, 0.25), (0.5, 0.5)\}$.

Metric. For minimization problems, regret is the realized objective minus the optimal objective, as in Eq. (3). For maximization problems, the order is reversed so that regret remains nonnegative. Following Mandi et al. (2024), we report mean relative regret for shortest path, scheduling, knapsack, and matching, dividing each instance's regret by its optimal objective value before averaging over the test set. Portfolio optimization uses absolute regret because the optimal return can be zero. We multiply portfolio regret by 10^3 for presentation in the tables. For PolyStepOR, optimal reference solutions are used only to compute these evaluation metrics.

Benchmark	Scenario	Cases	Problem	Predictor	Parameters	Train / Val / Test
Shortest path	Degree	$\{1, 2, 4, 6, 8\}$	LP	Linear	240	1000 / 250 / 10000
Portfolio	Degree	$\{1, 4, 8, 16\}$	QP	Linear	300	1000 / 250 / 10000
Knapsack	Capacity	$\{60, 120, 180\}$	0-1 ILP	Linear	9	550 / 100 / 139
Energy scheduling	Load	$\{1, 2, 3\}$	ILP	Linear	9	550 / 100 / 139
Matching	Instance type	$\{1, 2, 3\}$	ILP	MLP	573,601	22 / 5 / 5

Table 3: Problems, solvers, predictors, and splits of the classical benchmarks of Mandi et al. (2024).

B.1.2 Experimental Setups

PolyStepOR. PolyStepOR minimizes the minibatch mean of the realized decision cost in Eq. (7), reversing the objective sign for maximization problems. Training, early stopping, and checkpoint selection use no optimal reference decisions. We use one fixed configuration per problem without hyperparameter search (Table 17). The predictor is initialized randomly using the evaluation seed and optimized through a fixed per-layer search basis, which spans the full parameter space for the linear predictors and is capped for the matching network. Each step evaluates $2Kd$ candidates on a shared minibatch sampled at that step, and matching uses all training instances.

Validation realized cost is evaluated approximately 20 times over the step budget, and training stops after 10 consecutive checks without improvement. We restore the checkpoint with the lowest validation cost and evaluate it once on the test set. Before computing the softmax weights, we subtract each row's minimum probe cost and apply mean-absolute-cost normalization. The common shift leaves the weights unchanged, whereas the normalization sets their effective temperature. The exact reference-cost invariance in Proposition 1 therefore requires the normalization scale to remain unchanged when reference costs are subtracted. The geometric results also have a restricted scope. Shortest path, knapsack, scheduling, and matching admit finite sets of discrete decisions, whereas

portfolio has a continuous feasible set. Proposition 2 additionally requires the stated affine prediction map, so its exact formula applies to the linear predictors but not directly to the nonlinear matching network.

Baselines. We run MSE, SPO+ (Elmachtoub & Grigas, 2022), DBB (Pogancic et al., 2020), I-MLE (Niepert et al., 2021), FY (Blondel et al., 2020), and the solution-caching losses Listwise, Pairwise, Pairwise(diff), and MAP (Mulamba et al., 2021; Mandi et al., 2022) using the benchmark code and hyperparameters selected by Mandi et al. (2024). These methods use Adam, a plateau learning-rate schedule, and the original epoch budgets, returning the final-epoch model. The caching losses initialize their solution cache with all training optima and solve 5% of instances per step.

Computational cost. Table 4 reports the cost of an example PolyStepOR run. Each step evaluates $2Kd$ candidates on B shared training instances, requiring $2KdB$ solver calls before validation. These solves dominate training time and use the number of solver processes listed in the table. Portfolio solves run in a single-threaded solver within the training process.

Problem	d	$2Kd$	B	Solver calls per step	Mean stop / budget (steps)	Solver processes	Training (h)	Inference per instance (ms)
Shortest path	240	480	128	61 440	139.5 / 150	8	0.35 ± 0.05	2.36
Portfolio	300	600	128	76 800	92.5 / 100	1	2.04 ± 1.03	2.15
Knapsack	9	18	128	2304	98.0 / 100	1	0.06 ± 0.01	0.89
Energy	9	18	128	2304	93.8 / 100	32	1.46 ± 0.46	288.14
Matching	512	1024	17	17 408	80.7 / 100	32	1.25 ± 0.90	143.25

Table 4: Cost of one PolyStepOR run on the classical benchmarks, with training time as mean $\pm$ standard deviation over the settings and ten seeds. Inference is one forward pass plus one solve, measured on load 2 for energy.

B.1.3 Extended Results

Table 5 reports regret averaged across the settings of each problem. Tables 6 and 7 report individual settings, and Table 8 provides paired tests over the ten evaluation seeds. Reported uncertainties are standard deviations across seeds. The lowest-mean baseline in the test table is selected from the methods shown in the main-text figures, which exclude the solution-caching methods included in the extended result tables.

Table 5: Test regret on the classical benchmarks averaged on all settings of each problem over ten seeds. Bold underlined marks the lowest mean in each column and bold the second lowest.

Method	Shortest path	Portfolio ($\times 10^3$)	Knapsack	Energy	Matching
MSE	0.1470 ± 0.0029	1.0738 ± 0.0008	0.0870 ± 0.0089	0.0224 ± 0.0021	0.9352 ± 0.0060
SPO+	0.1303 ± 0.0002	0.2617 ± 0.0315	0.0829 ± 0.0183	0.0151 ± 0.0014	0.9177 ± 0.0130
DBB	0.1228 ± 0.0033	141.5336 ± 0.0659	**0.0616 ± 0.0016**	**0.0150 ± 0.0015**	0.9171 ± 0.0098
I-MLE	**0.1073 ± 0.0007**	131.6175 ± 1.2281	0.0620 ± 0.0024	**0.0149 ± 0.0015**	0.9165 ± 0.0174
FY	**0.1034 ± 0.0007**	107.8452 ± 1.0827	0.1467 ± 0.0152	0.0151 ± 0.0016	0.9304 ± 0.0139
Listwise	0.1128 ± 0.0006	9.2482 ± 7.2742	0.1493 ± 0.0274	0.0277 ± 0.0037	**0.9005 ± 0.0148**
Pairwise	0.1140 ± 0.0027	**0.1109 ± 0.0155**	0.2270 ± 0.0563	0.0276 ± 0.0031	**0.8979 ± 0.0156**
Pairwise(diff)	0.3562 ± 0.0049	0.5248 ± 1.1732	0.1074 ± 0.0294	0.0170 ± 0.0021	0.9033 ± 0.0104
MAP	0.1367 ± 0.0022	**0.0955 ± 0.0117**	0.1328 ± 0.0248	0.0152 ± 0.0013	0.9088 ± 0.0092
PolyStepOR	0.1121 ± 0.0016	0.1602 ± 0.0309	**0.0611 ± 0.0016**	**0.0150 ± 0.0016**	0.9223 ± 0.0107

Portfolio. Portfolio optimization has a continuous feasible set and a quadratic risk constraint. Averaged across polynomial degrees, PolyStepOR achieves the lowest regret among the methods in the main-text figures ($p = 0.002$ against SPO+). In the complete comparison, however, MAP and Pairwise achieve lower average regret. DBB, I-MLE, and FY produce substantially higher regret, consistent with the qualitative findings of Mandi et al. (2024). MSE performs best at degree 1, where a linear model can represent the conditional mean return.

Table 6: Test regret per setting on the synthetic classical benchmarks over ten seeds (portfolio in absolute regret $\times 10^3$). Bold underlined marks the lowest mean in each column and bold the second lowest.

Method	Shortest path, degree deg				
	deg = 1	deg = 2	deg = 4	deg = 6	deg = 8
MSE	**0.153 ± 0.000**	**0.101 ± 0.000**	0.085 ± 0.001	0.131 ± 0.002	0.266 ± 0.013
SPO+	**0.154 ± 0.000**	**0.101 ± 0.000**	0.076 ± 0.000	0.099 ± 0.001	0.222 ± 0.000
DBB	0.164 ± 0.003	0.126 ± 0.009	0.083 ± 0.002	0.092 ± 0.005	0.148 ± 0.006
I-MLE	0.159 ± 0.002	0.104 ± 0.002	**0.071 ± 0.000**	**0.087 ± 0.001**	**0.116 ± 0.002**
FY	0.157 ± 0.001	0.102 ± 0.000	**0.071 ± 0.001**	**0.081 ± 0.001**	**0.106 ± 0.002**
Listwise	0.169 ± 0.001	0.104 ± 0.000	0.077 ± 0.002	0.089 ± 0.000	0.126 ± 0.001
Pairwise	0.171 ± 0.014	0.105 ± 0.000	0.078 ± 0.001	0.091 ± 0.000	0.124 ± 0.001
Pairwise(diff)	0.163 ± 0.000	0.117 ± 0.000	0.215 ± 0.004	0.340 ± 0.008	0.946 ± 0.019
MAP	0.156 ± 0.000	0.103 ± 0.000	0.074 ± 0.001	0.119 ± 0.002	0.232 ± 0.010
PolyStepOR	0.157 ± 0.001	0.105 ± 0.002	0.075 ± 0.001	0.088 ± 0.002	0.134 ± 0.008

Method	Portfolio, degree deg, absolute regret ($\times 10^3$)			
	deg = 1	deg = 4	deg = 8	deg = 16
MSE	**0.116 ± 0.003**	0.038 ± 0.000	0.565 ± 0.000	3.576 ± 0.000
SPO+	0.261 ± 0.025	0.292 ± 0.091	0.196 ± 0.052	0.297 ± 0.045
DBB	126.527 ± 0.004	119.848 ± 0.003	134.911 ± 0.002	184.848 ± 0.265
I-MLE	114.044 ± 0.801	108.812 ± 0.443	122.268 ± 0.672	181.346 ± 4.286
FY	102.869 ± 1.013	101.373 ± 1.057	104.740 ± 1.524	122.399 ± 4.397
Listwise	18.994 ± 13.899	7.865 ± 12.390	9.753 ± 15.096	0.381 ± 0.172
Pairwise	0.371 ± 0.063	**0.000 ± 0.000**	**0.011 ± 0.017**	**0.062 ± 0.003**
Pairwise(diff)	0.406 ± 0.006	**0.000 ± 0.000**	**0.001 ± 0.000**	1.692 ± 4.687
MAP	**0.195 ± 0.043**	0.010 ± 0.025	0.022 ± 0.016	0.155 ± 0.022
PolyStepOR	0.423 ± 0.049	0.055 ± 0.048	0.056 ± 0.077	**0.107 ± 0.042**

Shortest path. The selected path changes only when predictions cross a decision boundary, so probes that preserve the path provide no cost difference for training. FY uses the optimal path as its target, while DBB and I-MLE receive the true cost vector as the derivative with respect to the decision. PolyStepOR receives only the scalar minibatch cost of each candidate. These differences describe the available training signals but do not by themselves establish the cause of the performance gap. PolyStepOR's regret exceeds that of the lowest-mean baseline by approximately 0.004 at degree 1 and 0.028 at degree 8. Training reaches the 150-step budget without early stopping in 24 of the 50 runs, including nine of the ten degree-8 runs. This indicates that the stopping criterion was not reached within the available budget, although it does not establish that longer training would close the gap. PolyStepOR nevertheless has lower mean regret than MSE at degrees 4, 6, and 8.

Matching. Averaged across instance types, the methods in the main-text figures remain within 2.0% of MSE, while Pairwise reduces regret by approximately 4.0% relative to MSE. Mean relative regret exceeds 0.87 in every reported setting, indicating that all methods remain far from the optimal matching objective. Interpretation is limited by the five test instances per type and the different training splits.

Knapsack and scheduling. Across all six settings, we detect no significant difference between PolyStepOR and the baseline with the lowest mean regret in the main-text comparison, with paired $p \geq 0.19$. PolyStepOR also has the lowest knapsack regret averaged across capacities among all ten methods. Both problems use linear predictors with nine parameters, allowing the full parameter space to be probed with 18 candidates per step. This makes each update substantially less expensive in solver calls than for the higher-dimensional predictors.

B.2 Extended in-prediction OR metrics and results

B.2.1 Problem Setups

We consider the integer optimization problems with predicted constraints introduced by Silvestri et al. (2026). Predictions determine the first-stage decision, which is evaluated under the realized

Table 7: Test regret per setting on the classical benchmarks with real data over ten seeds. Bold underlined marks the lowest mean in each column and bold the second lowest.

Method	Knapsack, capacity C		
	$C = 60$	$C = 120$	$C = 180$
MSE	0.149 ± 0.023	0.077 ± 0.005	0.035 ± 0.004
SPO+	0.142 ± 0.030	0.068 ± 0.030	0.038 ± 0.035
DBB	0.110 ± 0.006	**0.052 ± 0.003**	**0.023 ± 0.002**
I-MLE	**0.108 ± 0.005**	0.053 ± 0.004	0.025 ± 0.002
FY	0.289 ± 0.033	0.109 ± 0.016	0.043 ± 0.004
Listwise	0.275 ± 0.060	0.131 ± 0.044	0.042 ± 0.021
Pairwise	0.302 ± 0.066	0.217 ± 0.051	0.161 ± 0.142
Pairwise(diff)	0.167 ± 0.028	0.100 ± 0.052	0.056 ± 0.045
MAP	0.164 ± 0.013	0.104 ± 0.043	0.130 ± 0.061
PolyStepOR	**0.108 ± 0.004**	**0.052 ± 0.004**	**0.024 ± 0.002**

Method	Energy-cost aware scheduling, load L		
	$L = 1$	$L = 2$	$L = 3$
MSE	0.021 ± 0.003	0.030 ± 0.002	0.017 ± 0.002
SPO+	0.015 ± 0.002	0.020 ± 0.002	0.010 ± 0.001
DBB	**0.015 ± 0.002**	0.020 ± 0.003	0.010 ± 0.002
I-MLE	**0.015 ± 0.002**	0.020 ± 0.003	**0.010 ± 0.002**
FY	0.016 ± 0.002	**0.019 ± 0.002**	0.010 ± 0.002
Listwise	0.024 ± 0.004	0.028 ± 0.004	0.031 ± 0.007
Pairwise	0.032 ± 0.003	0.030 ± 0.005	0.020 ± 0.003
Pairwise(diff)	0.016 ± 0.002	0.022 ± 0.003	0.013 ± 0.003
MAP	0.015 ± 0.002	0.020 ± 0.002	0.010 ± 0.001
PolyStepOR	0.015 ± 0.002	**0.020 ± 0.002**	**0.010 ± 0.002**

Method	Diverse bipartite matching, instance type I		
	$I = 1$	$I = 2$	$I = 3$
MSE	0.934 ± 0.010	0.933 ± 0.007	0.939 ± 0.007
SPO+	0.924 ± 0.017	0.924 ± 0.025	0.905 ± 0.016
DBB	0.917 ± 0.022	0.914 ± 0.021	0.921 ± 0.018
I-MLE	0.919 ± 0.025	0.911 ± 0.024	0.920 ± 0.016
FY	0.939 ± 0.020	0.929 ± 0.025	0.924 ± 0.016
Listwise	**0.916 ± 0.030**	0.907 ± 0.017	**0.879 ± 0.018**
Pairwise	**0.908 ± 0.012**	**0.905 ± 0.024**	0.881 ± 0.019
Pairwise(diff)	0.916 ± 0.018	**0.903 ± 0.020**	0.891 ± 0.029
MAP	0.928 ± 0.021	0.919 ± 0.020	**0.880 ± 0.018**
PolyStepOR	0.924 ± 0.015	0.928 ± 0.014	0.915 ± 0.024

parameters and corrected through recourse when necessary. The published data-generation procedure uses the mapping of Elmachtoub & Grigas (2022), with degree 5, $p = 5$ features, and noise half-width 0.5, to generate Poisson distribution parameters. Realized problem parameters are then sampled from these distributions. Differences between this description and the released data are documented below. Each problem uses five datasets with three random splits each, giving 15 runs per setting. We retain the benchmark's notation ρ for the recourse penalty in this subsection.

Knapsack with stochastic weights (KP weight). Given a set of items I with known values v_i and capacity C, the first stage selects items using the predicted weights $\hat{\boldsymbol{w}}$,

$$\hat{\boldsymbol{x}} \in \arg\max_{\boldsymbol{x}\in\{0,1\}^{|I|}} \Big\{ \sum_{i\in I} v_i x_i : \sum_{i\in I} \hat{w}_i x_i \le C \Big\}.$$

After observing the true weights $\boldsymbol{w}$, recourse restores feasibility by choosing additions $\boldsymbol{u}^+$ and removals $\boldsymbol{u}^-$,

$$\max_{\boldsymbol{u}^+,\boldsymbol{u}^-\in\{0,1\}^{|I|}} \sum_{i\in I} \Big(\tfrac{1}{\rho} v_i u_i^+ - \rho v_i u_i^- \Big) \quad \text{s.t.} \quad \sum_{i\in I} w_i(\hat{x}_i + u_i^+ - u_i^-) \le C, \quad \hat{\boldsymbol{x}} \ge \boldsymbol{u}^-, \quad \hat{\boldsymbol{x}} + \boldsymbol{u}^+ \le \mathbf{1}. \tag{40}$$

The binary vectors $\boldsymbol{u}^+$ and $\boldsymbol{u}^-$ identify the items added and removed. Relative to the initial selection value, each addition contributes v_i/ρ and each removal incurs ρv_i, where $\rho > 1$. The total post-hoc

Problem	Setting	Lowest-mean baseline	Lower mean	p Wilcoxon	p sign-flip
Shortest path	$\deg = 1$	MSE	MSE	0.002	0.002
	$\deg = 2$	SPO+	SPO+	0.002	0.002
	$\deg = 4$	FY	FY	0.002	0.002
	$\deg = 6$	FY	FY	0.002	0.002
	$\deg = 8$	FY	FY	0.002	0.002
	pooled	FY	FY	0.002	0.002
Portfolio	$\deg = 1$	MSE	MSE	0.002	0.002
	$\deg = 4$	MSE	MSE	0.846	0.393
	$\deg = 8$	SPO+	PolyStepOR	0.006	0.006
	$\deg = 16$	SPO+	PolyStepOR	0.002	0.002
	pooled	SPO+	PolyStepOR	0.002	0.002
Knapsack	$C = 60$	I-MLE	PolyStepOR	0.322	0.518
	$C = 120$	DBB	DBB	0.557	0.465
	$C = 180$	DBB	DBB	0.695	0.514
	pooled	DBB	PolyStepOR	0.322	0.268
Energy	$L = 1$	DBB	DBB	0.322	0.303
	$L = 2$	FY	FY	0.193	0.068
	$L = 3$	I-MLE	I-MLE	0.432	0.488
	pooled	I-MLE	I-MLE	1.000	0.496
Matching	$I = 1$	DBB	DBB	0.625	0.471
	$I = 2$	I-MLE	I-MLE	0.131	0.059
	$I = 3$	SPO+	SPO+	0.275	0.328
	pooled	I-MLE	I-MLE	0.557	0.352

Table 8: Paired two-sided tests of PolyStepOR against the lowest-mean baseline of the main-text figures, over the ten paired seeds of each setting. "Lower mean" indicates the method with the lower mean of the pair.

value is the initial selection value plus the optimized recourse objective. The problem has 50 items and $\rho \in \{5, 10, 20\}$.

Knapsack with stochastic capacity (KP capacity). This variant predicts the capacity while keeping item weights and values known. The first-stage constraint uses $\hat{C}$, and recourse uses the realized capacity C in Eq. (40) (Table 8 of Silvestri et al. (2026)). The problem has 50 items and $\rho \in \{5, 10, 20\}$.

Weighted set multi-cover (WSMC). Given items I and covers J, let $a_{ij} = 1$ if cover j covers item i, let κ_j be the cost of cover j, and let d_i be the coverage requirement of item i. The first stage solves

$$\hat{\boldsymbol{x}} \in \arg\min_{\boldsymbol{x} \in \mathbb{Z}^{|J|}_{\geq 0}} \Big\{ \sum_{j \in J} \kappa_j x_j : \sum_{j \in J} a_{ij} x_j \geq \hat{d}_i \ \ \forall i \in I \Big\},$$

where x_j counts how often cover j is selected and the requirements $\hat{\boldsymbol{d}}$ are predicted. After the true requirements $\boldsymbol{d}$ are observed, each unit of unmet demand for item i incurs a recourse charge equal to ρ times the largest cost among covers containing that item. The resulting post-hoc cost is

$$\sum_{j \in J} \kappa_j \hat{x}_j + \rho \sum_{i \in I} \Big(\max_{j:\, a_{ij}=1} \kappa_j \Big) \max\Big(0,\ d_i - \sum_{j \in J} a_{ij} \hat{x}_j \Big). \tag{41}$$

The availability matrices follow the generator of Silvestri et al. (2026), and the cover costs are drawn uniformly from $[1, 100]$. We use 10 items and 50 covers with $\rho \in \{1, 5, 10\}$, as in Table 3 of Silvestri et al. (2026), and 5 items and 25 covers, as in their Table 9.

Metrics. The primary metric of Silvestri et al. (2026) is the relative post-hoc regret. It divides the post-hoc regret of Eq. (5) by the magnitude of the optimal cost under the true parameters and averages over the test instances. We also report the fraction of test instances whose first-stage decision

is infeasible under the realized parameters, before recourse, and the mean squared prediction error (MSE). For knapsack, regret uses the maximization convention, subtracting the achieved post-hoc value from the optimal value.

B.2.2 Experimental Setups

Data and splits. We use the authors' code and data generators. The released code splits each dataset into 720 training, 80 validation, and 50 test instances, while the paper states an 80/10/10 split. The released knapsack data use multiplicative noise 0.1 and additive noise 0.03 instead of the stated half-width 0.5. On these data, our baseline reruns, including CombOptNet where available, reproduce the knapsack results of Tables 2 and 8 of Silvestri et al. (2026) within one standard deviation. We generate the WSMC data with the authors' generator at the stated half-width 0.5. Our reruns then reproduce their Table 9 for 5×25, whereas the MSE and SFGE values of their Table 3 for 10×50 are about twice ours.

PolyStepOR. We train the benchmark's linear predictors using the original data loaders and optimization solvers. The training loss is the post-hoc cost in Eq. (8), with the optimal reference term omitted. Validation uses the same cost for early stopping and checkpoint selection, so neither training nor selection requires optimal reference decisions.

All settings use one fixed configuration without hyperparameter search. We use a fixed per-layer search basis without a dimension cap. The predictor's output scaling remains fixed. When d is not a multiple of q, the last of the $\lceil d/q \rceil$ blocks is filled with dummy coordinates that map to no parameter, so probes along them return the current cost. Each step evaluates candidates on a shared minibatch, following the fixed order of the original data loader. Training stops after 10 consecutive epochs without improvement in validation post-hoc cost. We restore the best validation checkpoint and evaluate it once on the test set. Candidate decisions are computed using the authors' Gurobi models, with a 30-second time limit for knapsack. The fixed batch order differs from the independent sampling assumed in the stationarity analysis.

Baselines. The baseline denoted MSE follows Silvestri et al. (2026). It fits a Gaussian predictive distribution by minimizing negative log-likelihood, with a standard deviation that does not depend on the features. Thus, its training objective is likelihood-based despite its benchmark label. CombOptNet (Paulus et al., 2021) uses the authors' implementation. Its released WSMC loader provides no validation split, so we add one following the knapsack loader's procedure. We do not evaluate CombOptNet on KP capacity, for which the original paper reports no result. SFGE uses the authors' settings, including Gaussian predictions with a learned standard deviation, standardized batch values, and Adam with learning rate 0.005. Early stopping is based on validation regret. In these implementations, CombOptNet trains on optimal reference decisions and SFGE evaluates an objective containing optimal reference costs. MSE and PolyStepOR use neither. This describes the reference usage of the evaluated implementations, rather than an inherent requirement of score-function estimation.

Computational cost. KP capacity uses five CPU cores with four parallel solver processes. The remaining problems use eight cores and eight solver processes. Memory use is approximately 4 GB per run. Across penalty settings, the mean stopping epoch ranges from 18.6 to 19.7 for KP capacity, 28.2 to 29.3 for KP weight, 17.7 to 23.7 for WSMC 10×50, and 19.6 to 25.4 for WSMC 5×25. Estimated training time per run averages 52 minutes for KP capacity (range 17–124 minutes), 6.7 hours for KP weight (2.4–12.5 hours), 2.8 hours for WSMC 10×50 (0.5–6.8 hours), and 62 minutes for WSMC 5×25 (12–219 minutes). These estimates combine time per epoch with the number of training epochs because some runs resumed across time-limited jobs. Inference requires one predictor evaluation and one first-stage solve. Gurobi solve times per test instance are 18.4 ms for KP capacity, 3.9 ms for KP weight, 13.9 ms for WSMC 10×50, and 7.8 ms for WSMC 5×25. The reported timings including model construction and evaluation range from 13 to 44 ms.

B.2.3 Extended Results

Tables 9 to 12 report regret, prediction error, and infeasibility for each problem and penalty setting. Table 13 reports paired comparisons with SFGE and MSE across the 15 runs.

Table 9: Knapsack with stochastic capacity (KP capacity), Table 8 of Silvestri et al. (2026) over 15 runs. Bold underlined marks the lowest value in each block and column, and bold the second lowest.

Method	Rel. post-hoc regret	MSE	Infeas. ratio
50 items, $\rho = 5$			
MSE	0.5556 ± 0.3529	**10.81**	**0.667 ± 0.257**
SFGE	**0.3577 ± 0.2152**	**16.63**	**0.749 ± 0.295**
PolyStepOR	**0.2975 ± 0.1666**	21.87	0.752 ± 0.289
50 items, $\rho = 10$			
MSE	1.2126 ± 0.7797	**10.81**	**0.667 ± 0.257**
SFGE	**0.5081 ± 0.2784**	**23.58**	0.907 ± 0.063
PolyStepOR	**0.3802 ± 0.1958**	32.04	**0.847 ± 0.212**
50 items, $\rho = 20$			
MSE	2.5205 ± 1.6313	**10.81**	**0.667 ± 0.257**
SFGE	**0.8216 ± 0.5031**	**32.66**	0.955 ± 0.036
PolyStepOR	**0.4772 ± 0.2415**	37.68	**0.877 ± 0.168**

Table 10: Knapsack with stochastic weights (KP weight), Table 2 of Silvestri et al. (2026) over 15 runs. Bold underlined marks the lowest value in each block and column, and bold the second lowest.

Method	Rel. post-hoc regret	MSE	Infeas. ratio
50 items, $\rho = 5$			
MSE	0.1676 ± 0.0357	7.88×10^4	**0.929 ± 0.031**
CombOptNet	0.1901 ± 0.0289	5.09×10^7	**0.907 ± 0.032**
SFGE	**0.1297 ± 0.0114**	3.48×10^5	0.987 ± 0.017
PolyStepOR	**0.1501 ± 0.0210**	1.58×10^6	0.964 ± 0.021
50 items, $\rho = 10$			
MSE	0.3190 ± 0.0814	7.88×10^4	**0.929 ± 0.031**
CombOptNet	0.4015 ± 0.0659	5.09×10^7	**0.907 ± 0.032**
SFGE	**0.1760 ± 0.0221**	3.66×10^5	0.989 ± 0.012
PolyStepOR	**0.2293 ± 0.0372**	2.01×10^6	0.980 ± 0.025
50 items, $\rho = 20$			
MSE	0.6146 ± 0.1745	7.88×10^4	**0.929 ± 0.031**
CombOptNet	0.8233 ± 0.1401	5.09×10^7	**0.907 ± 0.032**
SFGE	**0.2205 ± 0.0276**	3.73×10^5	0.993 ± 0.009
PolyStepOR	**0.2884 ± 0.0558**	2.01×10^6	0.984 ± 0.024

KP capacity. PolyStepOR achieves the lowest mean post-hoc regret at every penalty level, with a significant paired difference from SFGE in each setting. Increasing the predicted capacity enlarges the feasible set and can lead to selections that exceed the realized capacity. Recourse then removes items at ρ times their value, whereas additions contribute only $1/\rho$ of their value. Larger penalties therefore increase the cost of removals relative to additions. At $\rho = 10$ and 20, PolyStepOR produces fewer infeasible first-stage decisions than SFGE, with ratios of 0.85 and 0.88 compared with 0.91 and 0.96. Its regret also grows less across the penalty settings, from 0.30 to 0.48, compared with 0.36 to 0.82 for SFGE. MSE has the lowest prediction error and infeasibility ratio but higher post-hoc regret, illustrating that these metrics measure different aspects of decision quality.

KP weight. SFGE achieves the lowest mean regret at every penalty level, while PolyStepOR improves over MSE and CombOptNet. First-stage infeasibility is high for all methods, ranging from approximately 0.90 to 0.99, so recourse plays a substantial role in evaluation. Unlike KP capacity,

Table 11: Weighted set multi-cover 10×50, Table 3 of Silvestri et al. (2026) over 15 runs. Bold underlined marks the lowest value in each block and column, and bold the second lowest.

Method	Rel. post-hoc regret	MSE	Infeas. ratio
	10×50, $\rho = 1$		
MSE	1.2960 ± 0.5676	$\underline{1.04 \times 10^5}$	0.912 ± 0.048
CombOptNet	5.2044 ± 2.1887	1.89×10^7	1.000 ± 0.000
SFGE	**1.1373 ± 0.4368**	3.33×10^5	**0.809 ± 0.125**
PolyStepOR	**1.1205 ± 0.5002**	4.87×10^5	**0.839 ± 0.091**
	10×50, $\rho = 5$		
MSE	5.7115 ± 2.7809	$\underline{1.04 \times 10^5}$	0.912 ± 0.048
CombOptNet	110.7887 ± 30.2424	1.89×10^7	1.000 ± 0.000
SFGE	**2.5460 ± 0.9283**	6.43×10^5	**0.340 ± 0.113**
PolyStepOR	**2.5253 ± 1.0119**	8.88×10^5	**0.452 ± 0.121**
	10×50, $\rho = 10$		
MSE	11.2307 ± 5.5511	$\underline{1.04 \times 10^5}$	0.912 ± 0.048
CombOptNet	440.7395 ± 118.7445	1.89×10^7	1.000 ± 0.000
SFGE	**3.3108 ± 1.1913**	7.80×10^5	**0.235 ± 0.088**
PolyStepOR	**3.3701 ± 1.5660**	1.08×10^6	**0.345 ± 0.091**

Table 12: Weighted set multi-cover 5×25, Table 9 of Silvestri et al. (2026) over 15 runs. Bold underlined marks the lowest value in each block and column, and bold the second lowest.

Method	Rel. post-hoc regret	MSE	Infeas. ratio
	5×25, $\rho = 1$		
MSE	1.6856 ± 1.1119	$\underline{8.07 \times 10^4}$	0.728 ± 0.163
CombOptNet	6.0568 ± 4.1781	8.66×10^6	1.000 ± 0.000
SFGE	**1.2319 ± 0.5118**	2.98×10^5	**0.533 ± 0.280**
PolyStepOR	**1.0940 ± 0.3827**	7.20×10^5	**0.568 ± 0.248**
	5×25, $\rho = 5$		
MSE	7.4364 ± 5.6404	$\underline{8.07 \times 10^4}$	0.728 ± 0.163
CombOptNet	163.2305 ± 106.4945	8.66×10^6	1.000 ± 0.000
SFGE	**2.3798 ± 0.7306**	5.28×10^5	**0.181 ± 0.102**
PolyStepOR	**2.3524 ± 0.8307**	7.38×10^5	**0.279 ± 0.142**
	5×25, $\rho = 10$		
MSE	14.6249 ± 11.3160	$\underline{8.07 \times 10^4}$	0.728 ± 0.163
CombOptNet	654.3986 ± 426.2526	8.66×10^6	1.000 ± 0.000
SFGE	**2.8663 ± 0.8869**	6.33×10^5	**0.120 ± 0.074**
PolyStepOR	**3.2120 ± 1.2114**	7.18×10^5	**0.184 ± 0.097**

which predicts one scalar, this problem predicts 50 weights jointly. PolyStepOR searches 270 of the predictor's 300 parameters using one probe radius, compared with six parameters for KP capacity. Its prediction MSE is approximately 4.5 to 5.5 times that of SFGE. These differences characterize the more demanding setting but do not isolate the cause of the regret gap.

WSMC. On WSMC 10×50, we detect no significant difference in post-hoc regret between PolyStepOR and SFGE. Both outperform MSE by a larger margin as the penalty increases. Their first-stage infeasibility also decreases between $\rho = 1$ and $\rho = 10$, from 0.84 to 0.35 for PolyStepOR and from 0.81 to 0.24 for SFGE, while MSE remains at 0.91. This is consistent with stronger penalties encouraging decisions that satisfy more of the realized coverage requirements.

Table 13: Paired two-sided tests over the 15 paired runs of each cell (5 datasets $\times$ 3 splits) on the relative post-hoc regret. "Lower mean" names the method with the lower mean of the pair, and "Runs" counts the runs in which PolyStepOR has the lower regret.

		Against SFGE				Against MSE	
Problem	ρ	Lower mean	p Wilcoxon	p sign-flip	Runs	Lower mean	p Wilcoxon
KP (capacity)	5	PolyStepOR	0.002	< 0.001	12/15	PolyStepOR	0.002
	10	PolyStepOR	< 0.001	< 0.001	14/15	PolyStepOR	< 0.001
	20	PolyStepOR	< 0.001	< 0.001	15/15	PolyStepOR	< 0.001
KP (weight)	5	SFGE	0.004	0.005	2/15	PolyStepOR	0.022
	10	SFGE	< 0.001	< 0.001	0/15	PolyStepOR	< 0.001
	20	SFGE	< 0.001	< 0.001	1/15	PolyStepOR	< 0.001
WSMC 10×50	1	PolyStepOR	0.252	0.520	9/15	PolyStepOR	0.001
	5	PolyStepOR	0.679	0.711	10/15	PolyStepOR	< 0.001
	10	SFGE	0.561	0.880	9/15	PolyStepOR	< 0.001
WSMC 5×25	1	PolyStepOR	0.151	0.059	8/15	PolyStepOR	0.041
	5	PolyStepOR	0.762	0.863	8/15	PolyStepOR	< 0.001
	10	SFGE	0.048	0.162	3/15	PolyStepOR	< 0.001

The smaller WSMC 5×25 problem shows a similar pattern. At $\rho = 10$, SFGE has lower mean regret, with a Wilcoxon p-value of 0.048 but a sign-flip p-value of 0.162. Evidence for a difference in this setting therefore depends on the test used.

B.3 Extended Details on Network Districting

B.3.1 Problem Setup

We follow the network districting problem of Ahmed et al. (2024). A city is represented as a graph whose vertices are basic units (BUs) with stochastic customer demand, together with a depot. The N BUs are partitioned into $k = \lfloor N/t \rfloor$ connected districts, each constrained to contain within $\pm 20\%$ of the target size t. For a district d, the routing cost is the expected length of the shortest depot tour serving its realized demand. Using the authors' evaluator with $|\Omega| = 100$ demand scenarios, the cost of a districting λ is

$$\mathrm{Cost}(\lambda) = \sum_{d \in \lambda} \frac{1}{|\Omega|} \sum_{\omega \in \Omega} \mathrm{TSP}(d, \boldsymbol{\xi}^{\omega}). \tag{42}$$

DistrictNet does not optimize this routing objective directly. Instead, a graph neural network predicts edge weights for a capacitated minimum spanning tree (CMST) surrogate, whose solution induces the district partition. At test time, the surrogate is solved using the authors' iterated local search, with a time limit of 20 minutes for instances below 400 BUs and 60 minutes otherwise. PolyStepOR retains the same predictor, surrogate problem, solver, and downstream evaluator, but trains the predictor directly from the districting costs in Eq. (42). :chatgpt-content-referenceindex="0"

Data. Training uses the 100 cities of Ahmed et al. (2024), each containing $N = 30$ BUs with target size $t = 3$ and generated from 27 English cities outside the test set. DistrictNet is trained from optimal districtings of these instances, obtained by enumerating feasible districts and solving the resulting optimization problem. Generating such labels becomes rapidly expensive: Ahmed et al. (2024) report approximately 400 CPU-core days for one instance with 60 BUs and 10 districts. The test set contains the 35 instances of their Table 1, corresponding to seven cities (Bristol, Leeds, London, Lyon, Manchester, Marseille, and Paris) with $N = 120$ and $t \in \{3, 6, 12, 20, 30\}$. We additionally evaluate on the Ile-de-France instance with 2000 BUs and $t = 20$. Both test settings are therefore substantially larger than the 30-BU training instances. :chatgpt-content-referenceindex="1"

Evaluation. Each returned districting is evaluated by Eq. (42) using the same 100 demand scenarios for a given problem. For method m, we report its mean relative cost to PolyStepOR over the 35

test problems,

$$\frac{100}{35}\sum_{p=1}^{35}\frac{\mathrm{Cost}_m(p)-\mathrm{Cost}_{\mathrm{PS}}(p)}{\mathrm{Cost}_{\mathrm{PS}}(p)}, \tag{43}$$

together with a paired two-sided Wilcoxon test over the 35 instances. Ahmed et al. (2024) use the analogous relative-cost metric with DistrictNet as the reference and a one-sided test.

B.3.2 Experimental Setup

PolyStepOR. PolyStepOR trains the authors' graph network, containing $25{,}857$ parameters, on the same 100 training cities. For each candidate predictor, the predicted edge weights define a CMST problem, and the resulting districting is evaluated using the authors' precomputed district costs. The training objective is the mean districting cost over all 100 cities. Neither the optimal training districtings nor pretrained DistrictNet parameters are used, and each run starts from a random initialization determined by its seed. All remaining components, including the network architecture, data, demand scenarios, optimization model, and downstream evaluator, are inherited from Ahmed et al. (2024).

We report both an untuned and a tuned configuration, with their hyperparameters listed in Table 17. The tuned configuration is selected using the second set of 100 small cities provided by Ahmed et al. (2024), which serves as an in-distribution validation set. Within each run, the returned predictor is the probed candidate with the lowest training cost. For one untuned run, a single training solve did not terminate; this run therefore uses a finite solve limit, retaining the incumbent when available and assigning a fixed failure cost otherwise.

Baselines. DistrictNet is retrained with the authors' code at each of the ten PolyStepOR seeds and evaluated on the same demand scenarios. The results for BD, FIG, PredGNN, and AvgTSP are taken directly from Ahmed et al. (2024). For Ile-de-France, all baseline values, including DistrictNet, are the published results of Ahmed et al. (2024).

Compute. For the untuned configuration, each PolyStepOR training step evaluates 1024 candidate predictors. A step takes 15.9 minutes on 64 CPU cores on average (range: 10.7–29.3 minutes) and 38.3 minutes on 16 cores, giving approximately 8 hours for a 30-step run on 64 cores. Seeds 17, 42, and 71 were trained on 16 cores and took 16.8–20.9 hours; the remaining seeds took 8.3–10.9 hours. Candidate evaluations are parallelized across worker processes, each holding a copy of the training data, resulting in a peak memory usage of approximately 55 GB on 64 cores.

The tuned configuration evaluates 4096 candidates per step and requires 27–31 minutes per step on 128 cores, corresponding to 13.5–14.5 hours per run. Retraining DistrictNet takes 32.5 minutes on 2 cores. At inference time, both methods use the authors' iterated local search with the same time limits. Runtime is approximately 21.4 minutes per $N = 120$ instance (20.9–23.3 minutes) and 64.2 minutes on Ile-de-France. Evaluating the returned districtings over the demand scenarios requires an additional 10.7 minutes per $N = 120$ instance on average, and 7–14 minutes per method on Ile-de-France.

B.3.3 Extended Results

Table 14 reports the tuned PolyStepOR configuration for each training seed together with DistrictNet retrained at the same seed. Table 15 aggregates the tuned and untuned configurations over their respective seeds. On the 35 benchmark problems, the tuned configuration and DistrictNet have similar costs, with no significant difference under the paired Wilcoxon test ($p = 0.14$). The untuned configuration performs worse, with DistrictNet reducing cost by 2.0% on average ($p = 2 \times 10^{-10}$). On Ile-de-France, the tuned PolyStepOR runs achieve a mean cost of 2197.0, with the best run reaching 2156.3, compared with the published DistrictNet cost of 2205.7. Figure 4 visualizes the districtings produced by both methods on the examples considered by Ahmed et al. (2024). :chatgpt-content-referenceindex="2"

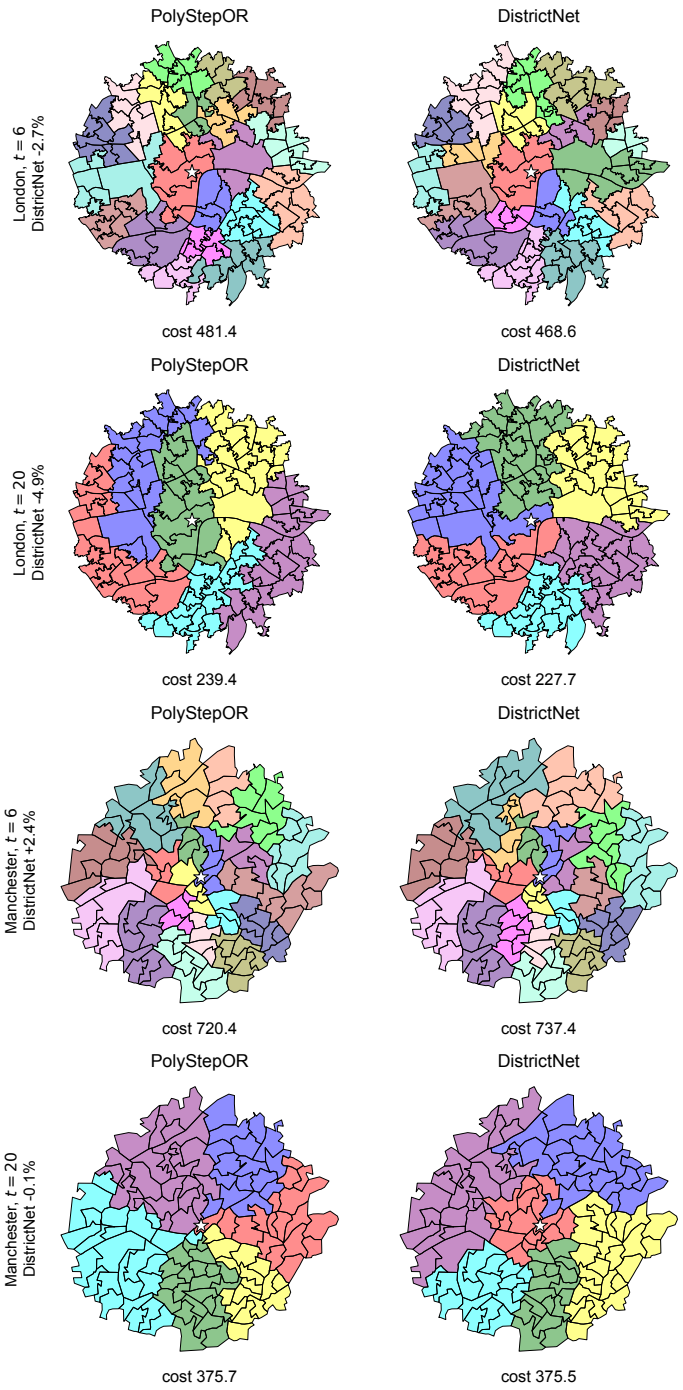

Figure 4: Districtings produced by PolyStepOR and DistrictNet at seed 1234 on the examples visualized by Ahmed et al. (2024): London and Manchester at $t = 6$ and $t = 20$. The cost is shown below each map, and the row label reports the relative cost of DistrictNet with respect to PolyStepOR.

Table 14: Districting results per training seed for tuned PolyStepOR and DistrictNet retrained with the authors' code at the same seed. Results are reported over the 35 benchmark problems together with the PolyStepOR cost on Ile-de-France. Relative cost is measured for DistrictNet with respect to PolyStepOR, so negative values indicate lower DistrictNet cost. The p-value is from a paired two-sided Wilcoxon test over the 35 problems.

	35 problems					Ile-de-France
Seed	PolyStepOR	DistrictNet	Rel. (%)	p	PolyStepOR cheaper	PolyStepOR
3	551.7	551.4	-0.71	0.53	15/35	2156.3
5	552.2	562.0	+0.79	6×10^{-3}	22/35	2268.3
7	555.6	557.8	-0.16	0.64	19/35	2190.7
11	554.5	556.0	-0.97	0.40	13/35	2219.5
13	554.4	557.2	+0.54	0.08	25/35	2183.0
17	551.5	556.1	+0.50	0.34	20/35	2200.6
19	554.4	557.4	-0.89	0.70	18/35	2191.6
42	557.3	561.0	-0.64	0.39	16/35	2203.9
71	552.4	558.6	+0.90	0.14	20/35	2162.7
1234	556.3	559.5	+0.55	0.20	21/35	2193.7
Mean over seeds	554.0	557.7	−0.05			2197.0

Table 15: Performance of tuned and untuned PolyStepOR configurations. Results on the 35 benchmark problems are compared with DistrictNet retrained at the same seeds. Ile-de-France reports the mean and best PolyStepOR cost across seeds.

		35 problems				Ile-de-France	
Configuration	Seeds	PolyStepOR	Rel. DistrictNet (%)	p	PolyStepOR cheaper	Mean	Best
Tuned	10	554.0	-0.05	0.14	18/35	2197.0	2156.3
Untuned	10	565.3	-1.97	2×10^{-10}	1/35	2220.5	2186.6

B.4 Extended Details on Brass Alloy Allocation

B.4.1 Problem Setups

We consider the brass alloy production problem of Mandi et al. (2026), adapted from Hu et al. (2023a). A factory purchases ore from $N = 10$ suppliers to meet copper (Cu) and zinc (Zn) requirements at minimum cost. Supplier n charges q_n per unit of ore, which contains a_{nm} units of metal $m \in \{\mathrm{Cu}, \mathrm{Zn}\}$. The metal contents are unknown at decision time and must be predicted. Given these contents, the purchase problem is modeled as:

$$\min_{\boldsymbol{x}} \sum_{n=1}^{N} q_n x_n \quad \text{s.t.} \quad \sum_{n=1}^{N} a_{nm} x_n \geq b_m \quad \forall m \in \{\mathrm{Cu}, \mathrm{Zn}\}, \qquad \boldsymbol{x} \geq \mathbf{0},$$

where x_n is the quantity purchased from supplier n. The requirements are $b_{\mathrm{Cu}} = 627.54$ and $b_{\mathrm{Zn}} = 369.72$. At decision time, predicted contents replace a_{nm} in the constraints. The dataset contains 500 instances, split into 350 training, 50 validation, and 100 test instances. Each metal content is predicted from 4,096 features using a fully connected network with one hidden layer of 512 units. Although Mandi et al. (2026) describe a linear program, their implementation restricts the purchase quantities to integers. We follow that implementation and add $\boldsymbol{x} \in \mathbb{Z}^N_{\geq 0}$ to the formulation above.

Metrics. Let $\mathcal{I}$ denote the test instances whose predicted purchases satisfy the requirements under the true metal contents, and let r_i be the optimal purchase cost for instance i. Following Eq. 23 of Mandi et al. (2026), we report

$$\mathrm{Infeas} = 1 - \frac{|\mathcal{I}|}{N_{\mathrm{test}}}, \qquad \mathrm{CRegret} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \frac{\boldsymbol{q}_i^\top \hat{\boldsymbol{x}}_i - r_i}{r_i}. \tag{44}$$

Infeasibility is measured before correction. Conditional regret is computed only on each method's feasible purchases and is undefined when $\mathcal{I}$ is empty. Consequently, a low conditional regret can reflect performance on a small or easier subset of instances rather than good performance across the full test set.

B.4.2 EXPERIMENTAL SETUPS

PolyStepOR. We train the authors' network from random initialization using their training loop. Feasible purchases are evaluated by their purchase cost. For an infeasible purchase, the authors' correction scales the quantities until both metal requirements are satisfied. Let $\boldsymbol{a}_{i,m} \in \mathbb{R}^N$ collect the true contents of metal m across suppliers in instance i. The correction factor is

$$\tau_i = \max\left\{1, \max_{m\in\{\mathrm{Cu},\mathrm{Zn}\}} \frac{b_m}{\boldsymbol{a}_{i,m}^\top \hat{\boldsymbol{x}}_i}\right\},$$

provided the denominators are positive. As in the authors' evaluator, the corrected purchase $\tau_i\hat{\boldsymbol{x}}_i$ is not rounded, so it can be fractional. The denominators are positive for every instance, since every true content in the data is at least 15 and a returned purchase is nonzero because the requirements are positive. Writing $\boldsymbol{a}_i$ for the collection of true metal contents, the training loss is:

$$\ell(\hat{\boldsymbol{x}}_i, \boldsymbol{a}_i) = \boldsymbol{q}_i^\top(\tau_i\hat{\boldsymbol{x}}_i) + \mu_i(\tau_i - 1)\boldsymbol{q}_i^\top\hat{\boldsymbol{x}}_i, \tag{45}$$

where μ_i is the instance-specific penalty coefficient supplied with the data. The first term is the corrected purchase cost, and the second penalizes the additional purchase. For an initially feasible decision, $\tau_i = 1$ and the loss reduces to its original cost. If the solver returns no purchase, the assigned loss is $100\,\mu_i\|\boldsymbol{q}_i\|_1$. These evaluations require no optimal reference purchase. Unlike the other experiments, the configuration is chosen by validation cost among the points of a 32-point scrambled Sobol design trained on separate tuning seeds, and then retrained on the evaluation seeds (Table 17). The search subspace is rebuilt at each step from previous displacements and random directions, rather than using the fixed basis of Section 4.2, and the number of probe radii drops from 2 to 1 after three consecutive steps with decreasing cost. These practical variants differ from the fixed-setting algorithm considered in the stationarity analysis. Training follows the authors' setup with shuffling between epochs, and we evaluate the final-epoch model, solving every candidate with the authors' Gurobi model in a single solver process.

Baselines. For infeasibility and conditional regret, we use the results reported by Mandi et al. (2026) for MSE, CombOptNet, SFL (Nandwani et al., 2022), 2sPtO (Hu et al., 2023a), and ODECE with $\alpha \in \{0.2, \ldots, 0.8\}$. We additionally run SFGE using the authors' estimator and published post-hoc regret objective. Three of its five seeds produce no feasible test purchase, so conditional regret is averaged over the remaining two seeds. For the full-test post-hoc regret comparison, we separately rerun MSE and ODECE using the authors' code and the same five evaluation seeds as PolyStepOR. These reruns are distinct from the published baseline values in Table 16.

Feasibility. Following Mandi et al. (2026), the purchase $\hat{\boldsymbol{x}}_i$ predicted for test instance i is feasible when it meets both requirements under the true contents,

$$\hat{\boldsymbol{x}}_i \text{ is feasible} \iff \boldsymbol{a}_{i,m}^\top\hat{\boldsymbol{x}}_i \geq b_m \quad \forall m \in \{\mathrm{Cu}, \mathrm{Zn}\}. \tag{46}$$

The set $\mathcal{I}$ in Eq. (44) collects the test instances that satisfy Eq. (46), and the check is made on the purchase before any correction.

Computational cost. A PolyStepOR training run takes 2.9 hours on average (2.8 to 3.3 over the five seeds), using one solver process. The solver calls take about 27% of the time. The rest is dominated by the forward passes of the network over all probe candidates, which run on the CPU alongside the solver. Evaluating the candidates as one batch on a GPU, as the PolyStep library supports, is much cheaper. In our GPU-batched runs of the same network, the forward passes took about 11% of the training time with 24 times more candidates per step.

B.4.3 EXTENDED RESULTS

Table 16 reports the values shown in Figure 3. The PolyStepOR row corresponds to the moving mode. Its first-stage infeasibility is 0.555 ± 0.013, compared with 0.532 for MSE, while its conditional regret is 0.154 ± 0.005, compared with 0.169. Thus, PolyStepOR has slightly lower reported conditional regret but a higher infeasibility rate. The standard deviations describe variability across its five evaluation seeds. ODECE varies the emphasis on feasibility through α. At $\alpha = 0.4$, its

infeasibility rate of 0.571 is close to PolyStepOR's, with conditional regret of 0.195. Increasing α reduces infeasibility but raises conditional regret, reaching 0.174 and 0.486, respectively, at $\alpha = 0.8$. PolyStepOR uses the dataset's prescribed recourse penalties rather than an analogous sweep over α, so its reported configuration provides one operating point.

SFL and CombOptNet are infeasible on 99.1% and 89.6% of test instances, respectively, leaving small subsets on which to compute conditional regret. The infeasibility rate of 2sPtO is lower at 40.0%, but its conditional regret is 8.539. SFGE has an infeasibility rate of 73.9% and conditional regret of 0.268, with the latter defined for only two of its five evaluation seeds. Because conditional regret uses different feasible subsets, these values alone do not establish an overall ranking of decision quality. We therefore also evaluate post-hoc regret across the full test set, including correction and penalty costs. PolyStepOR achieves 0.231 ± 0.005, compared with 0.243 ± 0.007 for our MSE reruns. ODECE ranges from 0.230 at $\alpha = 0.3$ to 0.543 at $\alpha = 0.8$. This comparison evaluates all methods on the same instances and complements the separate assessment of feasibility before correction.

Table 16: Brass alloy production, with infeasibility before repair and regret over each method's feasible test purchases; lower is better. The baselines except SFGE are the values reported by Mandi et al. (2026), and PolyStepOR is mean $\pm$ standard deviation over five seeds.

Method	Infeasibility	Conditional regret
MSE	0.532	0.169
CombOptNet	0.896	0.181
SFL	0.991	0.019
2sPtO	0.400	8.539
ODECE ($\alpha = 0.2$)	0.790	0.081
ODECE ($\alpha = 0.3$)	0.729	0.123
ODECE ($\alpha = 0.4$)	0.571	0.195
ODECE ($\alpha = 0.5$)	0.471	0.270
ODECE ($\alpha = 0.6$)	0.324	0.365
ODECE ($\alpha = 0.7$)	0.276	0.406
ODECE ($\alpha = 0.8$)	0.174	0.486
SFGE	0.739	0.268
PolyStepOR	0.555 ± 0.013	0.154 ± 0.005

B.5 Reproducibility Configurations

We provide the hyperparameter configurations, as well as the seeds used in each experiment to better guide the reproducibility of our algorithm, which are presented in Table 17. The following seeds are

Table 17: PolyStepOR settings behind every reported number.

Problem	Schedule	ε		s		r		d	q	K	Momentum	B	Budget (steps)
		start	end	start	end	start	end						
Classical OR													
Shortest path	cosine	10	0.1	5	1	10	2	240	8	1	0	128	150
Portfolio	cosine	5	0.3	32	8	2	0.5	300	8	1	0	128	100
Knapsack	cosine	10	0.1	5	1	10	2	9	8	1	0	128	100
Energy scheduling	cosine	10	0.1	5	1	10	2	9	8	1	0	128	100
Matching	cosine	10	0.1	5	1	10	2	512	8	1	0	17	100
In-constraint Prediction													
KP capacity	constant	0.5	0.5	1.0	1.0	0.5	0.5	6	8	1	0	32	11,500
KP weight	constant	0.5	0.5	1.0	1.0	0.5	0.5	270	8	1	0	32	11,500
WSMC 5×25	constant	0.5	0.5	1.0	1.0	0.5	0.5	30	8	1	0	32	11,500
WSMC 10×50	constant	0.5	0.5	1.0	1.0	0.5	0.5	60	8	1	0	32	11,500
Districting													
Tuned	linear	0.174	0.174	86.4	19.8	18.3	18.3	512	512	4	0.9	100	30
Untuned	cosine	10	0.1	5	1	10	2	512	512	1	0	100	30
Brass Alloy Production													
Brass alloy	linear	0.707	0.707	2.67	1.26	461	461	128	8	2	0	32	220

used for the reported experiments:

- Classical OR: Seeds $\in [0, 9]$
- In-constraint Predictions: data-set seeds 0 to 4 times split seeds 0 to 2 (15 runs)
- Districting: Seeds $\in 3, 5, 7, 11, 13, 17, 19, 42, 71, and 1234$.
- Brass Alloy Production: Seeds $\in [11, 15]$