REINFORCEMENT LEARNING TO ACCELERATE PRIMAL-DUAL HYBRID GRADIENT FOR LINEAR PROGRAMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Primal–dual hybrid gradient (PDHG) methods solve large-scale linear programs (LPs) using GPU-friendly matrix–vector products and projections, but their practical performance depends on coordinating algorithm parameters, acceleration, and restarts. We introduce GALLOP, which uses reinforcement learning to jointly learn continuous algorithm parameters and discrete restart decisions without differentiating through the solver. Its generalized accelerated PDHG update combines separate primal and dual extrapolation, history corrections, and restart anchoring with independently adjustable coefficients. We train a dimension-agnostic feedback policy using a groupwise proximal policy optimization objective that clips likelihood ratios separately for different control groups and excludes inactive acceleration controls on restart transitions. We evaluate GALLOP on six LP families and a public item-placement benchmark. On the main evaluation settings across the six families, GALLOP reduces iteration counts by factors of 1.9–5.6 and achieves up to a $16.0\times$ speedup in algorithm wall-clock time over MPAX. With one policy trained per family, the learned policies generalize without retraining to within-family LPs $3\times$–$400\times$ larger than the largest training instances, including Transport LPs with 10.24 million variables.

1 INTRODUCTION

Linear programming (LP) is a fundamental problem class in constrained optimization (Dantzig, 1963). The simplex and interior-point methods are widely used to solve LPs (Dantzig, 1963; Wright, 1997). Recently, due to the scalability limitations of these traditional methods (Applegate et al., 2026), primal–dual hybrid gradient (PDHG) method (Chambolle & Pock, 2011) based LP algorithms have been explored for large-scale problems (Applegate et al., 2021; Lu & Yang, 2025; Lu et al., 2025). PDHG iteration consists of matrix–vector products and projections which are inexpensive and GPU-friendly. However, PDHG can require a large number of iterations to attain high-accuracy solutions (Applegate et al., 2023).

PDHG-based LP solvers (Lu et al., 2025; Zhou et al., 2025) improve their practical convergence through heuristic adjustment of algorithmic parameters, such as the step size and primal weight, and acceleration techniques, such as restarted Halpern acceleration (Halpern, 1967; Lu & Yang, 2024) and Anderson acceleration (Anderson, 1965). Many of these heuristics and acceleration schemes are motivated by theoretical convergence analyses (Lu & Yang, 2024; Applegate et al., 2023; Zhou et al., 2025). However, existing global convergence guarantees do not cover the full combination of heuristic adaptations used in practical solvers, whose effectiveness has been demonstrated empirically (Lu et al., 2023; 2025). Beyond convergence guarantees, a separate design challenge is to determine how these adaptations should be selected and coordinated to reduce solution time for a given family of LPs. This leaves an opportunity to further accelerate PDHG-based LP solvers by learning algorithm parameters and heuristic rules from LP data.

Learning to optimize (L2O) uses problem data to accelerate optimization (Gregor & LeCun, 2010). We focus on L2O methods that use a learned policy throughout the solve until convergence, without handing the remaining iterations to an external solver. Venkataraman & Amos (2021) and Martin & Belgioioso (2026) use algorithm unrolling for fixed-point optimization. However, their neural network designs are tied to the problem dimension and therefore do not support size generalization. PDHG-Net (Li et al., 2024) unrolls the basic PDHG iteration without incorporating discrete restart

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.

decisions or the additional acceleration schemes used by practical PDHG-based solvers. It does not run the learned updates until convergence, but instead uses the network output to warm-start PDLP.

RLQP (Ichnowski et al., 2021) uses reinforcement learning (RL) to learn the penalty parameter in OSQP (Stellato et al., 2020). While effective, RLQP applies model-free RL to tune continuous parameters of an iterative algorithm whose updates can be differentiated. In this setting, deep unfolding enables gradients of the optimization objective to be propagated directly through the solver, avoiding the additional value-function estimation and exploration required by RL. For example, DeepQP (Saravanos et al., 2025) unfolds OSQP and learns its algorithm parameters end-to-end, providing a more direct gradient-based alternative.

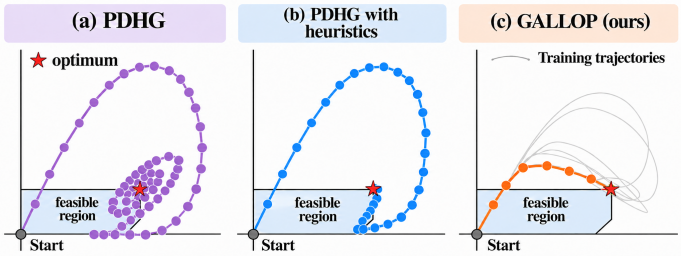


Figure 1: Conceptual comparison of (a) vanilla PDHG, (b) PDHG with heuristic acceleration and restarts, and (c) GALLOP with learned solver parameters and restarts.

To solve large LPs faster, we learn a policy that jointly controls continuous algorithm parameters and discrete restart decisions in a PDHG-based LP algorithm. We use RL to learn these controls jointly because ordinary backpropagation through the solver does not directly provide gradients for hard restart decisions. The policy uses a fixed number of solver-state features and outputs a fixed number of controls, independent of the problem size. This dimension-agnostic design allows us to train on smaller LPs and apply the learned policy to larger LPs without retraining.

We augment PDHG with a generalized acceleration step and a restart mechanism, and refer to the resulting update scheme as **generalized accelerated PDHG (GA-PDHG)**. This step combines separate primal and dual extrapolation along the current PDHG update, history corrections from recent iterate differences, and a correction toward the restart anchor. We learn the coefficients of these terms together with the step size, primal weight, and restart decision. These controls provide a broad action space for RL to explore combinations of acceleration and restarting. We call our proposed algorithm **GALLOP**, short for **G**eneralized **A**cceleration and Reinforcement **L**earning for **L**inear **O**ptimization with **P**DHG.

In experiments on six LP families, we demonstrate GALLOP is $1.9\times$–$5.6\times$ faster in terms of number of iterations relative to MPAX's restarted reflected Halpern solver (Lu et al., 2024). Algorithm wall-clock speedups reach $16.0\times$ on Random LPs with $300{,}000$ variables, $10.1\times$ on Auction LPs with $358{,}400$ variables, and $3.7\times$ on Transport LPs with 10.24 million variables. Our trained policies generalize to problems $3\times$–$400\times$ larger than the largest training size in a zero shot setting, using the family-specific size measures. In particular, the Transport policy trained on LPs with at most $25{,}600$ variables, scales to LPs with 10.24 million variables, a $400\times$ size generalization without retraining. Ablations on Transport show that all five learned control groups contribute to reducing iteration counts, including the generalized-acceleration coefficients. Our main contributions are as follows.

- We propose **GA-PDHG**, a generalized accelerated PDHG update scheme that combines independently controlled primal and dual extrapolation, history corrections, and a restart-anchor correction. This parameterization encompasses established acceleration schemes and their combinations, providing a broad search space for RL to discover acceleration strategies.
- We introduce **GALLOP**, an RL-based L2O framework for large-scale LPs. To accommodate the conditional discrete-continuous action space of GA-PDHG, we develop a groupwise PPO objective that clips likelihood ratios separately for each control group and masks inactive acceleration controls on restart transitions.
- We demonstrate acceleration and size generalization across six LP families, with iteration-count reductions by factors of 1.9–5.6 and up to a $16.0\times$ speedup in algorithm wall-clock time over MPAX. The learned policies generalize to LPs $3\times$–$400\times$ larger than the largest training instances under family-specific size measures, including Transport LPs with 10.24 million variables, without retraining.

2 PRIMAL–DUAL HYBRID GRADIENT FOR LARGE-SCALE LINEAR PROGRAMMING

PDLP (Applegate et al., 2021) is a first-order LP solver based on the primal–dual hybrid gradient (PDHG) method (Chambolle & Pock, 2011). Its updates use matrix–vector products and projections, avoiding matrix factorization. Basic PDHG can require many iterations to reach high accuracy. When PDHG is applied to LP, acceleration techniques and heuristics are introduced to reduce the number of iterations. This section presents the LP formulation and PDHG updates and reviews these enhancements in different PDLP variants.

2.1 PROBLEM FORMULATION: LINEAR PROGRAMMING

We consider the LP

$$\min_{x \in X} \quad c^\top x \quad \text{subject to} \quad Px = q, \quad Gx \geq h, \quad X = \{x : l \leq x \leq u\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is the primal variable and $c \in \mathbb{R}^n$ is the objective coefficient vector. The LP has m_e equality constraints specified by $P \in \mathbb{R}^{m_e \times n}$ and $q \in \mathbb{R}^{m_e}$, and m_i inequality constraints specified by $G \in \mathbb{R}^{m_i \times n}$ and $h \in \mathbb{R}^{m_i}$. The primal variable x is bounded with X where $l \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$, and $l \leq u$ componentwise. Define $A = [P^\top \; G^\top]^\top \in \mathbb{R}^{m \times n}$ and $b = [q^\top \; h^\top]^\top \in \mathbb{R}^m$, where $m = m_e + m_i$. The corresponding dual vector is $y = [(y^{\text{eq}})^\top \; (y^{\text{in}})^\top]^\top \in \mathbb{R}^m$, with equality multipliers $y^{\text{eq}} \in \mathbb{R}^{m_e}$ and inequality multipliers $y^{\text{in}} \in \mathbb{R}^{m_i}_+$. Linear programs (LPs) are commonly solved with the simplex (Dantzig, 1963) or interior-point methods (Wright, 1997). Standard implementations of these methods include matrix factorizations, whose computational and memory costs can become prohibitive at large scales.

2.2 PRIMAL–DUAL HYBRID GRADIENT FOR LINEAR PROGRAMMING

PDHG uses proximal gradients of the Lagrangian to update the primal and dual variables without matrix factorization. To derive these first-order updates, we express Eq. (1) as the equivalent convex–concave min-max problem

$$\min_{x \in X} \max_{y \in Y} \quad L(x, y) := c^\top x + (y^{\text{eq}})^\top (q - Px) + (y^{\text{in}})^\top (h - Gx) = c^\top x - y^\top Ax + b^\top y, \tag{2}$$

where $Y := \mathbb{R}^{m_e} \times \mathbb{R}^{m_i}_+$ leaves the equality multipliers unrestricted and the inequality multipliers nonnegative. For fixed $y \in Y$, minimizing the Lagrangian over X gives the dual objective

$$\inf_{x \in X} L(x, y) = b^\top y + \inf_{x \in X} (c - A^\top y)^\top x. \tag{3}$$

For the dual objective to be finite, the reduced cost $c - A^\top y$ must satisfy the sign conditions imposed by X. We denote the set of such reduced costs by Λ.

Throughout the paper, $z := (x^\top, y^\top)^\top \in \mathbb{R}^{n+m}$ denotes a stacked primal–dual vector. Subscripts and accents on z apply to both blocks x and y. At iteration t, $\hat{z}_t$ is the PDHG input and $z_t \in X \times Y$ is its output. At iteration t, projected descent and ascent with primal extrapolation give

$$x_t = \text{proj}_X \left[\hat{x}_t - \frac{\eta}{\omega} (c - A^\top \hat{y}_t) \right], \tag{4a}$$

$$y_t = \text{proj}_Y \left[\hat{y}_t + \omega\eta \left(b - A \left(2x_t - \hat{x}_t \right) \right) \right]. \tag{4b}$$

Here proj denotes Euclidean projection, $\eta > 0$ is the step size, and $\omega > 0$ is the primal weight, giving primal and dual step sizes η/ω and $\eta\omega$, respectively. Writing $\mathcal{H}$ for this PDHG map, we have $z_t = \mathcal{H}(\hat{z}_t; \omega, \eta)$. With iteration-dependent parameters $\eta_t, \omega_t > 0$, $z_{t+1} = \mathcal{H}(\hat{z}_{t+1}; \omega_t, \eta_t)$. Thus the current pair $(\hat{z}_t, z_t)$ was computed with $(\omega_{t-1}, \eta_{t-1})$, with (ω_0, η_0) reserved for initialization. For any $\omega, \eta > 0$, z is a fixed point of the PDHG map, i.e. $\mathcal{H}(z; \omega, \eta) = z$, if and only if z satisfies the KKT conditions of Eq. (1).

PDHG enhancements for LP. PDHG-based LP solvers choose a constant step size or adaptively choose it through trial-update acceptance tests (Chambolle & Pock, 2011; Applegate et al.,

2021). The primal weight is fixed or adjusted at restarts or fixed intervals to balance primal–dual displacements, typically through exponential moving averaging (EMA) in log space or proportional–integral–derivative (PID) control (Lu et al., 2025). Restarting (Hinder & Lubin, 2020) resets the averaging or acceleration state from a selected iterate after sufficient progress, a loss of progress following an initial decrease, or an excessively long epoch. The progress is measured by the normalized duality gap, KKT error, or fixed-point residual (Applegate et al., 2023; Lu & Yang, 2024). Acceleration uses step-size-weighted averaging (Applegate et al., 2021), Halpern anchoring with a diminishing anchor weight and optional reflection, or Anderson history corrections computed by least-squares residual minimization (Anderson, 1965; Zhou et al., 2025). With suitable parameter choices and restart rules, restarted averaging and Halpern acceleration exploit LP sharpness to achieve accelerated linear convergence (Applegate et al., 2023; Lu & Yang, 2024). See Appendix C for the details on these enhancements across representative solvers.

3 Reinforcement Learning for PDHG with Acceleration

The enhancement rules reviewed in PDHG-based LP solvers are hand-designed feedback controls based on solver state. Their criteria address individual decisions, such as balancing primal–dual displacements, testing restart thresholds, or minimizing a local residual model. These heuristics motivate exploring whether a more expressive policy that coordinates these decisions using a broader set of solver-state features can further accelerate convergence. We therefore learn such a policy from solver trajectories using a common reward that induces faster convergence.

3.1 PDHG with Generalized Acceleration

To enable a learned policy to explore both established acceleration rules and their combinations, we introduce a family of accelerated PDHG updates. Specifically, we combine extrapolation, history corrections, and anchoring with independently adjustable coefficients to formulate **generalized accelerated PDHG (GA-PDHG)**. For a fixed, finite memory length $k \geq 0$, define the output-history matrix $\mathbf{K}_t := [z_t - z_{t-1} \quad \cdots \quad z_{t-k+1} - z_{t-k}] \in \mathbb{R}^{(n+m)\times k}$. Unavailable differences and those discarded at a restart are replaced by zero. The vector $z_\star$ is the fixed anchor within the current epoch, and s counts the non-restart iterations within the current epoch. The coefficients $\alpha_t^x, \alpha_t^y \geq -1$ control primal and dual relaxation or extrapolation, $\boldsymbol{\beta}_t \in \mathbb{R}^k$ controls the history correction, and $\zeta_t \in \mathbb{R}$ controls the anchor correction. Let I_d denote the d-dimensional identity matrix and $e_t \in \{0, 1\}$ the restart decision. When $e_t = 0$ (no restart), the GA-PDHG update is

$$\hat{z}_{t+1} = z_t + \begin{pmatrix} \alpha_t^x I_n & 0 \\ 0 & \alpha_t^y I_m \end{pmatrix} (z_t - \hat{z}_t) + \mathbf{K}_t \boldsymbol{\beta}_t + \zeta_t (z_\star - z_t), \tag{5a}$$

$$z_{t+1} = \mathcal{H}(\hat{z}_{t+1}; \omega_t, \eta_t). \tag{5b}$$

When $e_t = 1$ (restart), the restart branch sets $\hat{z}_{t+1} = z_t$. Note that GA-PDHG always ends with a PDHG mapping and takes its output as the iterate, allowing the acceleration coefficients to explore combinations beyond convex combinations and generate proposals outside $X \times Y$ while ensuring $z_{t+1} \in X \times Y$. After a restart, we set $z_\star \leftarrow z_{t+1}$, $\mathbf{K}_{t+1} \leftarrow 0$, and $s \leftarrow 0$.

Table 1 shows how coefficient choices in Eq. (5a) correspond to five acceleration and relaxation schemes for fixed-point problems. GA-PDHG thus offers a unified search space that encompasses these schemes as special cases. For general coefficient choices, Proposition 3.1 gives sufficient conditions under which the output of a convergent GA-PDHG trajectory is primal–dual optimal in the limit. Note that we do not claim global convergence for arbitrary learned control sequences. See Appendix B for details.

Table 1: Acceleration and relaxation schemes for fixed-point problems via GA-PDHG coefficients.

Method	$\alpha_t^x = \alpha_t^y = \alpha_t$	$\boldsymbol{\beta}_t$	ζ_t
Halpern (Halpern, 1967)	0	$\mathbf{0}$	$\frac{1}{s+2}$
Reflected Halpern (Lu & Yang, 2024)	$\frac{s+1}{s+2}$	$\mathbf{0}$	$\frac{1}{s+2}$
Anderson (Anderson, 1965)	0	$\boldsymbol{\beta}_t^{\mathrm{AA}}$	0
Krasnosel'skiĭ–Mann (KM) (Krasnosel'skii, 1955; Mann, 1953)	$\lambda - 1$	$\mathbf{0}$	0
Fast KM (Boţ & Nguyen, 2023)	$\frac{\nu}{2(s+\nu)} - 1$	$\frac{s}{s+\nu}$	0

$\boldsymbol{\beta}_t^{\mathrm{AA}}$ is obtained by solving a least squares problem. $\lambda \in (0, 2)$. For Fast KM, $k = 1$, $\nu > 2$ and $s \geq 1$.

Proposition 3.1 (Optimality of convergent GA-PDHG). *Suppose $\boldsymbol{\beta}_t$ is bounded, η_t, ω_t are uniformly positive and bounded, and there exists $\delta > 0$ such that $1 + \alpha_t^x, 1 + \alpha_t^y \geq \delta$ whenever $e_t = 0$. If both $\{\hat{z}_t\}$ and $\{z_t\}$ converge to finite limits and $\|\zeta_t (z_\star - z_t)\|_2 \longrightarrow 0$ along the trajectory, then the limit of $\{z_t\}$ is a primal–dual optimal solution of Eq. (1).*

Selecting the coefficients of GA-PDHG can be viewed as an optimal control problem in which the GA-PDHG iteration defines a discrete-time dynamical system and the objective is to accelerate convergence. Figure 2 illustrates a feedback control loop in which the solver inputs are selected based on the current solver state. At iteration t, the solver state S_t contains the LP data, input–output pair $(z_t, \hat{z}_t)$, anchor $z_\star$, and history $\mathbf{K}_t$, the previous control variables, convergence records, and iteration counters. From this state, we construct an observation $o_t = \mathrm{Obs}(S_t)$ whose dimension is independent of the problem size. A policy network inputs o_t and outputs an action a_t comprising the restart decision e_t and GA-PDHG coefficients $\alpha_t^x, \alpha_t^y, \boldsymbol{\beta}_t, \zeta_t$, and PDHG parameters ω_t, η_t. Using this action, the solver performs the GA-PDHG update and updates its diagnostics and counters to obtain S_{t+1}. The next observation $o_{t+1} = \mathrm{Obs}(S_{t+1})$ is then constructed and passed to the policy for the next decision.

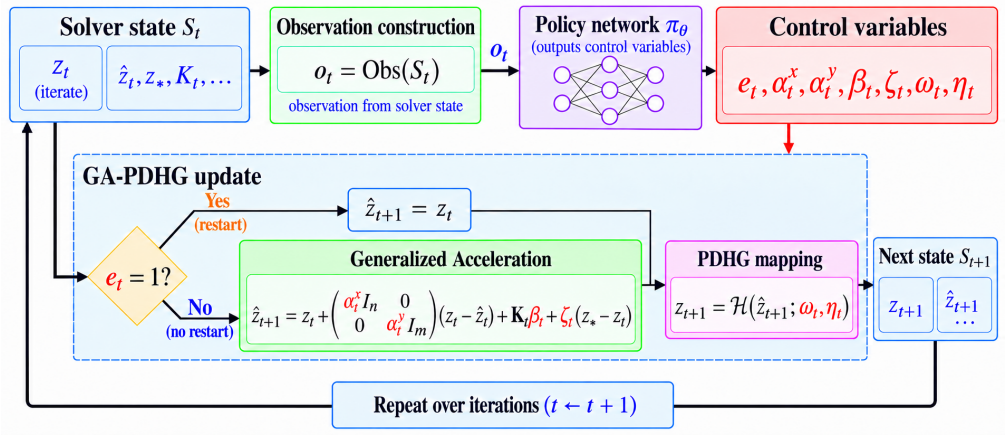


Figure 2: Feedback control loop for GA-PDHG. Red symbols denote control variables selected by the policy network from solver observations.

3.2 Reinforcement Learning Formulation

We use reinforcement learning (RL) to train the feedback policy π_θ for GA-PDHG and refer to the resulting solver as **GALLOP**. Each LP solve corresponds to an episode, during which the policy selects solver controls at each iteration to maximize the expected cumulative reward. To support size generalization, we design the observation and control spaces to have dimensions independent of n and m, allowing the same policy to be evaluated on LPs of different sizes. At each decision step, the policy receives the observation $o_t = \mathrm{Obs}(S_t)$, which summarizes the solver state. Log-normalized primal and dual residuals, the duality gap, their changes, and fixed-point residual progress describe current error and recent improvement. Iteration count, restart age, progress from the anchor, primal–dual displacement balance, relative primal weight, and its one-step change characterize the current epoch. A normalized history Gram matrix, inner products with the PDHG direction, and anchor magnitude and alignment describe the scales and relative directions of the available corrections. Given this observation, the policy selects controls in a conditional hybrid action space, in which the discrete restart decision determines which continuous controls affect the solver transition. The policies for e_t, η_t, and ω_t are trained using both restart and non-restart transitions, whereas the policies for the acceleration coefficients $(\alpha_t^x, \alpha_t^y, \boldsymbol{\beta}_t, \zeta_t)$ are trained only on non-restart transitions.

We measure the quality of iterate $z \in X \times Y$ in original LP coordinates using the primal residual, dual residual, and duality-gap measure. Let $\bar{c} := c - A^\top y$ be the reduced cost and $\lambda := \mathrm{proj}_\Lambda(\bar{c})$ its projection onto Λ. The primal and dual residuals and the duality-gap measure are defined as

$$r_p(z) := \mathrm{proj}_Y(b - Ax), \quad r_d(z) := \bar{c} - \lambda, \quad r_g(z) := c^\top x - \left(b^\top y + \min_{x \in X} \lambda^\top x\right). \tag{6}$$

We denote the logarithms of their normalized magnitudes by $p(z)$, $d(z)$, and $g(z)$ and define the merit function as

$$M^{\mathrm{rel}}(z) := \max\{p(z), d(z), g(z)\}. \tag{7}$$

For the target tolerance ϵ_{tol}, an episode ends successfully when the convergence criterion $M^{\mathrm{rel}}(z_{t+1}) \le \log \epsilon_{\mathrm{tol}}$ is met. If the episode does not converge, it is truncated after $T_{\max}$ controlled iterations. With $[a]^+ := \max\{a, 0\}$, we use the following reward function to encourage reaching tolerance in fewer iterations,

$$r_t = M^{\mathrm{rel}}(z_t) - M^{\mathrm{rel}}(z_{t+1}) - \chi\left[M^{\mathrm{rel}}(z_{t+1}) - \log\epsilon_{\mathrm{tol}}\right]^+ - \frac{1}{T_{\max}} + \mathbf{1}\left[M^{\mathrm{rel}}(z_{t+1}) \le \log\epsilon_{\mathrm{tol}}\right]. \tag{8}$$

Here, $\chi \ge 0$ weights the excess-error penalty $\left[M^{\mathrm{rel}}(z_{t+1}) - \log\epsilon_{\mathrm{tol}}\right]^+$ and $\mathbf{1}[\cdot]$ is the indicator function. The difference in merit $M^{\mathrm{rel}}(z_t) - M^{\mathrm{rel}}(z_{t+1})$ rewards a reduction in KKT error from one iteration to the next, even when the error remains above the target tolerance. The excess-error penalty encourages reducing errors earlier along the trajectory. Summed over an episode without discounting, the last two terms contribute $1 - T/T_{\max}$ for success after T iterations and -1 for truncation without success, favoring successful solves that reach tolerance in fewer iterations.

3.3 Policy Learning with Proximal Policy Optimization

We design a groupwise proximal policy optimization (PPO) objective that reflects both conditional activation and the distinct control roles in GA-PDHG. We assign actions to five groups, $\mathcal{G} = \{e, \beta, (\alpha, \zeta), \eta, \omega\}$, with a Bernoulli policy for restart and Gaussian policies for the continuous controls. For trajectories collected under fixed policy parameters θ_{old}, let a_t^j be the recorded action of group $j \in \mathcal{G}$, and let $\pi_\theta^j(a_t^j|o_t)$ denote its probability mass or density under policy parameters θ. On a transition where group j is active, its likelihood ratio is $\rho_t^j := \frac{\pi_\theta^j(a_t^j|o_t)}{\pi_{\theta_{\mathrm{old}}}^j(a_t^j|o_t)}$. On restart transitions, we marginalize the unused acceleration controls in groups β and (α, ζ) out of the likelihood and set $\rho_t^j \equiv 1$ for these inactive groups, so they contribute no policy gradient. A shared state-value critic $V^\phi(o_t)$ provides value estimates for generalized advantage estimation (GAE) (Schulman et al., 2016). Let $\tilde{\Psi}_t$ denote the resulting advantage normalized over the collected transitions. All groups use this common advantage to learn from the return of their joint execution. The objective of our groupwise PPO is

$$\mathcal{J}(\theta) = \sum_{j\in\mathcal{G}} \hat{\mathbb{E}}_t\left[\min\left(\rho_t^j\tilde{\Psi}_t,\ \mathrm{clip}\left(\rho_t^j, 1-\varepsilon^{\mathrm{clip}}, 1+\varepsilon^{\mathrm{clip}}\right)\tilde{\Psi}_t\right)\right]. \tag{9}$$

Here, $\hat{\mathbb{E}}_t$ averages over the collected transitions. $\varepsilon^{\mathrm{clip}} > 0$ is the clipping parameter, which is applied to each group's likelihood ratio ρ_t^j. For the standard PPO (Schulman et al., 2017), clipping is applied to the joint action likelihood ratio $\prod_{j\in\mathcal{G}} \rho_t^j$. Under the joint clipping, changes in one group's likelihood can suppress the gradient for the other group's policy even when the the other group's likelihood ratio is close to one. In particular, restart probability is often below 10%, hence a small logit change has little effect on the non-restart likelihood. On the other hand β is sampled from k-dimensional Gaussian, which accumulates coordinate contributions in its log-likelihood ratio.

Table 2: Clipping ratio (%).

Problem class	β	e
Transport	3.79	0.35
Auction	8.99	0.94
Random LPs	2.01	0.13

Table 2 shows that clipping of β happens more frequently than restart e, motivating separate clipping decisions. Under our groupwise PPO objective in Eq. (9), each group's clipping depends only on its own ratio, so a change in one group's ratio cannot by itself clip another group's ratio. Similar discussions can be found in Fan et al. (2019). At inference, we fix θ, threshold the restart probability at 0.5, and transform the Gaussian means into continuous controls. GALLOP then executes the feedback loop in Figure 2 until the KKT tolerance is reached or the iteration limit is exhausted. The critic is used only during training. See Appendix F for details.

Off-policy actor–critic methods, such as SAC (Haarnoja et al., 2018), can also be extended to hybrid action spaces (Delalleau et al., 2019). However, these methods require the hybrid structure to be represented in both the policy and the action-value critic, including the treatment of inactive continuous controls. Because environment interaction is inexpensive in our setting, the sample-efficiency gains from off-policy data reuse offer limited benefit.

4 EXPERIMENTS

4.1 EXPERIMENT SETTING

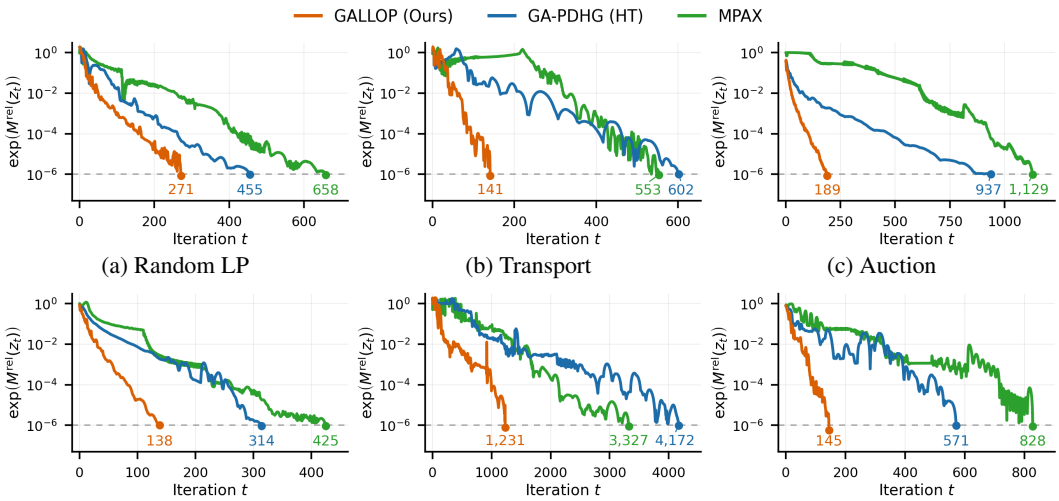
We evaluate our method on six different LP families in Table 3; Random LP, Transport, Auction, Min-cost flow, Robust inventory management, and L_∞ regression. We train one policy per family and evaluate on unseen LPs with varying sizes within the same family. Appendix D gives the problem details and size lists. We compare GALLOP with MPAX's r^{2}HPDHG solver (Lu et al., 2024) and GA-PDHG (HT), a hand-tuned baseline that shares our solver implementation. GA-PDHG (HT) uses the reflected Halpern coefficients in Table 1, and $\eta_t = 0.998/\|A\|_2$, with restarting rules and PID controller weight updates from cuPDLPx (Lu et al., 2025). Because MPAX checks convergence every 100 iterations by default whereas our implementation checks every iteration, we use saved trajectories to identify the first iterate satisfying $M^{\mathrm{rel}}(z_t) \le \log \epsilon_{\mathrm{tol}}$, with $\epsilon_{\mathrm{tol}} = 10^{-6}$. We report iterations and matrix–vector products to this common tolerance, including products from MPAX's rejected adaptive steps. Algorithm wall clock measures GPU execution of the main solve loop after compilation and excludes JIT compilation and preconditioning.

Table 3: Density $\mathrm{nnz}(A)/(mn)$ and m/n at each family's largest training size.

Family	Density	m/n
Random LP	1.28%	0.5
Transport	0.625%	0.0125
Auction	0.313%	0.25
Min-cost flow	0.0417%	0.167
Robust inventory	0.326%	0.341
L_∞ regression	100%	26.7

4.2 CONVERGENCE AND SIZE GENERALIZATION

Figure 3 plots the exponentiated merit $\exp(M^{\mathrm{rel}}(z_t))$ against iteration count. GALLOP reaches the tolerance 10^{-6} in fewer iterations than both baselines on all six plotted instances. Table 4 reports medians over four unseen LPs per family and size.



(a) Random LP (b) Transport (c) Auction

(d) Min-cost flow (e) Robust inventory (f) L_∞ regression

Figure 3: Convergence on one unseen LP per family at the largest training size. Labels mark the first iteration meeting tolerance 10^{-6} (dashed line).

The first row for each family corresponds to the largest training size ($1\times$), and the following rows evaluate the same policy on larger LPs without retraining. At $1\times$, GALLOP requires fewer iterations and matrix–vector products for all six families. GALLOP generalizes to $15\times$ for Random LP, $400\times$ for Transport, $14\times$ for Auction, $12\times$ for Min-cost flow, $3\times$ for Robust inventory, and $7\times$ for L_∞ regression, with gains in all three metrics at these sizes. Size multipliers are measured by the number of variables, except for Robust inventory (item count) and L_∞ regression (constraint count at fixed $n = 9$).

Table 4: Test results (four-LP medians). The "Size ratio" column reports the ratio of test size to each family's largest training size. Size is measured in n except for robust inventory (item count N) and L_∞ regression (m at fixed $n = 9$). Ours denotes GALLOP, HT denotes GA-PDHG (HT).

			Iterations			Matrix–vector products			Algorithm wall clock (ms)		
Family	(m, n)	Size ratio	Ours	HT	MPAX	Ours	HT	MPAX	Ours	HT	MPAX
Random LP	(10,000, 20,000)	1×	**284**	440	664	**570**	883	1,384	**110**	140	408
	(30,000, 60,000)	3×	**280**	421	663	**561**	844	1,384	**151**	199	1,175
	(150,000, 300,000)	15×	**290**	428	702	**582**	859	1,468	**450**	626	7,202
Transport	(320, 25,600)	1×	**164**	633	548	**329**	1,268	1,150	**39**	107	87
	(1,600, 640,000)	25×	**153**	533	536	**308**	1,068	1,128	**102**	286	308
	(6,400, 10,240,000)	400×	**200**	570	524	**403**	1,142	1,105	**1,998**	5,222	7,334
Auction	(6,400, 25,600)	1×	**190**	928	980	**381**	1,859	2,025	**70**	256	198
	(32,000, 128,000)	5×	**190**	923	994	**382**	1,848	2,055	**84**	316	479
	(89,600, 358,400)	14×	**192**	926	1,066	**387**	1,853	2,198	**134**	521	1,352
Min-cost flow	(4,800, 28,800)	1×	**164**	302	440	**329**	606	928	**49**	68	76
	(24,000, 144,000)	5×	**234**	356	444	**469**	715	939	**80**	94	109
	(57,600, 345,600)	12×	**242**	370	482	**485**	741	1,010	**115**	142	147
Robust inventory	(1,500, 4,401)	1×	**1,905**	4,266	3,875	**3,812**	8,535	7,872	507	915	**432**
	(2,980, 8,801)	2×	**2,311**	7,768	5,348	**4,624**	15,537	10,854	**570**	1,494	661
	(4,460, 13,201)	3×	**2,874**	25,798	6,121	**5,749**	51,599	12,391	**858**	5,978	895
L_∞ regression	(240, 9)	1×	**150**	586	834	**303**	1,175	1,734	**24**	54	82
	(1,200, 9)	5×	**236**	673	760	**475**	1,348	1,583	**41**	78	69
	(1,680, 9)	7×	**358**	752	782	**718**	1,505	1,626	**62**	93	78

Figure 4 highlights the corresponding algorithm wall clock savings for random LP and Transport. At 400×, Transport takes 1,998 ms for GALLOP, compared with 5,222 ms for GA-PDHG (HT) and 7,334 ms for MPAX. Tables 11 and 12 provide results for all test sizes.

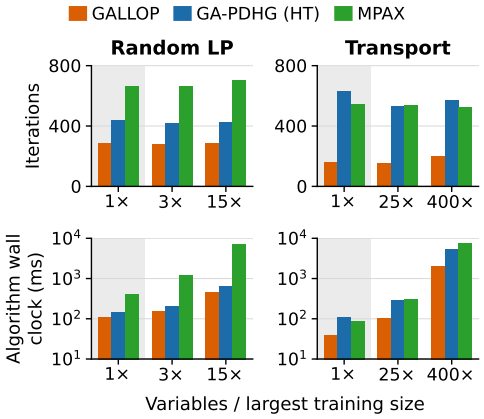


Figure 4: Size generalization (four-LP medians). Shading marks the largest training size.

4.3 Ablation Study

We evaluate the contribution of each learned control group on Transport at the largest training size, $m = 320$ and $n = 25{,}600$. At evaluation time, we replace one control group with a hand-tuned rule and retain the other learned controls without retraining. We also test Anderson acceleration in place of the learned history coefficients. Table 5 reports the median number of iterations to reach $\epsilon_{\text{tol}} = 10^{-6}$ on all four LPs. Among substitutions with hand-tuned rules, replacing the learned step size has the largest effect, increasing the iterations from 165 to 457. Replacing any of the other four groups increases the median by 10%–36%. These results support the effectiveness of GA-PDHG and retaining all five learned control groups in the trained policy. Since the remaining controls are kept fixed rather than retrained, this experiment measures reliance of the learned joint policy on each control group, rather than the best performance achievable after removing that group and retraining.

Table 5: Transport control ablations (four-LP medians, without retraining).

Control group	Replaced by	Iterations
— (GALLOP as trained)		**165.0**
Step size η_t	$0.998/\|\|A\|\|_2$	457.0
Primal weight ω_t	PID rule Eq. (65)	224.5
$\alpha_t^x, \alpha_t^y, \zeta_t$	Halpern	209.0
History $\boldsymbol{\beta}_t$	$\boldsymbol{\beta}_t = \mathbf{0}$	190.5
History $\boldsymbol{\beta}_t$	Anderson	908.0
Restart flag e_t	three conditions Eq. (66)	181.0
All groups	GA-PDHG (HT)	633.0

4.4 Comparison on Public dataset

We evaluate on the public item-placement benchmark from Li et al. (2024). IP-S has $(m, n) = (15{,}525, 31{,}350)$ and $\mathrm{nnz}(A) = 5{,}291{,}250$, while IP-L has $(m, n) = (91{,}575, 266{,}450)$ and $\mathrm{nnz}(A) = 94{,}826{,}250$.

We train GALLOP on 128 IP-S instances and evaluate the same policy on ten unseen instances of IP-S and IP-L. IP-L evaluation tests zero-shot size generalization without retraining.

Tables 6 and 7 report mean iteration counts for each solvers. GALLOP and MPAX use tolerance 10^{-6}. PDLP uses the released PDHG-Net settings with absolute tolerance 10^{-6} and relative tolerance 10^{-4}. This stopping criterion is looser than GALLOP's. PDLP is the OR-Tools 9.8.3296 version run on our machine. The IP-S result of PDHG-Net is quoted from Table 2 in Li et al. (2024), which predicts a initialization for PDLP, hence the label PDHG-Net + PDLP.

Table 6: IP-S. Mean iteration counts.

Solver	Iterations
GALLOP	**186.7**
MPAX	1,089.0
PDHG-Net + PDLP	422
PDLP	550.4

Table 7: Comparison results on IP-L. Zero-shot evaluation.

Solver	Iterations
GALLOP	**272.2**
MPAX	OOM
PDLP	851.2

OOM: out of memory on RTX 4090.

On IP-S, GALLOP requires $5.83\times$ and $2.95\times$ less number of iterations relative to MPAX and PDLP, respectively, and uses fewer iterations than the reported PDHG-Net + PDLP result. On IP-L, the reduction relative to PDLP is a factor of $3.13\times$. Its iteration count increases only $1.46\times$ despite $8.50\times$ more variables and $17.92\times$ more nonzeros, demonstrating size generalization on this benchmark.

5 Related Work

RL for discrete-continuous hybrid action spaces. P-DQN (Xiong et al., 2018) and MP-DQN (Bester et al., 2019) use action-value critics, unlike GALLOP's state-value critic. H-PPO (Fan et al., 2019) also uses a state-value critic and separately clips actor objectives; DISC (Han & Sung, 2019) instead clips coordinatewise importance ratios. GALLOP specializes actor-wise clipping to five functional solver-control groups, marginalizing acceleration controls on restart transitions while retaining step-size and primal-weight updates.

L2O for LP. Prior work learns simplex initial bases (Fan et al., 2023), imitates pivot experts (Liu et al., 2024), or emulates interior-point methods (Qian et al., 2024). These approaches target initialization or expert-derived decisions rather than jointly learning PDHG acceleration and restarting. More directly related, PDHG-Net (Li et al., 2024) unrolls PDHG to warm-start PDLP. GALLOP instead retains learned feedback throughout the solve, without handing subsequent iterations to an external solver.

RL in L2O. RL learns simplex pivot-rule selection (Suriyanarayana et al., 2022) and presolve routines (Kuang et al., 2025), addressing discrete solver decisions rather than hybrid iterative control. RLQP (Ichnowski et al., 2021) adapts continuous penalty parameters of the alternating direction method of multipliers (ADMM), whereas TFPnP (Wei et al., 2020) combines continuous parameter tuning with discrete termination for inverse imaging. GALLOP instead jointly controls PDHG step sizes, primal weights, generalized acceleration, and restarts through a dimension-agnostic policy. Its discrete decisions reset acceleration state rather than terminate optimization.

6 Conclusion and Limitations

We introduced GALLOP, which uses reinforcement learning to jointly control PDHG parameters, acceleration coefficients, and restart decisions. Its GA-PDHG update combines primal and dual extrapolation, history corrections, and an anchor correction in a common action space. A dimension-agnostic policy enables deployment on larger LPs without retraining. Across six LP families, GALLOP reduces iteration counts relative to MPAX and achieves algorithm wall-clock speedups of up to $16.0\times$. Policies trained on smaller LPs generalize to sizes up to $400\times$ larger, with additional evidence of size generalization on the public item-placement benchmark.

Limitations. Available GPU memory limited the maximum LP sizes evaluated in this study. Experiments with more GPU memory would allow us to test whether the observed size generalization extends to substantially larger LPs.

AI DISCLOSURE

In this work, we used generative AI tools for generate synthetic data sets (LP family generation), assist in the writing of proofs, refine hypotheses, design and provide feedback on research methodology and experiments, implement methods, assist with translation, interpret results (Required). We also used generative AI tools to create and modify scientific figures and images, suggest experimental parameters, create and edit software code, draft parts of a research paper, summarize and analyze existing literature, identify research gaps, suggest a structure for a research paper. We have not used AI to generate new research idea. LLM-generated code was verified and tested by the author. We take responsibility for final content of this work, including texts, claims or artifacts produced with the aid of generative AI.

REFERENCES

Brandon Amos. Tutorial on amortized optimization. *Foundations and Trends in Machine Learning*, 16(5):592–732, 2023. ISSN 1935-8237. doi: 10.1561/2200000102.

Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. *Advances in Neural Information Processing Systems*, 34:20243–20257, 2021.

David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. *Mathematical Programming*, 201(1):133–184, 2023. doi: 10.1007/s10107-022-01901-9.

David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, and Warren Schudy. PDLP: A practical first-order method for large-scale linear programming. *Mathematical Programming Computation*, 2026. doi: 10.1007/s12532-026-00309-2.

Craig J. Bester, Steven D. James, and George D. Konidaris. Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces. *arXiv preprint arXiv:1905.04388*, 2019. URL https://arxiv.org/abs/1905.04388.

Radu Ioan Boţ and Dang Khoa Nguyen. Fast Krasnosel'skiĭ–Mann algorithm with a convergence rate of the fixed point iteration of $o(1/k)$. *SIAM Journal on Numerical Analysis*, 61(6):2813–2843, 2023. doi: 10.1137/22M1504305.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40(1):120–145, 2011.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022.

George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

Olivier Delalleau, Maxim Peter, Eloi Alonso, and Adrien Logut. Discrete and continuous action representation for practical RL in video games. *arXiv preprint arXiv:1912.11077*, 2019. URL https://arxiv.org/abs/1912.11077.

Zhenan Fan, Xinglu Wang, Oleksandr Yakovenko, Abdullah Ali Sivas, Owen Ren, Yong Zhang, and Zirui Zhou. Smart initial basis selection for linear programs. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 9650–9664. PMLR, 2023. URL https://proceedings.mlr.press/v202/fan23d.html.

Zhou Fan, Rui Su, Weinan Zhang, and Yong Yu. Hybrid actor-critic reinforcement learning in parameterized action space. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 2279–2285. International Joint Conferences on Artificial Intelligence Organization, 2019. doi: 10.24963/ijcai.2019/316. URL https://www.ijcai.org/proceedings/2019/316.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on machine learning*, pp. 399–406, 2010.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Benjamin Halpern. Fixed points of nonexpanding maps. *Bulletin of the American Mathematical Society*, 73(6):957–961, 1967.

Seungyul Han and Youngchul Sung. Dimension-wise importance sampling weight clipping for sample-efficient reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2586–2595. PMLR, 2019. URL https://proceedings.mlr.press/v97/han19b.html.

John R. Hershey, Jonathan Le Roux, and Felix Weninger. Deep unfolding: Model-based inspiration of novel deep architectures, 2014.

Oliver Hinder and Miles Lubin. A generic adaptive restart scheme with applications to saddle point algorithms. *arXiv preprint arXiv:2006.08484*, 2020.

Jeffrey Ichnowski, Paras Jain, Bartolomeo Stellato, Goran Banjac, Michael Luo, Francesco Borrelli, Joseph Gonzalez, Ion Stoica, and Ken Goldberg. Accelerating quadratic optimization with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 21043–21055. Curran Associates, Inc., 2021.

Mark Aleksandrovich Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi matematicheskikh nauk*, 10(1):123–127, 1955.

Yufei Kuang, Xijun Li, Jie Wang, Fangzhou Zhu, Meng Lu, Zhihai Wang, Jia Zeng, Houqiang Li, Yongdong Zhang, and Feng Wu. Accelerate presolve in large-scale linear programming via reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(8): 6660–6672, 2025. doi: 10.1109/TPAMI.2025.3562286.

Bingheng Li, Linxin Yang, Yupeng Chen, Senmiao Wang, Haitao Mao, Qian Chen, Yao Ma, Akang Wang, Tian Ding, Jiliang Tang, and Ruoyu Sun. PDHG-unrolled learning-to-optimize method for large-scale linear programming. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29164–29180. PMLR, 2024.

Tianhao Liu, Shanwen Pu, Dongdong Ge, and Yinyu Ye. Learning to pivot as a smart expert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8073–8081, 2024. doi: 10.1609/aaai.v38i8.28646.

Haihao Lu and Jinwen Yang. Restarted Halpern PDHG for linear programming. *arXiv preprint arXiv:2407.16144*, 2024.

Haihao Lu and Jinwen Yang. cuPDLP.jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia. *Operations Research*, 73(6):3440–3452, 2025. doi: 10.1287/opre.2024.1069.

Haihao Lu, Jinwen Yang, Haodong Hu, Qi Huangfu, Jinsong Liu, Tianhao Liu, Yinyu Ye, Chuwen Zhang, and Dongdong Ge. cuPDLP-C: A strengthened implementation of cuPDLP for linear programming by C language. *arXiv preprint arXiv:2312.14832*, 2023. URL https://arxiv.org/abs/2312.14832.

Haihao Lu, Zedong Peng, and Jinwen Yang. MPAX: mathematical programming in JAX. *arXiv preprint arXiv:2412.09734*, 2024.

Haihao Lu, Zedong Peng, and Jinwen Yang. cuPDLPx: A further enhanced GPU-based first-order solver for linear programming. *arXiv preprint arXiv:2507.14051*, 2025.

W Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

Andrea Martin and Giuseppe Belgioioso. Learning to accelerate Krasnosel'skii–Mann fixed-point iterations with guarantees. *arXiv preprint arXiv:2601.07665*, 2026. URL https://arxiv.org/abs/2601.07665.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4556–4565. PMLR, 2019.

Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38(2):18–44, 2021. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2020.3016905.

Alex Oshin, Rahul Vodeb Ghosh, Augustinos D. Saravanos, and Evangelos A. Theodorou. Deep FlexQP: Accelerated nonlinear programming via deep unfolding. In *The Fourteenth International Conference on Learning Representations*, Rio de Janeiro, Brazil, 2026a. doi: 10.48550/arXiv.2512.01565.

Alex Oshin, Rahul Vodeb Ghosh, and Evangelos A. Theodorou. Scalable deep unfolding of conic optimizers. In *Proceedings of the 10th Conference on Robot Learning*. PMLR, 2026b. doi: 10.48550/ARXIV.2606.13825.

Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, pp. 1762–1769. IEEE, 2011.

Chendi Qian, Didier Chételat, and Christopher Morris. Exploring the power of graph neural networks in solving linear optimization problems. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1432–1440. PMLR, 2024. URL https://proceedings.mlr.press/v238/qian24a.html.

Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. *Institut National Polytechnique de Toulouse, Toulouse,...*, 2001.

Rajiv Sambharya and Bartolomeo Stellato. Learning algorithm hyperparameters for fast parametric convex optimization, 2024.

Rajiv Sambharya and Bartolomeo Stellato. Data-driven performance guarantees for classical and learned optimizers. *Journal of Machine Learning Research*, 26(171):1–49, 2025.

Augustinos Saravanos, Hunter Kuperman, Alex Oshin, Arshiya Taj Abdul, Vincent Pacelli, and Evangelos Theodorou. Deep distributed optimization for large-scale quadratic programming. In *International Conference on Learning Representations*, volume 2025, pp. 93580–93619, 2025.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

Michael Sucker, Jalal Fadili, and Peter Ochs. Learning-to-optimize with PAC-bayesian guarantees: Theoretical considerations and practical implementation. *Journal of Machine Learning Research*, 26(211):1–53, 2025.

Varun Suriyanarayana, Onur Tavaslıoğlu, Ankit B. Patel, and Andrew J. Schaefer. Reinforcement learning of simplex pivot rules: A proof of concept. *Optimization Letters*, 16(8):2513–2525, 2022. doi: 10.1007/s11590-022-01880-y.

Shobha Venkataraman and Brandon Amos. Neural fixed-point acceleration for convex optimization. *arXiv preprint arXiv:2107.10254*, 2021.

Kaixuan Wei, Angelica Aviles-Rivero, Jingwei Liang, Ying Fu, Carola-Bibiane Schönlieb, and Hua Huang. Tuning-free plug-and-play proximal algorithm for inverse imaging problems. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10158–10169. PMLR, 2020. URL https://proceedings.mlr.press/v119/wei20b.html.

Stephen J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.

Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018. URL https://arxiv.org/abs/1810.06394.

Yingxin Zhou, Stefano Cipolla, and Phan Tu Vuong. Anderson Accelerated Primal-Dual Hybrid Gradient for solving LP. *arXiv preprint arXiv:2508.08062*, 2025.

APPENDIX

A EXTENDED RELATED WORK

Learning to Optimize and Deep unfolding. Learning to optimize (L2O) leverages data-driven machine learning in order to develop and improve optimization methods. Key survey papers include (Chen et al., 2022) and (Amos, 2023). Deep unfolding is a model-based L2O approach that interprets each iteration of an optimizer as a layer of a neural network (Hershey et al., 2014). Learnable parameters are introduced and the optimizer is unrolled for a fixed number of iterations and trained using backpropagation-through-time, much like a recurrent neural network (Monga et al., 2021). The idea has seen great success in the signal and image processing communities after the seminal work by Gregor & LeCun (2010) on learned iterative shrinkage and soft-thresholding (LISTA). These approaches focus on learning iteration-varying parameters that can accelerate the optimizer, amortize the cost of tuning, and improve generalization on unseen instances (Sambharya & Stellato, 2024; Amos, 2023). Deep unfolding has been recently extended to general classes of convex optimization problems, such as linear programming (Li et al., 2024), quadratic programming (Saravanos et al., 2025; Oshin et al., 2026a), and conic programming (Oshin et al., 2026b). Another line of recent work has studied guarantees of learned optimizers, including PAC-Bayes generalization bounds (Sucker et al., 2025) and worst-case performance bounds (Sambharya & Stellato, 2025). However, deep unfolding often struggles to scale to high-dimensional problems in machine learning. This is primarily due to the intractable memory requirement of storing an entire trajectory of optimizer iterations in memory in order to backpropagate through them via supervised learning. These long-horizon unrolls are also numerically unstable, as they suffer from the vanishing/exploding gradient problem (Metz et al., 2019). Moreover, deep unfolding struggles to learn discrete decisions, such as when to restart in PDLP. Finally, supervised L2O requires optimal solutions, which are expensive to compute, especially for large-scale problems. These points make reinforcement learning a more suitable choice for the large-scale LPs that we target in this work.

B PROOFS FROM THE PAPER

B.1 FIXED POINTS OF PDHG AND LP OPTIMALITY

We first establish that solving the LP is equivalent to finding a fixed point of the PDHG operator $\mathcal{H}$. Fix $\omega, \eta > 0$ and write $\mathcal{H} = \mathcal{H}(\cdot; \omega, \eta)$. Let $Z^\star$ denote the set of primal–dual optimal solutions of Eq. (1). For $\alpha \in [0, 1]$, define $\mathcal{R}_\alpha(z) := (1 + \alpha)\mathcal{H}(z) - \alpha z$.

Define the primal–dual KKT mapping $\mathcal{F} : X \times Y \to 2^{\mathbb{R}^{n+m}}$ by

$$\mathcal{F}(z) = \mathcal{F}(x, y) := \begin{pmatrix} c - A^\top y + N_X(x) \\ Ax - b + N_Y(y) \end{pmatrix}, \tag{10}$$

where N_X and N_Y denote the normal-cone mappings of X and Y, respectively.

Lemma B.1 (KKT characterization)**.** *A primal–dual point $z^\star = (x^\star, y^\star)$ is optimal for Eq. (1) if and only if*

$$0 \in \mathcal{F}(z^\star). \tag{11}$$

Equivalently,

$$0 \in c - A^\top y^\star + N_X(x^\star), \tag{12}$$
$$0 \in Ax^\star - b + N_Y(y^\star), \tag{13}$$

where Eq. (13) is equivalent to

$$Px^\star = q, \tag{14}$$
$$Gx^\star \geq h, \tag{15}$$
$$y^{\text{in},\star} \geq 0, \tag{16}$$
$$y^{\text{in},\star} \odot (Gx^\star - h) = 0. \tag{17}$$

Proof. The saddle-point formulation in Eq. (2) is

$$\min_{x\in X}\max_{y\in Y}\left(c^\top x - y^\top Ax + b^\top y\right). \tag{18}$$

Its first-order optimality conditions are

$$0 \in c - A^\top y^\star + N_X(x^\star), \qquad 0 \in Ax^\star - b + N_Y(y^\star), \tag{19}$$

which are necessary and sufficient because the problem is convex–concave and polyhedral.

For the equality component, $y^{\text{eq}} \in \mathbb{R}^{m_e}$ is unrestricted, and therefore the corresponding normal cone is zero. Hence $Px^\star = q$. For the inequality component,

$$0 \in Gx^\star - h + N_{\mathbb{R}^{m_i}_+}(y^{\text{in},\star}), \tag{20}$$

which is equivalent to primal feasibility, dual feasibility, and complementary slackness. □

Lemma B.2 (Equivalence between LP solutions and PDHG fixed points)**.** *The primal–dual optimal set coincides with the fixed-point set of the PDHG operator:*

$$Z^\star = \mathrm{Fix}(\mathcal{H}) := \{z \in X \times Y : \mathcal{H}(z) = z\}. \tag{21}$$

Furthermore,

$$\mathrm{Fix}(\mathcal{R}_\alpha) = \mathrm{Fix}(\mathcal{H}) = Z^\star. \tag{22}$$

Proof. Let

$$z = \mathcal{H}(\hat{z}) = (x, y). \tag{23}$$

Eq. (4) gives

$$x = \mathrm{proj}_X\left[\hat{x} - \frac{\eta}{\omega}(c - A^\top \hat{y})\right], \tag{24}$$

$$y = \mathrm{proj}_Y\left[\hat{y} + \omega\eta\,(b - A(2x - \hat{x}))\right]. \tag{25}$$

Using the projection characterization

$$v = \mathrm{proj}_C(a) \quad \Longleftrightarrow \quad a - v \in N_C(v), \tag{26}$$

Eq. (24) and Eq. (25) imply

$$\frac{\omega}{\eta}(\hat{x} - x) + A^\top \hat{y} - c \in N_X(x), \tag{27}$$

$$\frac{1}{\omega\eta}(\hat{y} - y) + b - A(2x - \hat{x}) \in N_Y(y). \tag{28}$$

Suppose first that $\hat{z} = \mathcal{H}(\hat{z})$. Then $x = \hat{x}$ and $y = \hat{y}$, so

$$A^\top \hat{y} - c \in N_X(\hat{x}), \qquad b - A\hat{x} \in N_Y(\hat{y}). \tag{29}$$

Equivalently,

$$0 \in c - A^\top \hat{y} + N_X(\hat{x}), \qquad 0 \in A\hat{x} - b + N_Y(\hat{y}). \tag{30}$$

By Lemma B.1, $\hat{z} \in Z^\star$.

Conversely, let $\hat{z} \in Z^\star$. Then the KKT conditions imply

$$A^\top \hat{y} - c \in N_X(\hat{x}), \qquad b - A\hat{x} \in N_Y(\hat{y}). \tag{31}$$

The projection characterization gives

$$\hat{x} = \mathrm{proj}_X\left[\hat{x} - \frac{\eta}{\omega}(c - A^\top \hat{y})\right] \tag{32}$$

and

$$\hat{y} = \mathrm{proj}_Y\left[\hat{y} + \omega\eta(b - A\hat{x})\right]. \tag{33}$$

Since the first equation gives $x = \hat{x}$, the second equation agrees with Eq. (25). Hence $\mathcal{H}(\hat{z}) = \hat{z}$.

Finally,

$$\hat{z} - \mathcal{R}_\alpha(\hat{z}) = (1 + \alpha)\big(\hat{z} - \mathcal{H}(\hat{z})\big), \tag{34}$$

and therefore $\mathrm{Fix}(\mathcal{R}_\alpha) = \mathrm{Fix}(\mathcal{H})$. □

Lemma B.3 (Metric and nonexpansiveness)**.** *For fixed $\omega, \eta > 0$, define*

$$Q(\omega, \eta) := \begin{pmatrix} \frac{\omega}{\eta} I_n & A^\top \\ A & \frac{1}{\omega\eta} I_m \end{pmatrix}. \tag{35}$$

Then $Q \succ 0$ if and only if $\eta\|A\|_2 < 1$. Under this condition, $\mathcal{H} = \mathcal{H}(\cdot; \omega, \eta)$ is firmly nonexpansive in the norm $\|v\|_Q := \sqrt{v^\top Q v}$,

$$\|\mathcal{H}(\hat{z}) - \mathcal{H}(\hat{z}')\|_Q^2 \le (\mathcal{H}(\hat{z}) - \mathcal{H}(\hat{z}'))^\top Q(\hat{z} - \hat{z}'). \tag{36}$$

Consequently, it is nonexpansive in this norm, and the fixed-point residual

$$M^{\mathrm{fp}}(\hat{z}; \omega, \eta) := \|\mathcal{H}(\hat{z}; \omega, \eta) - \hat{z}\|_Q \tag{37}$$

vanishes if and only if $\hat{z} \in Z^\star$.

Proof. The Schur complement of the primal diagonal block in Eq. (35) gives

$$Q \succ 0 \iff I_m - \eta^2 A A^\top \succ 0 \iff \eta\|A\|_2 < 1. \tag{38}$$

Fix parameters satisfying this condition. For $z = \mathcal{H}(\hat{z}; \omega, \eta)$, adding the two projection inequalities gives, for every $z' \in X \times Y$,

$$\begin{aligned}(\hat{z} - z)^\top Q(z' - z) \le \left(c - A^\top y\right)^\top (x' - x) \\ + (Ax - b)^\top (y' - y).\end{aligned} \tag{39}$$

Apply this inequality to outputs $z = \mathcal{H}(\hat{z})$ and $z' = \mathcal{H}(\hat{z}')$, interchange the primed and unprimed variables, and add. The coupling terms cancel, yielding

$$\|z - z'\|_Q^2 \le (z - z')^\top Q(\hat{z} - \hat{z}'). \tag{40}$$

This proves firm nonexpansiveness (Lu & Yang, 2024). Applying Cauchy–Schwarz in the Q-inner product gives $\|z - z'\|_Q \le \|\hat{z} - \hat{z}'\|_Q$. Positive definiteness gives $M^{\mathrm{fp}}(\hat{z}; \omega, \eta) = 0$ if and only if $\mathcal{H}(\hat{z}) = \hat{z}$. Lemma B.2 then gives $\hat{z} \in Z^\star$. □

B.2 Proof of Proposition 3.1

Proof of Proposition 3.1. By convergence, write $\hat{z}_t \to \bar{\hat{z}}$ and $z_t \to \bar{z}$. Then

$$\|z_t - z_{t-1}\|_2 \longrightarrow 0, \qquad \|\hat{z}_{t+1} - \hat{z}_t\|_2 \longrightarrow 0. \tag{41}$$

Since $\boldsymbol{\beta}_t$ is bounded, choose $B_\beta < \infty$ such that $\|\boldsymbol{\beta}_t\|_2 \le B_\beta$ for all t. The fixed, finite history length gives, for sufficiently large t,

$$\|\mathbf{K}_t \boldsymbol{\beta}_t\|_2 \le B_\beta \left(\sum_{j=0}^{k-1} \|z_{t-j} - z_{t-j-1}\|_2^2 \right)^{1/2} \longrightarrow 0. \tag{42}$$

For $k = 0$, the history correction and the sum are zero. At a non-restart step, rearranging Eq. (5a) gives

$$\begin{pmatrix} (1 + \alpha_t^x)(x_t - \hat{x}_t) \\ (1 + \alpha_t^y)(y_t - \hat{y}_t) \end{pmatrix} = \hat{z}_{t+1} - \hat{z}_t - \mathbf{K}_t \boldsymbol{\beta}_t - \zeta_t (z_\star - z_t). \tag{43}$$

Set $C_\delta := \max\{1, \delta^{-1}\}$. The lower bound $1 + \alpha_t^x, 1 + \alpha_t^y \ge \delta$ on nonrestart steps, the triangle inequality, and the vanishing anchor correction assumed in Proposition 3.1 give

$$\|z_t - \hat{z}_t\|_2 \le C_\delta \left(\|\hat{z}_{t+1} - \hat{z}_t\|_2 + \|\mathbf{K}_t \boldsymbol{\beta}_t\|_2 + \|\zeta_t (z_\star - z_t)\|_2 \right) \longrightarrow 0. \tag{44}$$

At a restart, $\hat{z}_{t+1} = z_t$, so the same bound holds because $C_\delta \ge 1$. Thus $\bar{\hat{z}} = \bar{z}$. Since η_t and ω_t are uniformly positive and bounded, there exist constants satisfying

$$0 < \underline{\eta} \le \eta_t \le \overline{\eta} < \infty, \qquad 0 < \underline{\omega} \le \omega_t \le \overline{\omega} < \infty. \tag{45}$$

Hence a subsequence $t_j \to \infty$ satisfies

$$(\omega_{t_j}, \eta_{t_j}) \longrightarrow (\omega_\infty, \eta_\infty) \in [\underline{\omega}, \overline{\omega}] \times [\underline{\eta}, \overline{\eta}]. \tag{46}$$

Euclidean projections onto the nonempty closed convex sets X and Y are continuous, so $\mathcal{H}$ is jointly continuous in its input and positive parameters. Taking limits in Eq. (5b) along this subsequence gives

$$\bar{z} = \lim_{j \to \infty} z_{t_j + 1} = \lim_{j \to \infty} \mathcal{H}\left(\hat{z}_{t_j+1}; \omega_{t_j}, \eta_{t_j}\right) = \mathcal{H}(\bar{z}; \omega_\infty, \eta_\infty). \tag{47}$$

Thus $\bar{z}$ is a fixed point of $\mathcal{H}(\cdot; \omega_\infty, \eta_\infty)$. By Lemma B.2, $\bar{z} \in Z^\star$, completing the proof. □

B.3 SATISFYING THE ASSUMPTIONS IN IMPLEMENTATION

We relate the deterministic inference rule in Appendix F.1 to the assumptions of Proposition 3.1. The trained network parameters are fixed during each solve. The implementation enforces the coefficient bounds and makes the anchor correction vanish whenever the output sequence converges. Convergence of both $\{\hat{z}_t\}$ and $\{z_t\}$ remains a premise of the proposition.

Bounded history coefficients and positive relaxation. Each continuous actor has a final tanh hidden layer followed by a linear output layer. For any scalar mean output, write

$$\mu(o) = w_\mu^\top h_\mu(o) + b_\mu, \qquad h_\mu(o) \in [-1,1]^{64}, \qquad |\mu(o)| \le \|w_\mu\|_1 + |b_\mu|. \tag{48}$$

The right-hand side is a finite constant for the trained network, independent of the observation and iteration. At inference, $\boldsymbol{\beta}_t = \boldsymbol{\mu}_t^\beta$, so the fixed number k of history coefficients is uniformly bounded. The exponential transforms give $\alpha_t^x, \alpha_t^y > 0$, satisfying the proposition's relaxation lower bound with $\delta = 1$, and a bounded positive anchor multiplier $\widetilde{\zeta}_t = \exp\left(\mu_t^\zeta\right) \le C_\zeta$ for some $C_\zeta < \infty$.

Step-size and primal-weight bounds. Let $a_A > 0$ denote the fixed spectral-norm denominator, including the numerical safeguard, for the matrix supplied to PDHG. Let $L_\omega > 0$ be the configured primal-weight range in Eq. (93). The sigmoid and tanh transforms give

$$\frac{0.1}{a_A} \le \eta_t \le \frac{5}{a_A}, \qquad \omega_0 e^{-L_\omega} \le \omega_t \le \omega_0 e^{L_\omega}. \tag{49}$$

Since a_A and ω_0 are positive and finite for each LP, both parameters are uniformly positive and bounded. The weight is set relative to ω_0 at every iteration, so these bounds do not accumulate over successive updates. Proposition 3.1 does not require $\eta_t \|A\|_2 < 1$.

Vanishing normalized anchor correction. The implemented coefficient is

$$\zeta_t = \begin{cases} \widetilde{\zeta}_t \dfrac{\|z_t - z_{t-1}\|_2}{\|z_\star - z_t\|_2}, & z_\star \neq z_t, \\ 0, & z_\star = z_t. \end{cases} \tag{50}$$

The zero case is handled explicitly before division. For either case, if $z_t \to \bar{z}$, then

$$\|\zeta_t (z_\star - z_t)\|_2 \le C_\zeta \|z_t - z_{t-1}\|_2 \longrightarrow 0. \tag{51}$$

Thus the anchor correction vanishes as required by Proposition 3.1, independently of the restart pattern. Only the anchor correction must vanish; the effective coefficient ζ_t need not itself be bounded.

Alternative Halpern-style parameterization. Using the epoch counter s from Section 3.1, an alternative parameterization is

$$\zeta_t = \min\left\{1, \frac{\widetilde{\zeta}_t}{s+1}\right\}, \qquad 0 \le \zeta_t \le \min\left\{1, \frac{C_\zeta}{s+1}\right\}. \tag{52}$$

The associated relaxation coefficients

$$\alpha_t^x = \exp\left(u_t^{\alpha^x}\right)(1-\zeta_t), \qquad \alpha_t^y = \exp\left(u_t^{\alpha^y}\right)(1-\zeta_t) \tag{53}$$

are nonnegative, satisfying the proposition's relaxation lower bound with $\delta = 1$. Suppose $z_t \to \bar{z}$. If only finitely many restarts occur, then $z_\star$ is eventually fixed and $s \to \infty$, so

$$\|\zeta_t (z_\star - z_t)\|_2 \le \frac{C_\zeta}{s+1} \|z_\star - z_t\|_2 \longrightarrow 0. \tag{54}$$

If infinitely many restarts occur, the iteration at which the current anchor was set tends to infinity. Since each new anchor is an output iterate, $z_\star \to \bar{z}$. Using $0 \le \zeta_t \le 1$ gives

$$\|\zeta_t (z_\star - z_t)\|_2 \le \|z_\star - z_t\|_2 \longrightarrow 0. \tag{55}$$

Thus the anchor correction vanishes as required by Proposition 3.1 for either restart pattern.

During training, Gaussian exploration has unbounded support, so the uniform bounds on $\boldsymbol{\beta}_t$ and $\widetilde{\zeta}_t$ established here do not extend automatically to an infinite sampled trajectory. Proposition 3.1 applies to any trajectory satisfying its hypotheses, whether its controls are deterministic or random.

B.4 COEFFICIENT MAPPINGS FOR CLASSICAL ACCELERATION

Fix $\omega, \eta > 0$ and write $\mathcal{H} = \mathcal{H}(\cdot; \omega, \eta)$. Within an uninterrupted run, write $u_s := \hat{z}_t$ and $v_s := z_t = \mathcal{H}(u_s)$ at local iteration s. The following mappings concern the acceleration recurrence with each method's standard initialization, independently of the rollout initialization and restart branch in Algorithm 1.

Halpern and reflected Halpern. Set $u_0 = z_\star$ and $\boldsymbol{\beta}_t = 0$. Substituting the first two rows of Table 1 into Eq. (5a) gives, respectively,

$$\begin{aligned} u_{s+1} &= \frac{s+1}{s+2}\mathcal{H}(u_s) + \frac{1}{s+2}u_0, \\ u_{s+1} &= \frac{s+1}{s+2}(2\mathcal{H}(u_s) - u_s) + \frac{1}{s+2}u_0. \end{aligned} \tag{56}$$

These are Halpern iteration on $\mathcal{H}$ and on its reflection $2\mathcal{H} - I$, respectively (Halpern, 1967; Lu & Yang, 2024).

Krasnosel'skiĭ–Mann. For constant $0 < \lambda < 2$, setting $\alpha_t^x = \alpha_t^y = \lambda - 1$, $\boldsymbol{\beta}_t = 0$, and $\zeta_t = 0$ gives

$$u_{s+1} = (1-\lambda)u_s + \lambda\mathcal{H}(u_s). \tag{57}$$

This is the KM relaxation of $\mathcal{H}$, with overrelaxation when $\lambda > 1$ (Boţ & Nguyen, 2023).

Fast KM. For $\nu > 2$, $k \geq 1$, and $s \geq 1$, the final row of Table 1 gives

$$\begin{aligned} u_{s+1} &= v_s + \left(\frac{\nu}{2(s+\nu)} - 1\right)(v_s - u_s) + \frac{s}{s+\nu}(v_s - v_{s-1}) \\ &= \left(1 - \frac{\nu}{2(s+\nu)}\right)u_s + \frac{\nu}{2(s+\nu)}\mathcal{H}(u_s) + \frac{s}{s+\nu}(\mathcal{H}(u_s) - \mathcal{H}(u_{s-1})). \end{aligned} \tag{58}$$

This is equation (2.8) of Boţ & Nguyen (2023), with their damping parameter α replaced by ν and their step parameter set to one. Initialize u_0, u_1 and store $v_0 = \mathcal{H}(u_0)$ and $v_1 = \mathcal{H}(u_1)$. The general Fast KM recurrence with a nonunit step also contains an input-difference term, so the table specifies the unit-step variant.

Anderson coefficients. The vector $\boldsymbol{\beta}_t^{\mathrm{AA}} \in \mathbb{R}^k$ specifies how much of each past output difference to add to the current PDHG output z_t. Anderson acceleration chooses these coefficients by using past changes in the fixed-point residual to approximately cancel the current residual. For the PDHG input–output pairs $(\hat{z}_i, z_i)$, where $z_i = \mathcal{H}(\hat{z}_i)$, define

$$d_t := z_t - \hat{z}_t, \qquad \mathbf{F}_t := [d_t - d_{t-1} \quad \cdots \quad d_{t-k+1} - d_{t-k}]. \tag{59}$$

The residual d_t vanishes exactly when $\hat{z}_t$ is a fixed point of $\mathcal{H}$. The columns of $\mathbf{F}_t$ record residual changes over the same history as the output differences in $\mathbf{K}_t$. For trial coefficients $b \in \mathbb{R}^k$, the residual model $d_t + \mathbf{F}_t b$ combines the current residual with these past changes. Undamped type-II Anderson acceleration chooses b so that $\mathbf{F}_t b$ cancels d_t as closely as possible in the weighted norm $\|v\|_Q^2 = v^\top Q v$. Here, $Q \succ 0$ is the metric in Eq. (35), with $\eta\|A\|_2 < 1$. Using the available history columns and fixing all unavailable coefficients to zero, this gives

$$\boldsymbol{\beta}_t^{\mathrm{AA}} \in \operatorname*{arg\,min}_{b \in \mathbb{R}^k} \|d_t + \mathbf{F}_t b\|_Q^2. \tag{60}$$

The optimization variables are the history coefficients, and the objective is the squared norm of this residual model, not the exact residual at the next iterate. Apply the fitted coefficient of each residual difference to the matching output difference to form the next PDHG input,

$$\hat{z}_{t+1} = z_t + \mathbf{K}_t \boldsymbol{\beta}_t^{\mathrm{AA}}. \tag{61}$$

This is the undamped Anderson update in difference form (Anderson, 1965), recovered from Eq. (5a) by setting $\alpha_t^x = \alpha_t^y = \zeta_t = 0$. For a full window and $b = \boldsymbol{\beta}_t^{\mathrm{AA}}$, both the residual model and the update use the affine weights $1 + b_1$, $b_{j+1} - b_j$ for $1 \leq j < k$, and $-b_k$ on the current and past residuals or outputs, respectively. These weights sum to one.

Table 1 describes the algebraic GA-PDHG family. The exponential parameterization in Appendix F.1 restricts the learned extrapolation coefficients to positive values, so it does not implement the negative coefficients of underrelaxed KM or the displayed Fast KM specialization.

C Details on PDHG-based LP solver enhancements

This appendix gives the enhancement rules summarized in Section 2.2. Table 8 compares these enhancements across representative solvers. The four subsections follow the table columns.

Table 8: Representative PDHG-based LP solver papers and their enhancement techniques.

Paper / solver	Step size	Primal weight	Restarting criteria	Acceleration method
PDLP (Applegate et al., 2021)	Adaptive	EMA	Normalized duality gap	Averaging
raPDHG (Applegate et al., 2023)	Constant	Constant	Normalized duality gap	Averaging
cuPDLP-C (Lu et al., 2023)	Adaptive	EMA	KKT error	Averaging
cuPDLP.jl (Lu & Yang, 2025)	Adaptive	EMA	KKT error	Averaging
rHPDHG / r^{2}HPDHG (Lu & Yang, 2024)	Adaptive	EMA	Fixed-point residual	Halpern / Reflected Halpern
MPAX (LP) (Lu et al., 2024)	Adaptive	EMA	Fixed-point residual	Reflected Halpern
cuPDLPx (Lu et al., 2025)	Constant	PID	Fixed-point residual	Reflected Halpern
AA-PDHG (Zhou et al., 2025)	Constant	†EMA	N/A	Anderson

EMA denotes exponential averaging in log space. PID denotes PID controller. N/A denotes no epoch restart. Paired acceleration entries follow the order of the variants. †AA-PDHG uses periodic EMA and resets the Anderson history at weight updates.

C.1 Step size

raPDHG, cuPDLPx, and AA-PDHG use a constant step size $\eta_t \equiv \eta$. The condition $\eta\|A\|_2 < 1$ ensures nonexpansiveness of the PDHG map in the induced norm (Appendix B.1). Choosing a constant step size from this condition can be conservative (Applegate et al., 2021), because it must hold for all possible updates, even when the current update permits a larger step. PDLP, cuPDLP.jl/C, MPAX, and rHPDHG/r^{2}HPDHG instead choose the step size η_t adaptively from trial updates. Concretely, for a trial update $z_{t+1} = \mathcal{H}(\hat{z}_{t+1}; \omega_t, \eta_t)$, the step size acceptance test verifies

$$\eta_t \le \frac{\|z_{t+1} - \hat{z}_{t+1}\|_{\omega_t}^2}{2\left|(y_{t+1} - \hat{y}_{t+1})^\top A\,(x_{t+1} - \hat{x}_{t+1})\right|}, \quad \text{where} \quad \|z\|_\omega^2 := \omega\|x\|_2^2 + \omega^{-1}\|y\|_2^2. \tag{62}$$

If the test fails, it reduces η_t and recomputes z_{t+1} until the condition holds. Checking only the current trial can permit steps beyond the sufficient spectral-norm bound. Rejected trials incur extra matrix–vector products. Although the test is motivated by the nonexpansiveness proof of the PDHG map, to the best of our knowledge, no convergence guarantee has been established for the adaptive step-size rule used in practical PDLP-style solvers (Applegate et al., 2021; Lu & Yang, 2024).

C.2 Primal weight

The primal weight adjusts the relative primal and dual step sizes to balance the scales of the primal and dual variables. The convergence of standard PDHG is proven for a constant $\omega_t \equiv \omega > 0$ and a constant step size $\eta < 1/\|A\|_2$ (Chambolle & Pock, 2011; Applegate et al., 2023). Adaptive primal-weight updates are not covered by this fixed-weight convergence guarantee. In the hand-designed adaptive schemes reviewed here, the primal weight is updated at the end of each block of iterations, with block boundaries set by restarts or fixed update intervals. It remains constant within each block. Superscript j indexes blocks, subscript t indexes individual iterations, and $\omega^j > 0$ is the weight in block j. Let (x^j, y^j) denote the starting point of block j. The displacement norms between successive block starting points are

$$d_x^j := \left\|x^{j+1} - x^j\right\|_2, \qquad d_y^j := \left\|y^{j+1} - y^j\right\|_2. \tag{63}$$

For d_x^j, d_y^j, define the log-balance error

$$b^j := \log\left(\frac{\omega^j d_x^j}{d_y^j}\right). \tag{64}$$

PDLP, cuPDLP-C/jl, rHPDHG/r^{2}HPDHG, MPAX, and AA-PDHG use exponential moving averaging (EMA) in log space, whereas cuPDLPx uses proportional–integral–derivative (PID) control. At the end of block j, the weight for the next block is computed as

$$\log\omega^{j+1} = \log\omega^j - \begin{cases} \theta_\omega b^j, & \text{EMA}, \\ K_p b^j + K_i \sum_{\ell=0}^{j} b^\ell + K_d\left(b^j - b^{j-1}\right), & \text{PID}, \end{cases} \tag{65}$$

where $\theta_\omega \in [0,1]$ is the smoothing coefficient, and K_p, K_i, K_d are PID gains. Both rules implement feedback control to balance the weighted displacements $\sqrt{\omega^j}d_x^j$ and $d_y^j/\sqrt{\omega^j}$, corresponding to $b^j = 0$. EMA changes $\log\omega^j$ by $-\theta_\omega b^j$, so the correction is proportional to the current error. PID control uses past errors through its integral and derivative terms (Lu et al., 2025).

C.3 Restarting

Restarting reinitializes PDHG from a selected point $z_\star$ and resets its past averaging or acceleration state. When restarting is combined with averaging or Halpern acceleration, suitable parameter choices and restart rules exploit LP sharpness to achieve accelerated linear convergence (Applegate et al., 2023; Lu & Yang, 2024). Restart timing affects both the theoretical iteration complexity and the practical convergence of restarted PDHG. A fixed-frequency scheme restarts after a prescribed number of iterations. In practice, the restarted solvers in Table 8 use adaptive restart heuristics based on the conditions

$$M_t \le \beta^{\text{suf}} M_\star \qquad \text{(sufficient progress)}, \tag{66a}$$

$$M_t \le \beta^{\text{nec}} M_\star \quad \text{and} \quad M_t > M_{t-1} \qquad \text{(loss of progress)}, \tag{66b}$$

$$s + 1 \ge \beta^{\text{art}} t \qquad \text{(long iterations after restart)}, \tag{66c}$$

Here, $M_t := M(z_t)$ is a restart metric evaluated at iterate z_t, and $M_\star := M(z_\star)$ is its value at the restart anchor $z_\star$. In Eq. (66c), s is the zero-based index within the current epoch, so $s+1$ counts its completed PDHG steps, while $t \ge 1$ counts all completed PDHG steps. These conditions are checked at a prescribed frequency using thresholds satisfying $0 < \beta^{\text{suf}} < \beta^{\text{nec}} < 1$ and $\beta^{\text{art}} \in (0,1)$. The metric M, manually tuned constant thresholds (β^{suf}, β^{nec}, β^{art}), and the condition-checking intervals differ across solvers, as do initialization rules such as more frequent checks during the first few iterations.

C.4 Acceleration

An acceleration step constructs the next PDHG input or a restart candidate from the current PDHG update and past iterates. The solvers in Table 8 use *Averaging*, *Halpern*, or *Anderson* accelerations.

Averaging. Averaging uses the step-size weighted average

$$\bar{z}_t = \frac{\sum_{i=t-s}^{t} \eta_{i-1} z_i}{\sum_{i=t-s}^{t} \eta_{i-1}}$$

as the restart candidate. When primal–dual iterates oscillate around a solution, averaging can cancel deviations in opposite directions. Restarted averaging with a fixed step size η and primal weight ω exploits LP sharpness to achieve accelerated linear convergence under the prescribed restart rules based on the normalized duality gap or KKT error (Applegate et al., 2023; Lu & Yang, 2025).

Halpern acceleration. Halpern acceleration combines the current PDHG input $\hat{z}_t$, output z_t, and fixed restart anchor $z_\star$ to form the next PDHG input

$$\hat{z}_{t+1} = z_t + \frac{\alpha(s+1)}{s+2}\left(z_t - \hat{z}_t\right) + \frac{1}{s+2}\left(z_\star - z_t\right). \tag{67}$$

Here, α is the reflection coefficient, with $\alpha = 0$ recovering standard Halpern acceleration and $\alpha = 1$ yielding reflected Halpern acceleration (Lu & Yang, 2024). The anchor coefficient $1/(s+2)$ decreases as s increases and returns to $1/2$ at each restart, when s is reset to zero. For feasible and bounded LPs, both variants achieve an $O(1/(s+1))$ fixed-point residual bound with fixed $\eta, \omega > 0$ satisfying $\eta\|A\|_2 < 1$. When combined with the restart schemes analyzed by Lu & Yang (2024), they achieve accelerated linear convergence.

Anderson acceleration. Anderson acceleration corrects the current PDHG output using recent output differences (Zhou et al., 2025). Using the k most recent PDHG output differences, Anderson acceleration takes the form

$$\hat{z}_{t+1} = z_t + \mathbf{K}_t \boldsymbol{\beta}_t^{\mathrm{AA}}, \qquad \mathbf{K}_t := [z_t - z_{t-1} \quad \cdots \quad z_{t-k+1} - z_{t-k}]. \tag{68}$$

The coefficients $\boldsymbol{\beta}_t^{\mathrm{AA}} \in \mathbb{R}^k$ are computed from recent PDHG input–output pairs $(\hat{z}_i, z_i)$. A small least-squares problem selects coefficients that combine past residual changes to approximately cancel the current residual $z_t - \hat{z}_t$. The same coefficients are then applied to the output differences to construct the next PDHG input. The resulting history correction has a quasi-Newton interpretation and can reduce error components that decay slowly under basic PDHG.

These acceleration methods form linear combinations of the current and past iterates. Averaging uses step-size weights, Halpern acceleration uses coefficients prescribed by convergence analysis, and Anderson acceleration computes coefficients by solving a least-squares problem.

D Further Details on Problem Classes

We describe the transportation, auction, minimum-cost flow, robust inventory management, and L_∞ polynomial regression generators used in Section 4.1. The transportation, auction, and minimum-cost flow generators construct primal–dual solutions satisfying the KKT conditions, so their optimal values are known without an auxiliary solve. The inventory generator solves each generated LP, while the regression generator also refines the resulting active set, as detailed below. Nonzero counts refer to the stacked constraint matrix A defined after Eq. (1), with variable bounds stored separately in X. Here $\mathrm{nnz}(A)$ counts the nonzero entries, and matrix density is $\mathrm{nnz}(A)/(mn)$, the complement of the zero fraction. Symbols for problem-specific data in the following subsections are local to each LP class.

D.1 Transport

Balanced transportation allocates mass from s sources to t destinations at minimum cost. In this subsection only, s, t are problem dimensions. Let $a \in \mathbb{R}^s_+$ and $d \in \mathbb{R}^t_+$ be the supply and demand vectors with $\sum_{i=1}^s a_i = \sum_{j=1}^t d_j$, and let $c_{ij} \in \mathbb{R}$ be the unit cost from source i to destination j. For transported amounts $x_{ij} \geq 0$, the LP is

$$\begin{aligned} \min_{x_{ij} \geq 0} \quad & \sum_{i=1}^{s} \sum_{j=1}^{t} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{t} x_{ij} = a_i, \qquad i = 1, \ldots, s, \\ & \sum_{i=1}^{s} x_{ij} = d_j, \qquad j = 1, \ldots, t. \end{aligned} \tag{69}$$

The generator obtains the marginals from a feasible plan supported on a spanning tree of the complete bipartite graph, optionally with additional support arcs. After vectorization, the LP has $n = st$ variables, $m_e = m = s + t$ equality constraints, and no inequality rows. Each column of $A = P$ contains two unit entries, one for its source and one for its destination, giving

$$\mathrm{nnz}(A) = 2st = 2n, \qquad \frac{\mathrm{nnz}(A)}{mn} = \frac{2}{s+t}. \tag{70}$$

All st transport arcs remain in the LP, so changing the optimal plan's support or introducing cost ties does not change matrix sparsity. One mass-balance equality is redundant and is retained.

D.2 AUCTION

We use a capacitated LP relaxation of combinatorial auction winner determination, where bids request bundles of items. For m items and n bids, let $B \in \{0,1\}^{m\times n}$ have $B_{ij} = 1$ when bid j includes item i, let $v \in \mathbb{R}^n$ contain bid values, and let $d \in \mathbb{R}^m_+$ contain item capacities. The variable $x_j \in [0,1]$ is the accepted fraction of bid j, and the LP is

$$\max_{x\in\mathbb{R}^n} \quad v^\top x \quad \text{subject to} \quad Bx \le d, \qquad 0 \le x \le \mathbf{1}. \tag{71}$$

In Eq. (1), this corresponds to $c = -v$, $G = -B$, and $h = -d$, with $m_e = 0$ and $m_i = m$. The generated capacities are not restricted to one. Each bundle contains $\kappa \in \{1,\dots,m\}$ distinct items sampled without replacement, so

$$\operatorname{nnz}(A) = \kappa n, \qquad \frac{\operatorname{nnz}(A)}{mn} = \frac{\kappa}{m}. \tag{72}$$

Bundle size determines column sparsity, while the item-popularity parameter changes row degrees and overlap among bids without changing the nonzero count. The finite bounds allow both full and partial bid acceptance in the relaxation.

D.3 MIN-COST FLOW

Capacitated minimum-cost flow routes a commodity through a directed network while satisfying node balances and arc capacities. Let V and E be the node and arc sets, with $m = |V|$ and $n = |E|$, and define the incidence matrix $B \in \mathbb{R}^{m\times n}$ to have -1 at the tail and $+1$ at the head of each arc. For arc costs $c \in \mathbb{R}^n$, capacities $u \in \mathbb{R}^n_{>0}$, and net inflows $d \in \mathbb{R}^m$ satisfying $\mathbf{1}^\top d = 0$, the flow vector $x \in \mathbb{R}^n$ solves

$$\min_{x\in\mathbb{R}^n} \quad c^\top x \quad \text{subject to} \quad Bx = d, \qquad 0 \le x \le u. \tag{73}$$

The generator randomly orients a spanning tree and adds distinct directed arcs without self-loops, ensuring that the underlying undirected graph is connected. Here $A = P = B$, $m_e = m$, and $m_i = 0$, and each column contains exactly two nonzeros, giving

$$\operatorname{nnz}(A) = 2n, \qquad \frac{\operatorname{nnz}(A)}{mn} = \frac{2}{m}. \tag{74}$$

Adding arcs increases row degrees and the total nonzero count, while column sparsity remains fixed. The incidence matrix has rank $m-1$, and the redundant conservation row is retained. Capacities affect which box constraints are active without changing matrix sparsity.

D.4 ROBUST INVENTORY MANAGEMENT

We consider N items over T periods with uncertain demand and shared purchasing and storage limits. For item $i = 1,\dots,N$ and period $t = 1,\dots,T$, let $x_t^{(i)}$, $u_t^{(i)}$, and $b_t^{(i)}$ denote inventory, purchases, and a nonnegative shortage buffer. The initial inventory $x_1^{(i)}$ is given. The nonnegative parameters $h_t^{(i)}$, $c_t^{(i)}$, $p_t^{(i)}$, $s_t^{(i)}$, and $v^{(i)}$ are the holding cost, purchase cost, selling price, shortage penalty, and unit volume. Let C and V be the per-period capital budget and storage capacity. Demand lies in a box, and purchases follow an itemwise causal affine policy,

$$\mathcal{D} = \prod_{i=1}^{N}\prod_{t=1}^{T}\left[\underline{d}_t^{(i)}, \bar{d}_t^{(i)}\right], \qquad u_t^{(i)} = \hat{u}_t^{(i)} + \sum_{\tau<t} k_{t,\tau}^{(i)} d_\tau^{(i)}. \tag{75}$$

The nominal purchases $\hat{u}_t^{(i)}$, feedback gains $k_{t,\tau}^{(i)}$, and buffers $b_t^{(i)}$ are chosen before demand is observed. Only purchases adapt to past demand, with no feedback between items. The underlying robust problem is

$$\begin{aligned}
\min_{\hat{u},k,\,b\ge 0}\ \max_{d\in\mathcal{D}} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T}\left[h_t^{(i)}x_t^{(i)} + c_t^{(i)}u_t^{(i)} - p_t^{(i)}d_t^{(i)} + \left(p_t^{(i)} + s_t^{(i)}\right)b_t^{(i)}\right] \\
\text{subject to} \quad & x_{t+1}^{(i)} = x_t^{(i)} + u_t^{(i)} - d_t^{(i)} + b_t^{(i)}, \qquad i = 1,\dots,N, \quad t = 1,\dots,T, \\
& x_t^{(i)} \ge 0, \quad u_t^{(i)} \ge 0, \quad \sum_i v^{(i)}x_t^{(i)} \le V, \quad \sum_i c_t^{(i)}u_t^{(i)} \le C, \qquad \forall d \in \mathcal{D}.
\end{aligned} \tag{76}$$

The last line applies for $t = 1, \ldots, T$ and, for the itemwise inequalities, every i. Holding costs are charged on x_t, the inventory entering period t. The generated problem has no terminal constraint on x_{T+1}. The buffer b_t is a deterministic feasibility slack, rather than the realized positive-part shortage, and has no demand-dependent upper bound.

LP formulation. Define the demand-independent inventory $a_t^{(i)}$ and demand coefficients $\beta_{t,\tau}^{(i)}$ by

$$\begin{aligned} a_t^{(i)} &= x_1^{(i)} + \sum_{r<t} \left(\hat{u}_r^{(i)} + b_r^{(i)} \right), \quad \beta_{t,\tau}^{(i)} = \sum_{r=\tau+1}^{t-1} k_{r,\tau}^{(i)} - 1, \quad \tau < t, \\ x_t^{(i)} &= a_t^{(i)} + \sum_{\tau<t} \beta_{t,\tau}^{(i)} d_\tau^{(i)}. \end{aligned} \tag{77}$$

Empty sums are zero. The total cost is $\Gamma_0 + \sum_{i,\tau} \gamma_\tau^{(i)} d_\tau^{(i)}$, where

$$\begin{aligned} \Gamma_0 &= \sum_{i=1}^{N} \sum_{t=1}^{T} \left[h_t^{(i)} a_t^{(i)} + c_t^{(i)} \hat{u}_t^{(i)} + \left(p_t^{(i)} + s_t^{(i)} \right) b_t^{(i)} \right], \\ \gamma_\tau^{(i)} &= \sum_{t=\tau+1}^{T} \left[h_t^{(i)} \beta_{t,\tau}^{(i)} + c_t^{(i)} k_{t,\tau}^{(i)} \right] - p_\tau^{(i)}. \end{aligned} \tag{78}$$

For each coefficient $q \in \{k_{t,\tau}^{(i)}, \beta_{t,\tau}^{(i)}, \gamma_\tau^{(i)}\}$, introduce $q^+, q^- \geq 0$ with $q = q^+ - q^-$. For complementary splits satisfying $q^+ q^- = 0$, the minimum and maximum of $q d_\tau^{(i)}$ over its interval are, respectively,

$$L_\tau^{(i)}\left(q^\pm\right) = \underline{d}_\tau^{(i)} q^+ - \bar{d}_\tau^{(i)} q^-, \qquad U_\tau^{(i)}\left(q^\pm\right) = \bar{d}_\tau^{(i)} q^+ - \underline{d}_\tau^{(i)} q^-. \tag{79}$$

Let $\mathcal{R}$ be the sum of all split variables,

$$\mathcal{R} = \sum_{i=1}^{N} \left[\sum_{\tau<t} \left(k_{t,\tau}^{(i),+} + k_{t,\tau}^{(i),-} + \beta_{t,\tau}^{(i),+} + \beta_{t,\tau}^{(i),-} \right) + \sum_{\tau=1}^{T} \left(\gamma_\tau^{(i),+} + \gamma_\tau^{(i),-} \right) \right]. \tag{80}$$

With a free epigraph variable η, the generated LP is

$$\begin{aligned} \min_{\eta, \hat{u}, b, k^\pm, \beta^\pm, \gamma^\pm} \quad & \eta + \varepsilon \mathcal{R} \\ \text{subject to} \quad & \Gamma_0 + \sum_{i=1}^{N} \sum_{\tau=1}^{T} U_\tau^{(i)} \left(\gamma_\tau^{(i),\pm} \right) \leq \eta, \\ & \sum_{i=1}^{N} c_t^{(i)} \left[\hat{u}_t^{(i)} + \sum_{\tau<t} U_\tau^{(i)} \left(k_{t,\tau}^{(i),\pm} \right) \right] \leq C, \quad t = 1, \ldots, T, \\ & \sum_{i=1}^{N} v^{(i)} \left[a_t^{(i)} + \sum_{\tau<t} U_\tau^{(i)} \left(\beta_{t,\tau}^{(i),\pm} \right) \right] \leq V, \quad t = 2, \ldots, T, \\ & \hat{u}_t^{(i)} + \sum_{\tau<t} L_\tau^{(i)} \left(k_{t,\tau}^{(i),\pm} \right) \geq 0, \qquad i = 1, \ldots, N, \quad t = 1, \ldots, T, \\ & a_t^{(i)} + \sum_{\tau<t} L_\tau^{(i)} \left(\beta_{t,\tau}^{(i),\pm} \right) \geq 0, \qquad i = 1, \ldots, N, \quad t = 2, \ldots, T, \\ & b,\ k^\pm,\ \beta^\pm,\ \gamma^\pm \geq 0. \end{aligned} \tag{81}$$

Here a_t, Γ_0, k, β, and γ are substituted using their definitions above. The definitions of β and γ impose linear equalities on the split variables. The nominal purchases $\hat{u}$ are unrestricted. Removing the common part of any split pair preserves these equalities and weakens every inequality, so complementary splits suffice without explicit complementarity constraints. Thus $\varepsilon = 0$ gives an exact LP reformulation of Eq. (76). The benchmark uses $\varepsilon = 10^{-6}$ and includes this finite regularization in its objective.

Generated instances. We use $T = 10$, $x_1^{(i)} = 0$, $v^{(i)} = 1$, $c_t^{(i)} = 0.5$, $h_t^{(i)} = 0.3$, $s_t^{(i)} = 0.1$, $C = 0.25N$, and $V = 0.155N$. Selling prices are sampled independently from $\mathcal{U}[1, 2]$. For $\mu_t = 0.25\,[1 + \cos(2\pi t/7)]$, demand bounds are $\underline{d}_t^{(i)} = \max\{0, \mu_t - 0.075\}$ and $\bar{d}_t^{(i)} = \mu_t + 0.075$. The generator checks the constant initial-inventory constraints once and omits their rows. With variable bounds stored separately, the LP has

$$n = 1 + 2NT\,(T+1)\,, \qquad m_e = \frac{NT\,(T+1)}{2}, \qquad m = 2T + N\left[\frac{T\,(T+5)}{2} - 1\right]. \tag{82}$$

At the stated settings, $\mathrm{nnz}\,(A) = 1075N + 1$. Each instance is solved with HiGHS to obtain a primal–dual reference solution.

D.5 L_∞ Polynomial Regression

L_∞ polynomial regression minimizes the largest absolute residual over a finite set of observations. This criterion is also called minimax or Chebyshev regression and differs from least-squares regression, which minimizes the sum of squared residuals. Let $\{(t_i, y_i)\}_{i=1}^N$ be observations with $t_i \in [-1, 1]$, and let $\{\phi_j\}_{j=1}^p$ be a basis for polynomials of degree at most $p-1$. Write $y = (y_1, \ldots, y_N)$ and define the design matrix $\Phi \in \mathbb{R}^{N\times p}$ by $\Phi_{ij} = \phi_j\,(t_i)$. The coefficient vector $\beta = (\beta_1, \ldots, \beta_p) \in \mathbb{R}^p$ solves

$$\min_{\beta\in\mathbb{R}^p} \|\Phi\beta - y\|_\infty = \min_{\beta\in\mathbb{R}^p} \max_{i=1,\ldots,N} \left|\sum_{j=1}^p \beta_j \phi_j\,(t_i) - y_i\right|. \tag{83}$$

Introducing an epigraph variable $\tau \geq 0$ gives the LP, where $\mathbf{1} \in \mathbb{R}^N$ denotes the all-ones vector.

$$\begin{aligned} \min_{\beta\in\mathbb{R}^p,\,\tau\geq 0} \quad & \tau \\ \text{subject to} \quad & -\tau\mathbf{1} \leq \Phi\beta - y \leq \tau\mathbf{1}. \end{aligned} \tag{84}$$

For the convention $Gx \geq h$ in Eq. (1), set $x = (\beta, \tau)$ and

$$G = \begin{bmatrix} \Phi & \mathbf{1} \\ -\Phi & \mathbf{1} \end{bmatrix}, \qquad h = \begin{bmatrix} y \\ -y \end{bmatrix}, \qquad c = \begin{bmatrix} \mathbf{0}_p \\ 1 \end{bmatrix}. \tag{85}$$

The coefficients β are free variables, whereas only τ has the lower bound zero. There are no equality constraints or finite upper bounds. Thus the generated LP has $n = p+1$ variables and $m_i = m = 2N$ inequality rows. Because P is empty, the stacked matrix is $A = G$. For continuously sampled nodes, this matrix is dense almost surely, with $\mathrm{nnz}\,(A) = 2N\,(p+1) = mn$.

Let T_k be the degree-k Chebyshev polynomial of the first kind, defined by $T_k\,(\cos\theta) = \cos(k\theta)$. For an excess degree $r \geq 1$ and noise scale $\sigma \geq 0$, the generator samples a target function in the Chebyshev basis and adds Gaussian observation noise.

$$y_i = \sum_{k=0}^{p+r-1} a_k T_k\,(t_i) + \sigma\epsilon_i, a_k, \epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}\,(0, 1)\,, \tag{86}$$

We use $r = 3$ and $\sigma = 0.1$. The nodes are sampled uniformly on $[-1, 1]$ by default. The generator also supports the Chebyshev nodes $t_i = \cos\left((2i-1)\,\pi/\,(2N)\right)$ and center-clustered nodes $t_i = u_i\,|u_i|$ for $u_i \sim \mathrm{Unif}\,[-1, 1]$.

The fitted basis is either the monomial basis $\phi_j\,(t) = t^{j-1}$ or the Chebyshev basis $\phi_j\,(t) = T_{j-1}\,(t)$. Both bases span the same degree-$(p-1)$ polynomial space, so paired instances with the same nodes and observations have the same optimal residual $\tau^\star$ in exact arithmetic. Their numerical conditioning can differ substantially. The monomial design is a Vandermonde matrix and can become ill-conditioned as the degree increases, while the Chebyshev representation is typically better conditioned on $[-1, 1]$. The basis choice therefore changes the LP representation without changing the regression function class. Unlike the preceding planted generators, we obtain the reference primal–dual solution by solving Eq. (84) with the HiGHS dual-simplex method and refining the active constraints by least squares.

D.6 TRAINING SET CONSTRUCTION

We generate a fixed training set for each family using the sizes and instance counts in Tables 9 and 10. Each instance is generated from its bucket seed plus its within-bucket index, and the same LPs are reused across PPO iterations. The training sets contain 224 transport, 448 auction, 224 min-cost flow, 256 robust inventory, 512 regression, and 384 random LP instances. We train one policy per family, select one checkpoint by held-out evaluation, and compare it on the same four unseen LPs per test size as both baselines. Tables 9 and 10 list the training, validation, and test sizes, whose instances use disjoint seed ranges. Training counts denote LPs per PPO rollout, held-out counts denote LPs per evaluation, and compared counts denote LPs per solver. Size multipliers are relative to the largest training size and measured in n, except for inventory (item count N) and regression (m at fixed $n = 9$). For inventory, $m = 20 + 74N$ and $n = 1 + 220N$ at a fixed horizon of 10 periods.

Table 9: Training, validation, and test sizes for transport, auction, and min-cost flow.

Family	Multiplier	m	n	$\mathrm{nnz}(A)$	LPs per split	
					Train / Held-out	Compared
Transport	0.125 ×	120	3,200	6,400	64 / –	–
	0.25 ×	160	6,400	12,800	64 / 16	4
	0.5 ×	240	12,800	25,600	64 / –	–
	1 ×	320	25,600	51,200	32 / 16	4
	2.5 ×	506	64,000	128,000	– / 8	4
	5 ×	720	128,000	256,000	– / 8	4
	10 ×	1,012	256,000	512,000	– / 4	4
	12 ×	1,112	307,200	614,400	– / –	4
	14 ×	1,200	358,400	716,800	– / –	4
	16 ×	1,280	409,600	819,200	– / –	4
	25 ×	1,600	640,000	1,280,000	– / –	4
	36 ×	1,920	921,600	1,843,200	– / –	4
	64 ×	2,560	1,638,400	3,276,800	– / –	4
	100 ×	3,200	2,560,000	5,120,000	– / –	4
	400 ×	6,400	10,240,000	20,480,000	– / –	4
Auction	0.125 ×	800	3,200	64,000	128 / 16	–
	0.25 ×	1,600	6,400	128,000	128 / 16	–
	0.5 ×	3,200	12,800	256,000	128 / 16	–
	1 ×	6,400	25,600	512,000	64 / 16	4
	2.5 ×	16,000	64,000	1,280,000	– / –	4
	4 ×	25,600	102,400	2,048,000	– / 8	–
	5 ×	32,000	128,000	2,560,000	– / –	4
	10 ×	64,000	256,000	5,120,000	– / –	4
	12 ×	76,800	307,200	6,144,000	– / –	4
	14 ×	89,600	358,400	7,168,000	– / –	4
Min-cost flow	0.125 ×	600	3,600	7,200	64 / –	–
	0.25 ×	1,200	7,200	14,400	64 / 16	–
	0.5 ×	2,400	14,400	28,800	64 / –	–
	1 ×	4,800	28,800	57,600	32 / 16	4
	2.5 ×	12,000	72,000	144,000	– / 8	4
	5 ×	24,000	144,000	288,000	– / 8	4
	10 ×	48,000	288,000	576,000	– / 4	4
	12 ×	57,600	345,600	691,200	– / –	4

Table 10: Training, validation, and test sizes for inventory, L_∞ regression, and random LP.

Family	Multiplier	m	n	$\mathrm{nnz}(A)$	LPs per split	
					Train / Held-out	Compared
Robust inventory	0.25 ×	390	1,101	5,376	64 / –	4
	0.5 ×	760	2,201	10,751	128 / 4	4
	1 ×	1,500	4,401	21,501	64 / 4	4
	1.5 ×	2,240	6,601	32,251	– / 4	4
	2 ×	2,980	8,801	43,001	– / 2	4
	3 ×	4,460	13,201	64,501	– / 2	4
	5 ×	7,420	22,001	107,501	– / 2	4
	8 ×	11,860	35,201	172,001	– / –	4
	10 ×	14,820	44,001	215,001	– / –	4
L_∞ regression	0.5 ×	120	9	1,080	128 / 16	4
	0.67 ×	160	9	1,440	128 / 16	4
	0.83 ×	200	9	1,800	128 / 16	4
	1 ×	240	9	2,160	128 / 16	4
	2 ×	480	9	4,320	– / 16	4
	5 ×	1,200	9	10,800	– / 16	4
	6 ×	1,440	9	12,960	– / –	4
	7 ×	1,680	9	15,120	– / –	4
Random LP	0.2 ×	2,000	4,000	512,000	64 / –	–
	0.4 ×	4,000	8,000	1,024,000	128 / –	4
	0.8 ×	8,000	16,000	2,048,000	128 / 16	–
	1 ×	10,000	20,000	2,560,000	64 / –	4
	2 ×	20,000	40,000	5,120,000	– / 4	4
	3 ×	30,000	60,000	7,680,000	– / –	4
	4 ×	40,000	80,000	10,240,000	– / –	4
	5 ×	50,000	100,000	12,800,000	– / 4	–
	6 ×	60,000	120,000	15,360,000	– / –	4
	10 ×	100,000	200,000	25,600,000	– / 2	4
	15 ×	150,000	300,000	38,400,000	– / –	4

E FULL RESULTS ON SIX PROBLEM CLASSES

Tables 11 and 12 report the complete size-generalization results for all six LP families. Each entry is the median over four unseen LPs at the indicated size, evaluated by all three solvers. Iterations and matrix–vector products are counted to the first iterate satisfying the common relative KKT tolerance $\epsilon_{\text{tol}} = 10^{-6}$. Algorithm wall clock measures execution after compilation and excludes JIT compilation and preconditioning, as in Section 4.1. The dagger marks the largest training size, and a dash denotes failure to reach the tolerance within the iteration limit.

Table 11: Full test results for transport, auction, and min-cost flow (four-LP medians).

		Iterations to ϵ_{tol}			Matrix–vector products			Algorithm wall clock (ms)		
Family	(m, n)	Ours	HT	MPAX	Ours	HT	MPAX	Ours	HT	MPAX
Transport	(160, 6,400)	**146**	488	550	**294**	979	1,158	**46**	119	76
	(320, 25,600)†	**164**	633	548	**329**	1,268	1,150	**39**	107	87
	(506, 64,000)	**155**	674	540	**312**	1,349	1,134	**46**	143	96
	(720, 128,000)	**150**	628	526	**302**	1,257	1,108	**48**	146	122
	(1,012, 256,000)	**140**	554	518	**282**	1,111	1,090	**52**	154	156
	(1,112, 307,200)	**158**	628	528	**317**	1,259	1,110	**57**	173	170
	(1,200, 358,400)	**152**	621	527	**307**	1,244	1,108	**61**	192	188
	(1,280, 409,600)	**148**	712	524	**299**	1,427	1,101	**66**	247	203
	(1,600, 640,000)	**153**	533	536	**308**	1,068	1,128	**102**	286	308
	(1,920, 921,600)	**152**	594	538	**305**	1,190	1,130	**135**	447	445
	(2,560, 1,638,400)	**149**	676	527	**300**	1,354	1,110	**248**	965	830
	(3,200, 2,560,000)	**149**	608	532	**300**	1,217	1,120	**385**	1,399	1,445
	(6,400, 10,240,000)	**200**	570	524	**403**	1,142	1,105	**1,998**	5,222	7,334
Auction	(6,400, 25,600)†	**190**	928	980	**381**	1,859	2,025	**70**	256	198
	(16,000, 64,000)	**190**	924	1,108	**382**	1,851	2,284	**81**	293	324
	(32,000, 128,000)	**190**	923	994	**382**	1,848	2,055	**84**	316	479
	(64,000, 256,000)	**192**	918	1,044	**387**	1,838	2,158	**114**	433	895
	(76,800, 307,200)	**192**	924	1,046	**387**	1,849	2,162	**118**	453	1,143
	(89,600, 358,400)	**192**	926	1,066	**387**	1,853	2,198	**134**	521	1,352
Min-cost flow	(4,800, 28,800)†	**164**	302	440	**329**	606	928	**49**	68	76
	(12,000, 72,000)	**180**	319	448	**363**	640	944	**63**	85	88
	(24,000, 144,000)	**234**	356	444	**469**	715	939	**80**	94	109
	(48,000, 288,000)	**224**	374	446	**451**	749	938	**91**	120	139
	(57,600, 345,600)	**242**	370	482	**485**	741	1,010	**115**	142	147

Ours: GALLOP; HT: GA-PDHG (HT). Bold: best per metric. †: largest training size. A dash denotes failure to reach $\epsilon_{\text{tol}} = 10^{-6}$ within the iteration limit.

Table 12: Full test results for inventory, L_∞ regression, and random LP (four-LP medians).

		Iterations to ϵ_{tol}			Matrix–vector products			Algorithm wall clock (ms)		
Family	(m, n)	Ours	HT	MPAX	Ours	HT	MPAX	Ours	HT	MPAX
Robust inventory	(390, 1,101)	**1,426**	2,892	3,774	**2,853**	5,787	7,682	**259**	397	309
	(760, 2,201)	**1,250**	4,242	3,072	**2,503**	8,485	6,256	**260**	659	482
	(1,500, 4,401)†	**1,905**	4,266	3,875	**3,812**	8,535	7,872	507	915	**432**
	(2,240, 6,601)	**2,140**	10,294	5,888	**4,282**	20,591	11,934	635	2,529	**591**
	(2,980, 8,801)	**2,311**	7,768	5,348	**4,624**	15,537	10,854	**570**	1,494	661
	(4,460, 13,201)	**2,874**	25,798	6,121	**5,749**	51,599	12,391	**858**	5,978	895
	(7,420, 22,001)	**3,833**	12,103	6,167	**7,668**	24,208	12,495	1,164	3,034	**975**
	(11,860, 35,201)	**6,308**	13,499	8,588	**12,618**	27,000	17,356	1,920	3,209	**1,505**
	(14,820, 44,001)	**7,832**	–	11,581	**15,666**	–	23,370	2,239	–	**1,834**
L_∞ regression	(120, 9)	**190**	465	702	**383**	932	1,466	**28**	42	69
	(160, 9)	**169**	554	907	**340**	1,110	1,876	**25**	51	81
	(200, 9)	**127**	533	740	**256**	1,068	1,544	**20**	49	72
	(240, 9)†	**150**	586	834	**303**	1,175	1,734	**24**	54	82
	(480, 9)	**270**	1,060	786	**543**	2,121	1,637	**44**	113	72
	(1,200, 9)	**236**	673	760	**475**	1,348	1,583	**41**	78	69
	(1,440, 9)	**308**	741	764	**619**	1,484	1,590	**54**	92	76
	(1,680, 9)	**358**	752	782	**718**	1,505	1,626	**62**	93	78
Random LP	(4,000, 8,000)	**264**	424	706	**531**	850	1,468	**87**	109	221
	(10,000, 20,000)†	**284**	440	664	**570**	883	1,384	**110**	140	408
	(20,000, 40,000)	**270**	416	650	**541**	834	1,359	**125**	164	773
	(30,000, 60,000)	**280**	421	663	**561**	844	1,384	**151**	199	1,175
	(40,000, 80,000)	**264**	429	666	**530**	860	1,388	**169**	244	1,612
	(60,000, 120,000)	**304**	426	774	**609**	854	1,609	**243**	310	2,443
	(100,000, 200,000)	**314**	455	711	**629**	912	1,481	**363**	490	5,283
	(150,000, 300,000)	**290**	428	702	**582**	859	1,468	**450**	626	7,202

Ours: GALLOP; HT: GA-PDHG (HT). Bold: best per metric. †: largest training size. A dash denotes failure to reach $\epsilon_{\text{tol}} = 10^{-6}$ within the iteration limit.

F Implementation Details

F.1 Network architecture and action distributions

The GALLOP policy uses separate tanh MLPs for the restart decision, history coefficients, relaxation and anchor coefficients, step size, and primal weight. A separate network estimates the value function. Each network has two hidden layers of width 64. The restart head produces $p_t^e \in (0, 1)$ through a sigmoid and samples

$$e_t \sim \text{Bernoulli}(p_t^e). \tag{87}$$

Conditional on o_t, the continuous heads sample independent Gaussian latents with the means $\mu_t^{\alpha^x}, \mu_t^{\alpha^y}, \boldsymbol{\mu}_t^\beta, \mu_t^\zeta, \mu_t^\eta, \mu_t^\omega$ output by their respective networks; all but $\boldsymbol{\mu}_t^\beta \in \mathbb{R}^k$ are scalars. Their standard deviations $\sigma_\alpha, \sigma_\beta, \sigma_\zeta, \sigma_\eta, \sigma_\omega > 0$ are learned globally and do not depend on o_t. Their logarithms are optimized with the corresponding actor and clipped to group-specific lower bounds and a common upper bound when the distributions are evaluated. Appendix G.1 gives the initialization values and bounds used for each family. The primal and dual relaxation latents share σ_α, and all history coordinates share σ_β. We restrict the learned coefficients to $\alpha_t^x, \alpha_t^y > 0$ to focus the policy search on extrapolation beyond the PDHG output, motivated by potential acceleration rather than damping through underrelaxation. Exponential transformations enforce positive relaxation coefficients and a positive anchor multiplier,

$$u_t^{\alpha^x} \sim \mathcal{N}\left(\mu_t^{\alpha^x}, \sigma_\alpha^2\right), \qquad \alpha_t^x = \exp\left(u_t^{\alpha^x}\right), \tag{88}$$

$$u_t^{\alpha^y} \sim \mathcal{N}\left(\mu_t^{\alpha^y}, \sigma_\alpha^2\right), \qquad \alpha_t^y = \exp\left(u_t^{\alpha^y}\right), \tag{89}$$

$$u_t^\zeta \sim \mathcal{N}\left(\mu_t^\zeta, \sigma_\zeta^2\right), \qquad \widetilde{\zeta}_t = \exp\left(u_t^\zeta\right). \tag{90}$$

The effective anchor coefficient ζ_t is computed by Eq. (50), with $\zeta_t = 0$ when $z_\star = z_t$. During training, the history coefficients are sampled from an unbounded Gaussian distribution,

$$\boldsymbol{\beta}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_t^\beta, \sigma_\beta^2 I_k\right). \tag{91}$$

At every iteration, including restarts, the step-size sample is transformed as

$$u_t^\eta \sim \mathcal{N}\left(\mu_t^\eta, \sigma_\eta^2\right), \qquad \eta_t = \frac{0.1 + 4.9\,\mathrm{sig}\left(u_t^\eta\right)}{\|A\|_2}. \tag{92}$$

Here $\mathrm{sig}(u) := 1/(1+e^{-u})$ is the sigmoid. The denominator estimates the spectral norm of the matrix supplied to PDHG. Sparse instances use a converged spectral-norm calculation during both training and evaluation; dense training uses 100 power iterations, while dense evaluation uses a converged calculation. The initialization step uses $0.998/\|A\|_2$. The learned step size can exceed $1/\|A\|_2$ and therefore does not enforce the fixed-step nonexpansiveness condition. At every iteration, including non-restart transitions, the primal-weight sample sets an absolute target relative to ω_0,

$$u_t^\omega \sim \mathcal{N}\left(\mu_t^\omega, \sigma_\omega^2\right), \qquad \omega_t = \omega_0 \exp\left(L_\omega \tanh\left(u_t^\omega\right)\right). \tag{93}$$

Here $L_\omega = 2$ for auction and $L_\omega = 4.6$ for the other five families. This weight is applied in the next PDHG map regardless of e_t; it does not multiply the previous weight. The rollout records the raw action

$$a_t = (e_t, u_t^{\alpha^x}, u_t^{\alpha^y}, \boldsymbol{\beta}_t, u_t^\zeta, u_t^\eta, u_t^\omega).$$

Let a_t^{act} consist of e_t, u_t^η, u_t^ω, and the acceleration groups enabled when $e_t = 0$. Marginalizing the unused draws gives the density of this effective action,

$$\begin{aligned}\pi_{\theta,\mathrm{act}}\left(a_t^{\mathrm{act}} \mid o_t\right) = (p_t^e)^{e_t}(1-p_t^e)^{1-e_t}\mathcal{N}\left(u_t^\eta; \mu_t^\eta, \sigma_\eta^2\right)\mathcal{N}\left(u_t^\omega; \mu_t^\omega, \sigma_\omega^2\right) \\ \times \begin{bmatrix}\mathcal{N}\left(u_t^{\alpha^x}; \mu_t^{\alpha^x}, \sigma_\alpha^2\right)\mathcal{N}\left(u_t^{\alpha^y}; \mu_t^{\alpha^y}, \sigma_\alpha^2\right) \\ \times \mathcal{N}\left(\boldsymbol{\beta}_t; \boldsymbol{\mu}_t^\beta, \sigma_\beta^2 I_k\right)\mathcal{N}\left(u_t^\zeta; \mu_t^\zeta, \sigma_\zeta^2\right)\end{bmatrix}^{1-e_t}.\end{aligned} \tag{94}$$

The step-size and primal-weight factors are active on every transition, and the acceleration factors only when $e_t = 0$. Inactive factors contribute zero to the log-density. PPO evaluates the density on the raw latents, so the output transformations require no Jacobian correction in the likelihood ratio.

Deterministic inference. At inference, $e_t = \mathbf{1}[p_t^e \geq 0.5]$, and every Gaussian latent is replaced by its mean before applying the same transformations. The observation map and action transforms are unchanged, and the trained parameters remain fixed throughout the solve. When $e_t = 0$, the transformed coefficients form the accelerated input in Eq. (5a); when $e_t = 1$, the input is z_t and the history is reset after the PDHG step. The step size and primal weight are applied on both branches at every iteration. The critic and Gaussian standard deviations are needed for training, but do not determine the inference actions.

F.2 GALLOP OBSERVATIONS

For $t \geq 1$, the observation o_t is evaluated before a_t using the carried weight ω_{t-1}. The initial weight ω_0 is used for the initialization step that produces z_1. It is $\|c\|_2/\|b\|_2$ in solver coordinates when both norms are positive, and 1 otherwise. The features below follow the convergence state, restart context, and local geometry described in Section 3.2.

Convergence state. The primal residual, dual residual, and duality-gap diagnostic in original LP coordinates are defined in Eq. (6). The sign cone is $\Lambda := \prod_{i=1}^n \Lambda_i$, where $\Lambda_i = \mathbb{R}$ if both bounds are finite, $\mathbb{R}_{\geq 0}$ if only l_i is finite, $\mathbb{R}_{\leq 0}$ if only u_i is finite, and $\{0\}$ if neither is finite. With elementwise $[a]^\pm := \max\{\pm a, 0\}$, the box minimum in $r_g(z)$ equals $l^\top[\bar{c}]^+ - u^\top[\bar{c}]^-$, with infinite-bound terms omitted. When $y \in Y$ and $r_d(z) = 0$, $r_g(z)$ is the primal–dual objective gap. Otherwise it is a finite diagnostic used together with the feasibility residuals. The corresponding log normalized metrics are

$$p(z) := \log\left(\frac{\max\left\{\|r_p(z)\|_\infty, \|[l-x]^+\|_\infty, \|[x-u]^+\|_\infty\right\}}{1+\max\left\{\|b\|_\infty, \|Ax\|_\infty\right\}}\right). \tag{95}$$

$$d(z) := \log\left(\frac{\max\left\{\|r_d(z)\|_\infty, \|[-y^{\mathrm{in}}]^+\|_\infty\right\}}{1+\max\left\{\|c\|_\infty, \|A^\top y\|_\infty\right\}}\right). \tag{96}$$

$$g(z) := \log\left(\frac{|r_g(z)|}{1+\max\left\{|c^\top x|, |c^\top x - r_g(z)|\right\}}\right). \tag{97}$$

All three metrics are evaluated in the original LP coordinates, with log arguments floored at 10^{-20}. The bound and dual-sign violation terms vanish on $X \times Y$. Thus these formulas also apply to the initial anchor $z_\star = 0$, which may lie outside X.

For the controller, fixed-point progress uses the executed direction $z_t - \hat{z}_t$ in solver coordinates, measured in a fixed metric,

$$M_t^{\mathrm{fp}} := \|z_t - \hat{z}_t\|_{Q(1,0.998/\|A\|_2)}. \tag{98}$$

The measurement metric uses the scaled A when preconditioning is enabled. Its spectral norm is computed by dense SVD for dense training batches and by a converged calculation for sparse batches and evaluation, independently of the dense training step-size estimate. For $t \geq 1$, it measures the step executed with $(\omega_{t-1}, \eta_{t-1})$. The observation includes progress relative to the initial step, the restart baseline $M_\star$, and the preceding step,

$$\log \frac{\max\{M_t^{\mathrm{fp}}, 10^{-20}\}}{\max\{M_1^{\mathrm{fp}}, 10^{-20}\}}, \qquad \log \frac{\max\{M_t^{\mathrm{fp}}, 10^{-20}\}}{\max\{M_\star, 10^{-20}\}}, \qquad \log \frac{\max\{M_t^{\mathrm{fp}}, 10^{-20}\}}{\max\{M_{t-1}^{\mathrm{fp}}, 10^{-20}\}}. \tag{99}$$

It also includes the one-step changes in the KKT metrics,

$$p(z_t) - p(z_{t-1}), \qquad d(z_t) - d(z_{t-1}), \qquad g(z_t) - g(z_{t-1}). \tag{100}$$

The one-step features are evaluated only for $t \geq 2$ and are set to zero at $t = 1$. The initial restart baseline is $M_\star = M_1^{\mathrm{fp}}$.

Restart context. The policy observes the log restart age and global iteration count, $\log(s+1)$ and $\log t$, and the KKT progress from the anchor,

$$p(z_t) - p(z_\star), \qquad d(z_t) - d(z_\star), \qquad g(z_t) - g(z_\star). \tag{101}$$

At initialization, the anchor metrics are evaluated at $z_\star = 0$. After a restart step, $z_\star \leftarrow z_{t+1}$ and $M_\star \leftarrow M_{t+1}^{\mathrm{fp}}$, so the anchor-relative KKT differences are zero. Primal–dual balance, the relative weight, and its one-step change are measured by

$$b_t^e := \log\left(\frac{\omega_{t-1}\|x_t - x_\star\|_2 + 10^{-20}}{\|y_t - y_\star\|_2 + 10^{-20}}\right), \qquad \log\left(\frac{\omega_{t-1}}{\omega_0}\right), \qquad \log\left(\frac{\omega_{t-1}}{\omega_{t-2}}\right). \tag{102}$$

The weight-change feature is set to zero at $t = 1$. The displacements in these features use the solver coordinates. KKT residuals evaluated at $\hat{z}_t$ are not supplied to the policy.

Local geometry. Partition the history directions into primal and dual blocks,

$$\mathbf{K}_t = \begin{pmatrix} \mathbf{K}_t^x \\ \mathbf{K}_t^y \end{pmatrix}, \qquad \mathbf{K}_t^x \in \mathbb{R}^{n \times k}, \quad \mathbf{K}_t^y \in \mathbb{R}^{m \times k}. \tag{103}$$

For $z = \left(x^\top, y^\top\right)^\top$, define the weighted norm

$$\|z\|_\omega^2 := \omega\|x\|_2^2 + \omega^{-1}\|y\|_2^2. \tag{104}$$

Using this norm and $\epsilon_{\mathrm{geom}} = 10^{-8}$, the normalized history Gram matrix and its alignment with the current direction are

$$\begin{aligned} \Gamma_t &:= \frac{\omega_{t-1}(\mathbf{K}_t^x)^\top \mathbf{K}_t^x + \omega_{t-1}^{-1}(\mathbf{K}_t^y)^\top \mathbf{K}_t^y}{\|z_t - \hat{z}_t\|_{\omega_{t-1}}^2 + \epsilon_{\mathrm{geom}}^2} \in \mathbb{R}^{k \times k}, \\ c_t^{\mathrm{hist}} &:= \frac{\omega_{t-1}(\mathbf{K}_t^x)^\top (x_t - \hat{x}_t) + \omega_{t-1}^{-1}(\mathbf{K}_t^y)^\top (y_t - \hat{y}_t)}{\|z_t - \hat{z}_t\|_{\omega_{t-1}}^2 + \epsilon_{\mathrm{geom}}^2} \in \mathbb{R}^k. \end{aligned} \tag{105}$$

The anchor features measure its relative magnitude and weighted alignment with the current direction,

$$\begin{aligned} \rho_t^{\mathrm{anc}} &:= \log \frac{\|z_\star - z_t\|_{\omega_{t-1}} + \epsilon_{\mathrm{geom}}}{\|z_t - \hat{z}_t\|_{\omega_{t-1}} + \epsilon_{\mathrm{geom}}}, \\ c_t^{\mathrm{anc}} &:= \frac{\omega_{t-1}(x_\star - x_t)^\top (x_t - \hat{x}_t) + \omega_{t-1}^{-1}(y_\star - y_t)^\top (y_t - \hat{y}_t)}{\left(\|z_\star - z_t\|_{\omega_{t-1}} + \epsilon_{\mathrm{geom}}\right)\left(\|z_t - \hat{z}_t\|_{\omega_{t-1}} + \epsilon_{\mathrm{geom}}\right)}. \end{aligned} \tag{106}$$

The history quantities undergo an elementwise signed logarithm before entering the policy,

$$\begin{aligned} &\operatorname{vech}\left(\operatorname{sign}(\Gamma_t)\log\left(1+|\Gamma_t|\right)\right), \qquad \operatorname{sign}(c_t^{\mathrm{hist}})\log\left(1+|c_t^{\mathrm{hist}}|\right), \\ &\rho_t^{\mathrm{anc}}, \qquad c_t^{\mathrm{anc}}. \end{aligned} \tag{107}$$

Here vech stacks the upper-triangular entries, including the diagonal, in row-major order. The sign, absolute value, logarithm, and their products act elementwise. The observation has $19+k+k(k+1)/2$ entries, giving 33 inputs for $k = 4$. These inputs are passed directly to the networks without running normalization or input clipping.

F.3 Policy optimization

At each PPO iteration, the parameters $(\theta_{\mathrm{old}}, \phi_{\mathrm{old}})$ are frozen while one episode is collected per LP instance. For an episode of T transitions, generalized advantage estimation (Schulman et al., 2016) uses discount factor $\gamma = 1$ and GAE coefficient $\kappa = 0.99$,

$$\begin{aligned} \delta_t &:= r_t + \gamma\left(1-\mathrm{success}_t\right)V^{\phi_{\mathrm{old}}}(o_{t+1}) - V^{\phi_{\mathrm{old}}}(o_t), \\ \hat{\Psi}_t &:= \sum_{j=0}^{T-t}(\gamma\kappa)^j\,\delta_{t+j}, \qquad 1 \le t \le T. \end{aligned} \tag{108}$$

Here $\mathrm{success}_t := \mathbf{1}\left[M^{\mathrm{rel}}(z_{t+1}) \le \log\epsilon_{\mathrm{tol}}\right]$. The sum ends with the episode. A successful episode has terminal value zero; truncation at $T_{\max}$ retains $V^{\phi_{\mathrm{old}}}(o_{T+1})$ in the final temporal-difference residual. The critic target and the advantage normalized over the full rollout buffer are

$$\hat{R}_t := \hat{\Psi}_t + V^{\phi_{\mathrm{old}}}(o_t), \qquad \tilde{\Psi}_t := \frac{\hat{\Psi}_t - \mathrm{mean}(\hat{\Psi})}{\mathrm{std}(\hat{\Psi}) + 10^{-8}}. \tag{109}$$

Here $\mathrm{mean}(\hat{\Psi})$ and $\mathrm{std}(\hat{\Psi})$ are the empirical mean and population standard deviation of all valid advantages in $\mathcal{B}$. Writing $\hat{\mathbb{E}}_t$ for the empirical average over that buffer, the critic loss is

$$\mathcal{L}_V(\phi) := \hat{\mathbb{E}}_t\left[\left(V^{\phi}(o_t) - \hat{R}_t\right)^2\right]. \tag{110}$$

Groupwise PPO. Let $\pi_{\theta,e}$ and $\pi_{\theta,\beta}$ denote the restart and history densities, $\pi_{\theta,\alpha\zeta}$ the product of the relaxation and anchor Gaussian densities, and $\pi_{\theta,\eta}, \pi_{\theta,\omega}$ the step-size and primal-weight latent densities. Each density below is evaluated at its recorded component of a_t, conditional on the recorded o_t; these arguments are suppressed. For the action groups $\mathcal{G} = \{e, \beta, \alpha\zeta, \eta, \omega\}$, the masked log-densities are

$$\boldsymbol{\ell}_t := \left(\log\pi_{\theta,e}, (1-e_t)\log\pi_{\theta,\beta}, (1-e_t)\log\pi_{\theta,\alpha\zeta}, \log\pi_{\theta,\eta}, \log\pi_{\theta,\omega}\right)^{\top}. \tag{111}$$

For $j \in \mathcal{G}$, let ℓ_t^j be the corresponding entry of $\boldsymbol{\ell}_t$ and $\ell_{t,\mathrm{old}}^j$ its value under θ_{old}. Each group uses its own likelihood ratio and clipped surrogate, with clipping parameter $\varepsilon^{\mathrm{clip}} = 0.2$,

$$\begin{aligned} \rho_t^j &:= \exp\left(\ell_t^j - \ell_{t,\mathrm{old}}^j\right), \\ \mathcal{J}_j(\theta) &:= \hat{\mathbb{E}}_t\left[\min\left(\rho_t^j\tilde{\Psi}_t, \operatorname{clip}\left(\rho_t^j, 1-\varepsilon^{\mathrm{clip}}, 1+\varepsilon^{\mathrm{clip}}\right)\tilde{\Psi}_t\right)\right]. \end{aligned} \tag{112}$$

Here $\mathrm{clip}(v, a, b) := \min\{b, \max\{a, v\}\}$. The five surrogates are summed; the advantages and recorded old log-densities are held fixed during optimization.

On restart rows, the acceleration groups have $\rho_t^j = 1$ and contribute no policy gradient. The restart, step-size, and primal-weight surrogates use all valid transitions in the common minibatches and their full-trajectory, globally normalized advantages $\tilde{\Psi}_t$. The primal-weight actor is therefore also updated when the rollout contains no restarts. Each actor and the critic has its own gradient clipping and Adam state. The critic minimizes $c_V\mathcal{L}_V$, with c_V specified in Appendix G.1.

Auxiliary restart supervision. Before the action is sampled, the conventional sufficient-progress, deterioration, and epoch-length criteria provide a binary label y_t^{prior}. The label is one if $M_t^{\text{fp}} \le 0.2M_\star$, if both $M_t^{\text{fp}} \le 0.6M_\star$ and $M_t^{\text{fp}} > M_{t-1}^{\text{fp}}$, or if $s \ge 0.2t$. At $t = 1$, the preceding residual in this test is taken to equal M_1^{fp}. It is zero otherwise and is always set to zero when $s = 0$. These tests use the policy trajectory and do not execute a restart. The auxiliary loss and the resulting actor loss are

$$\begin{aligned}\mathcal{L}_{\text{prior}} &:= -\hat{\mathbb{E}}_t\left[y_t^{\text{prior}} \log p_t^e + (1 - y_t^{\text{prior}}) \log(1 - p_t^e)\right], \\ \mathcal{L}_{\text{actor}} &:= -\sum_{j\in\mathcal{G}} \mathcal{J}_j + \lambda_{\text{prior}}\mathcal{L}_{\text{prior}}.\end{aligned} \tag{113}$$

The coefficient λ_{prior} is fixed within each run; its family-specific value is given in Appendix G.1. The binary cross-entropy term updates only the restart actor; its pre-action labels are held fixed during optimization. The rollout stores the observations, raw actions, masked log-densities, rewards, success indicators, critic values, and pre-action labels,

$$\tau = \left\{\left(o_t, a_t, \boldsymbol{\ell}_t, r_t, \text{success}_t, V^\phi(o_t), y_t^{\text{prior}}\right)\right\}_{t=1}^{T}. \tag{114}$$

F.4 Preconditioning

We apply 10 iterations of Ruiz scaling (Ruiz, 2001), followed by Pock–Chambolle scaling with exponent 1 (Pock & Chambolle, 2011). Let $D_r \in \mathbb{R}^{m\times m}$ and $D_c \in \mathbb{R}^{n\times n}$ be the resulting positive diagonal row and column scalings, with $D_r = \begin{pmatrix} D_P & 0 \\ 0 & D_G \end{pmatrix}$ partitioned by equality and inequality constraints. The scaled problem data and their relation to the original variables are

$$\begin{aligned} A^{\text{sc}} &= D_r A D_c, & b^{\text{sc}} &= D_r b, & c^{\text{sc}} &= D_c c, \\ P^{\text{sc}} &= D_P P D_c, & q^{\text{sc}} &= D_P q, & & \\ G^{\text{sc}} &= D_G G D_c, & h^{\text{sc}} &= D_G h, & & \\ l^{\text{sc}} &= D_c^{-1} l, & u^{\text{sc}} &= D_c^{-1} u. & & \end{aligned} \tag{115}$$

$$x = D_c x^{\text{sc}}, \qquad y = D_r y^{\text{sc}}. \tag{116}$$

In this subsection, superscript sc explicitly identifies scaled quantities, and unsuperscripted data and $z = \left(x^\top, y^\top\right)^\top$ refer to the original LP. The solver notation elsewhere omits sc; calls to p, d, g, and M^{rel} include the coordinate conversion in Eq. (116). PDHG operates in scaled coordinates, while convergence metrics, rewards, and termination are evaluated for the original LP. The residual transformations and gap invariance are

$$r_p^{\text{sc}} = D_r r_p, \qquad r_d^{\text{sc}} = D_c r_d, \tag{117}$$

$$r_g^{\text{sc}} = r_g. \tag{118}$$

As in Eq. (6), infinite-bound terms are omitted from the computed gap. Original residual norms can be recovered from scaled residuals,

$$\begin{aligned} \|r_p\|_\infty &= \|D_r^{-1} r_p^{\text{sc}}\|_\infty = \max_i \frac{\left|(r_p^{\text{sc}})_i\right|}{(D_r)_{ii}}, \\ \|r_d\|_\infty &= \|D_c^{-1} r_d^{\text{sc}}\|_\infty = \max_j \frac{|(r_d^{\text{sc}})_j|}{(D_c)_{jj}}. \end{aligned} \tag{119}$$

The normalized primal and dual metrics, including bound and dual-sign violations, are therefore

$$p(z) = \log\left(\frac{\max\left\{\|D_r^{-1} r_p^{\text{sc}}\|_\infty, \|D_c[l^{\text{sc}} - x^{\text{sc}}]^+\|_\infty, \|D_c[x^{\text{sc}} - u^{\text{sc}}]^+\|_\infty\right\}}{1 + \max\left\{\|b\|_\infty, \|D_r^{-1} A^{\text{sc}} x^{\text{sc}}\|_\infty\right\}}\right). \tag{120}$$

$$d(z) = \log\left(\frac{\max\left\{\|D_c^{-1} r_d^{\text{sc}}\|_\infty, \|D_G[-y^{\text{sc,in}}]^+\|_\infty\right\}}{1 + \max\left\{\|c\|_\infty, \|D_c^{-1}(A^{\text{sc}})^\top y^{\text{sc}}\|_\infty\right\}}\right). \tag{121}$$

The primal objective is also invariant, $(c^{\mathrm{sc}})^\top x^{\mathrm{sc}} = c^\top x$, so the gap metric is

$$g(z) = \log\left(\frac{|r_g^{\mathrm{sc}}(z^{\mathrm{sc}})|}{1 + \max\left\{|(c^{\mathrm{sc}})^\top x^{\mathrm{sc}}|, |(c^{\mathrm{sc}})^\top x^{\mathrm{sc}} - r_g^{\mathrm{sc}}(z^{\mathrm{sc}})|\right\}}\right). \tag{122}$$

These are evaluation formulas for the original metrics, with the same 10^{-20} log floor as in Appendix F.2. They use only scaled residuals, cached products, and diagonal scalings, requiring no additional multiplication by A or $A^\top$. The merit M^{rel}, tolerance test, and reward in Eq. (8) use these original-coordinate metrics. Preconditioned and unpreconditioned runs are thus evaluated at the same original-LP tolerance.

F.5 Matrix–vector product caching

For the primal and dual blocks of the vectors used in Eq. (5a), we retain

$$(Ax_t,\ A\hat{x}_t,\ Ax_\star,\ A\mathbf{K}_t^x) \qquad \text{and} \qquad (A^\top y_t,\ A^\top \hat{y}_t,\ A^\top y_\star,\ A^\top \mathbf{K}_t^y).$$

Applying the acceleration coefficients to these products reconstructs $A\hat{x}_{t+1}$ and $A^\top \hat{y}_{t+1}$ without another application of either matrix. The newest history products are $Ax_{t+1} - Ax_t$ and $A^\top y_{t+1} - A^\top y_t$. The remaining columns are shifted. At a restart, the products for the new anchor are copied from those of z_{t+1}, and the history matrices and their products are reset together.

For $t \geq 1$, define the PDHG direction $q_t := z_t - \hat{z}_t$, with primal and dual blocks $q_t^x := x_t - \hat{x}_t$ and $q_t^y := y_t - \hat{y}_t$. The implementation applies the matrices to the blocks of q_{t+1}. The only new sparse products are Aq_{t+1}^x for the dual update and $A^\top q_{t+1}^y$ for the residual features. The output products are recovered as $Ax_{t+1} = A\hat{x}_{t+1} + Aq_{t+1}^x$ and $A^\top y_{t+1} = A^\top \hat{y}_{t+1} + A^\top q_{t+1}^y$. The primal update uses the reconstructed $A^\top \hat{y}_{t+1}$. For the observation, we also cache the unweighted Gram matrices $(\mathbf{K}_t^x)^\top \mathbf{K}_t^x$ and $(\mathbf{K}_t^y)^\top \mathbf{K}_t^y$. Only their newest row and column require fresh inner products when a history direction is inserted. The remaining submatrices are shifted, and both Gram matrices are zeroed at restarts. GA-PDHG thus uses two new sparse matrix–vector products per iteration, with $O\left((m+n)\,k + k^2\right)$ additional work for acceleration and cached geometry updates. GALLOP additionally evaluates the policy at each iteration.

F.6 Full GALLOP algorithm

Algorithm 1 collects one training episode of GA-PDHG under policy π_θ on an LP instance I, storing its T transitions in τ. The critic V^ϕ supplies value estimates for training. Algorithm 2 trains the GALLOP policy, with $\mathcal{B}$ denoting the buffer of valid transitions from all episodes in a PPO iteration. Feature definitions and policy losses are provided in Appendix F.2 and Appendix F.3, respectively. Initialization takes one unaccelerated PDHG step before the first controlled iteration. The history stores only output differences inserted after initialization or the latest restart, with unavailable columns set to zero. In the history update, $\mathbf{K}_t(:, 1:k-1)$ denotes its first $k-1$ columns and is empty for $k = 1$. The returned V_{T+1}^ϕ is the terminal value or truncation bootstrap used by GAE. The cap $T_{\max}$ counts controlled iterations and excludes the initial unaccelerated PDHG step. We assume $\|A\|_2 > 0$ for the step-size formulas and require $\epsilon_{\mathrm{tol}} \geq 10^{-20}$ to respect the numerical floor in the log residuals.

Algorithm 1 GALLOP training rollout

Require: Positive integer cap $T_{\max}$, memory $k \geq 1$, tolerance ϵ_{tol}, and controls and features specified in Appendix F.1 and Appendix F.2

function GALLOP-ROLLOUT(π_θ, V^ϕ, I)
  Initialize $z_\star = \hat{z}_1 = z_0 = 0$, weight ω_0, $s \leftarrow 1$, $t \leftarrow 1$, $\tau \leftarrow \emptyset$
  $z_1 \leftarrow \mathcal{H}(\hat{z}_1; \omega_0, \eta_0)$, $\mathbf{K}_1 \leftarrow 0$
  $M_\star \leftarrow M_1^{\text{fp}}$ by Eq. (98); initialize observation baselines as in Appendix F.2
  Form o_1 using Appendix F.2
  while $t \leq T_{\max}$ **do**
    Evaluate y_t^{prior} and $V^\phi(o_t)$; sample a_t using Appendix F.1
    Record $\boldsymbol{\ell}_t$ by Eq. (111); set η_t, ω_t by Eq. (92) and Eq. (93)
    if $e_t = 0$ **then**
      Form $\hat{z}_{t+1}$ by Eq. (5a), using the transformed acceleration coefficients
    else
      $\hat{z}_{t+1} \leftarrow z_t$
    end if
    $z_{t+1} \leftarrow \mathcal{H}(\hat{z}_{t+1}; \omega_t, \eta_t)$; measure M_{t+1}^{fp} by Eq. (98)
    if $e_t = 1$ **then**
      $z_\star \leftarrow z_{t+1}$, $M_\star \leftarrow M_{t+1}^{\text{fp}}$, $\mathbf{K}_{t+1} \leftarrow 0$, $s \leftarrow 0$
    else
      $\mathbf{K}_{t+1} \leftarrow [z_{t+1} - z_t \quad \mathbf{K}_t(:, 1\!:\!k-1)]$, $s \leftarrow s + 1$
    end if
    Evaluate the KKT metrics and r_t by Eq. (8)
    $\text{success}_t \leftarrow \mathbf{1}\left[M^{\text{rel}}(z_{t+1}) \leq \log \epsilon_{\text{tol}}\right]$
    Append the transition to τ as in Eq. (114)
    $t \leftarrow t + 1$; form o_t using Appendix F.2
    if $\text{success}_{t-1} = 1$ **then**
      break
    end if
  end while
  $T \leftarrow t - 1$
  if $\text{success}_T = 1$ **then**
    $V_{T+1}^\phi \leftarrow 0$
  else
    $V_{T+1}^\phi \leftarrow V^\phi(o_{T+1})$ ▷ bootstrap at the episode cap
  end if
  return τ, V_{T+1}^ϕ
end function

Algorithm 2 PPO training of GALLOP

Require: LP dataset $\mathcal{D} = \{I^{(i)}\}_{i=1}^N$, number of PPO iterations J, history length k, control and standard-deviation bounds, $\gamma, \kappa, \varepsilon^{\text{clip}}, \lambda_{\text{prior}}, c_V, \chi, \epsilon_{\text{tol}}, T_{\max}, \epsilon_{\text{geom}}$

Initialize five actors, global log standard deviations, critic V^ϕ, and their optimizers
for $j = 1, \ldots, J$ **do**
  $(\theta_{\text{old}}, \phi_{\text{old}}) \leftarrow (\theta, \phi)$, $\mathcal{B} \leftarrow \emptyset$
  for $i = 1, \ldots, N$ in batches **do**
    $\tau^{(i)}, V_{T^{(i)}+1}^{\phi_{\text{old}}} \leftarrow$ GALLOP-ROLLOUT$(\pi_{\theta_{\text{old}}}, V^{\phi_{\text{old}}}, I^{(i)})$
    Compute $\hat{\Psi}_t^{(i)}$ by Eq. (108), using $V_{T^{(i)}+1}^{\phi_{\text{old}}}$ for the final bootstrap
    Append valid rollout records and their advantages to $\mathcal{B}$
  end for
  Compute $\tilde{\Psi}_t$ and $\hat{R}_t$ over the full buffer by Eq. (109)
  for several epochs of minibatches from $\mathcal{B}$ **do**
    Update all five actors on $\mathcal{L}_{\text{actor}}$ in Eq. (113)
    Update the critic on $c_V \mathcal{L}_V$ in Eq. (110)
  end for
end for
return θ, ϕ

G Details on Experiments

G.1 Implementation details

Hardware and numerical precision. All experiments use an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory and an Intel Core i9-13900K CPU. The implementation uses JAX 0.10.2 with CUDA 12 and Flax 0.12.8; the external baseline uses MPAX 0.2.4. The policy networks compute in float32, while the PDHG state, residuals, rewards, and advantage estimates use float64.

Optimization settings. Table 13 lists the settings of the runs that produced the selected checkpoints. All networks use Adam with a constant learning rate, four PPO epochs per rollout buffer, and a separate gradient-norm clip of 0.5 per network. All families use $k = 4$, $\gamma = 1$, $\kappa = 0.99$, PPO clipping $\varepsilon^{\text{clip}} = 0.2$, and no entropy bonus.

Table 13: Settings of the training runs used in the six-family comparison.

Family	Learning rate	PPO minibatch	c_V	λ_{prior}	L_ω
Transport	5×10^{-4}	16,384	0.5	0.02	4.6
Auction	5×10^{-4}	4,096	0.5	0	2.0
Min-cost flow	10^{-4}	16,384	0.5	0.02	4.6
Robust inventory	10^{-4}	16,384	0.5	0.05	4.6
L_∞ regression	5×10^{-4}	16,384	0.2	0.01	4.6
Random LP	10^{-4}	16,384	0.5	0.02	4.6

Exploration standard deviations. Each continuous control group has a learned global log standard deviation, shared across observations and updated with its actor. For $j \in \{\alpha, \beta, \zeta, \eta, \omega\}$, let ℓ_j denote this learned scalar. The distribution uses

$$\sigma_j = \exp\left(\text{clip}\left(\ell_j, \log \sigma_j^{\min}, \log 1.5\right)\right). \tag{123}$$

Table 14 gives the initialization and lower bounds from the selected runs. The two relaxation latents share σ_α, and all k history coordinates share σ_β. Inference replaces the Gaussian draws by their means.

Table 14: Gaussian exploration settings. Both tuples use the order $(\alpha, \beta, \zeta, \eta, \omega)$; the common upper bound is 1.5.

Family	Initial standard deviations	Minimum standard deviations
Transport	$(0.3, 0.2, 0.4, 0.4, 0.2)$	$(0.2, 0.1, 0.2, 0.2, 0.1)$
Auction	$(0.4, 0.2, 0.4, 0.6, 0.2)$	$(0.001, 0.001, 0.001, 0.12, 0.001)$
Min-cost flow	$(0.2, 0.3, 0.3, 0.4, 0.2)$	$(0.05, 0.2, 0.05, 0.05, 0.05)$
Robust inventory	$(0.05, 0.05, 0.07, 0.1, 0.07)$	$(0.01, 0.01, 0.01, 0.02, 0.02)$
L_∞ regression	$(0.4, 0.2, 0.4, 0.5, 0.2)$	$(0.2, 0.1, 0.05, 0.3, 0.1)$
Random LP	$(0.1, 0.1, 0.1, 0.1, 0.1)$	$(0.02, 0.01, 0.02, 0.02, 0.01)$

Batches across problem sizes. LPs with the same dimensions are grouped into a size bucket and solved in parallel within that bucket. Each PPO iteration collects one episode per training LP from every bucket before updating the policy. The valid transitions are then combined, their advantages are normalized over the full buffer, and shuffled PPO minibatches are formed using the sizes in Table 13. Thus rollout batches have fixed LP dimensions, while PPO minibatches can mix observations from different problem sizes because the observation and action dimensions depend only on k. We evaluate the same trained policy separately on each held-out and test bucket, including sizes larger than the training sizes listed in Appendix D.6.