UNSUPERVISED LEARNING OF SELF-CONSISTENT DUAL PRICES FOR COLUMN GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Column generation (CG) underpins algorithms for large-scale integer programming by generating columns without explicitly enumerating the full column pool. A fundamental challenge for CG is primal degeneracy in the restricted master problem (RMP), which can lead to repeated costly pricing subproblems that change dual prices without improving primal objective. To address this issue, we propose an unsupervised learning framework that trains a neural network to predict self-consistent dual prices (SCDP), whose displacement from the current RMP dual agrees with the ascent step over the dual objective induced by their own pricing responses. We prove that, at exact self-consistency, following SCDP guarantees sufficient ascent on the smoothed full-dual bound, thereby closing primal-dual gap even when the RMP objective stalls. We derive a proximal loss function from the smoothed full-dual objective and obtain its gradient directly from perturbed pricing responses, enabling gradient-based training. To make training practical, we establish that an unbiased gradient estimate can be obtained with a single perturbed pricing solve per training iteration. At deployment, the trained network predicts dual prices to guide column discovery, while exact pricing at the RMP dual preserves LP optimality. Experiments demonstrate runtime reductions of 63%, 94%, and 47% on the capacitated vehicle routing problem, its variant with time windows, and the cutting stock problem, respectively.

1 INTRODUCTION

Column generation (CG) solves linear programs whose columns are far too numerous to enumerate, and it is the core of branch-and-price algorithms for routing, scheduling, and cutting stock (Dantzig & Wolfe, 1960; Gilmore & Gomory, 1961; Barnhart et al., 1998; Desrosiers & Lübbecke, 2005). Each iteration of CG solves a restricted master problem (RMP) over the columns generated so far, receives its optimal dual, and calls a pricing subproblem (PP) that searches the full column pool for a column of negative reduced cost. One central bottleneck in CG is its slow convergence caused by degeneracy in RMPs, often associated with highly overlapping columns. When the RMP is degenerate, a new column may violate the dual constraint at the current optimal dual solution while leaving another solution on the RMP's optimal dual face feasible. Adding that column then leaves the RMP objective unchanged. Thus, pricing at an optimal RMP dual solution can produce columns that exclude only the current solution, leading to slow progress (Lübbecke & Desrosiers, 2005; Elhallaoui et al., 2005), as illustrated by standard CG in Figure 1 (gray line). Rather than relying on RMP to revise the dual solution after each round of columns, we ask:

> *Can we predict dual prices that guarantee progress toward solving the full master problem, without first generating the columns they induce?*

This question, however, involves a reciprocal dependence that is intrinsically difficult to resolve. Guaranteeing CG progress requires the predicted dual to account for the full column pool's response. Yet this response depends on the dual variables themselves, because changing the dual also changes which columns are generated and, consequently, the progress they support. One may address this issue in three ways (see Sec. 5 for a broader discussion). First, a new dual can be obtained by using previously generated columns, as in stabilization around a reference point (Pessoa et al., 2018). But this is blind to how full column pool responds to the new dual. Alternatively, one may also predict

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.

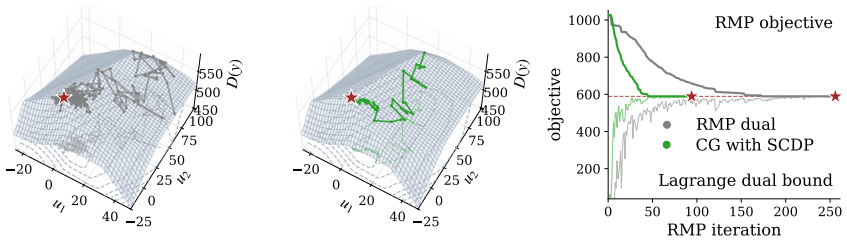


Figure 1: Illustration of SCDP-guided CG (middle) reaching LP optimality in 92 iterations on a CVRPTW instance (Solomon R201), versus 256 iterations for standard CG (left). The right panel shows the corresponding progress in RMP objectives and Lagrangian dual bound.

future information, with the optimal dual of the full master as the natural target (Kraul et al., 2023). But this requires instance-specific labels that may not generalize, and more importantly, knowing optimal dual does not translate into dual solutions useful to the current CG state. The third way is to directly tackle the dependence itself, by adjusting the dual using the columns it induces until the proposed move agrees with its own pricing response. Unfortunately, this is as difficult as solving the CG problem itself (Amor et al., 2009), and the gains are quickly outweighed by the excessive computational effort.

Contribution. We address the above reciprocal dependence by deriving a sufficient condition for dual-bound improvement that turns the search for improving dual solutions into an unsupervised learning problem. Once trained, the model predicts improving dual solutions from the current CG state without first generating the columns they induce. Figure 1 provides an example of the resulting acceleration (green line). Our contributions are as follows.

- We establish a connection between a dual's induced response and the progress it supports. Specifically, we define a *self-consistent dual price* (SCDP) whose displacement from the current RMP dual agrees with the ascent step induced by its own mean perturbed pricing response. We prove that exact self-consistency yields sufficient ascent on the smoothed full-dual bound, therefore guarantees to close primal-dual gap even when the RMP objective stalls.
- This self-consistency condition subsequently allows one to learn the dual without first computing them. We derive a strongly convex loss function whose unique minimizer is the SCDP and whose gradient measures the violation of self-consistency. Consequently, the pricing response at a predicted dual supplies the training signal needed to improve that prediction, without reference dual labels or differentiation through the combinatorial pricing solver.
- We make this learning objective practical by establishing when solutions of a single PP can be used to obtain unbiased gradient estimate without solving multiple PPs. Specifically, we prove a computable condition under which re-pricing the resulting column pool recovers exact full-pool responses across perturbations. For uncertified pools, we derive a correction that removes gradient bias using only one extra exact perturbed solve.

2 Learning Self-consistent Dual Prices for Column Generation

Preliminary. CG solves linear programs of the form

$$P = \min_{w \geq 0} \Big\{ \sum_{x \in \mathcal{R}} c_x w_x : \sum_{x \in \mathcal{R}} a_x w_x = b \Big\} \tag{MP}$$

without enumerating a column pool $\mathcal{R}$. Each column $x \in \mathcal{R}$ is a feasible structure of a combinatorial subproblem, such as a route or a cutting pattern, with cost c_x and coupling coefficients $a_x \in \mathbb{R}^m$. The pool is finite but typically exponential in size, and we assume that the master problem (MP) is feasible with finite optimal value. At iteration t, CG solves the restricted master problem (RMP) over a subset $\mathcal{R}_t \subseteq \mathcal{R}$, obtains its optimal dual y_t, and solves the pricing subproblem

$$\min_{x \in \mathcal{R}} \bar{c}_x(y_t), \qquad \bar{c}_x(q) = c_x - q^\top a_x. \tag{PP}$$

Columns of negative reduced cost are added to $\mathcal{R}_{t+1}$, and CG terminates when exact pricing returns none, in which case the RMP solution is optimal for (MP).

CG from the dual side. In the dual problem, every column corresponds to a constraint $a_x^\top q \le c_x$ on the dual vector $q \in \mathbb{R}^m$, and each PP acts as a separation oracle for the full dual polyhedron. The dual feasible region of the RMP contains only the generated column constraints, so it is an outer approximation of the full dual polyhedron, and y_t is one of its optimal solutions. A negative-reduced-cost column cuts off y_t (Figure 2, A1), but need not eliminate the entire optimal face. If the added cuts leave any point on that face feasible, reoptimizing the RMP returns another dual solution without changing the objective. While the RMP objective is determined by the generated columns, pricing searches the full column pool at y_t. Its response therefore provides first-order information about the Lagrangian dual function. We therefore seek to use this information to raise the lower bound, narrowing the primal–dual gap. This raises a direct question: *can we use pricing responses to construct dual solutions that improve the Lagrange dual bound, even when the RMP objective stalls?*

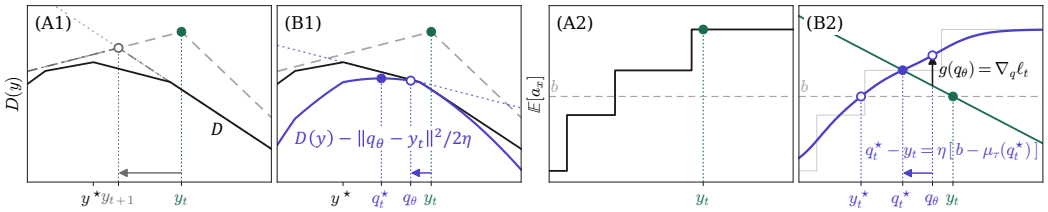


Figure 2: Illustration of our core idea. (A1) CG maximizes the RMP dual, which overestimates the full dual D. (A2) The response Ua_{x_q} is piecewise constant in q, so the subgradient $b - Ua_{x_q}$ jumps. (B1) Penalizing the move from y_t makes the maximizer unique, and a learned SCDP q_θ approximates it. (B2) The smoothed response u_τ is continuous, and $q_t^\star$ is where it meets $b - (q - y_t)/\eta$.

From cuts to learned dual bound ascent. We seek a sufficient condition under which a proposed dual price, together with its own pricing response, certifies ascent of a Lagrangian dual bound. Developing this condition and a differentiable objective for learning such dual prices requires addressing three obstacles, which the following subsections tackle in turn.

Obstacle (i) The Lagrangian dual function of (MP) equals $-\infty$ at any price for which some column has negative reduced cost. To make ascent possible, we need to bound the total column weight.

Obstacle (ii) The dual bound is generally nonsmooth because the changes in column selected by pricing subproblems can abruptly change its slope. To obtain a differentiable learning objective, we need to smooth the dual bound.

Obstacle (iii) Even after smoothing, an explicit gradient step need not improve the dual bound because the pricing response changes as the price moves. We therefore require the displacement from y_t to agree with the gradient induced by the response at the resulting price.

2.1 The Lagrangian dual function of the normalized master problem

To address obstacle (i), we construct a Lagrangian dual function that remains finite even when some columns have negative reduced cost. Choose $U > 0$ so that adding $\sum_{x\in\mathcal{R}} w_x \le U$ preserves the optimal value P and the feasibility of every RMP. An empty column 0 with $c_0 = 0$ and $a_0 = 0$ absorbs the unused weight, and with $\mathcal{R}_0 = \mathcal{R} \cup \{0\}$, the normalized master problem (NMP) is

$$P = \min_{w\ge 0} \Big\{ \sum_{x\in\mathcal{R}_0} c_x w_x : \sum_{x\in\mathcal{R}_0} a_x w_x = b, \quad \sum_{x\in\mathcal{R}_0} w_x = U \Big\}. \tag{NMP}$$

We now relax the coupling constraints $\sum_x a_x w_x = b$ using dual prices q, while retaining nonnegativity and the weight constraint $\sum_x w_x = U$. The remaining feasible set is a compact scaled simplex, so its linear objective has a finite minimum for every q, including prices with negative reduced costs. The resulting Lagrangian dual function is (with its full derivations in Appendix A.1)

$$D(q) = b^\top q + U \min_{x\in\mathcal{R}_0} \big(c_x - q^\top a_x\big). \tag{1}$$

As the minimum of finitely many affine functions, D is finite, concave, and piecewise affine. It satisfies $D(q) \leq P$ for every q, and LP strong duality gives $\max_q D(q) = P$ (Appendix A.2). Pricing also provides first-order information: if x_q attains the minimum in Eq.(1), then $b - U a_{x_q}$ is a subgradient of D at q. However, this vector stays constant while the same column minimizes reduced cost and can jump when the minimizing column changes (Figure 2, A2), motivating the smoothing in the next subsection.

2.2 Dual gradients from perturbed pricing

To tackle obstacle (ii), we smooth D by averaging the optimal pricing value under random cost perturbations, following Berthet et al. (2020). Applying independent perturbations to individual columns would require a separate cost for each of exponentially many columns, which is impractical. We hence perturb the primitive costs used in (PP), such as arc costs in routing, without changing column feasibility. Because columns share primitives, their perturbed scores are generally dependent. Here we establish conditions under which these perturbations yield a differentiable dual bound whose gradient can be estimated without bias from exact perturbed pricing responses.

Formally, let $z_x \in \mathbb{R}^d$ be the primitive representation of column x, for instance the arcs a route uses or the number of pieces of each item in a cutting pattern, with $z_0 = 0$. For a random vector $\epsilon \in \mathbb{R}^d$ and a temperature $\tau > 0$, the perturbed solver return and the smoothed full dual ar

$$x(q, \epsilon) \in \operatorname*{arg\,min}_{x \in \mathcal{R}_0} \left\{ \bar{c}_x(q) - \tau z_x^\top \epsilon \right\}, \quad D_\tau(q) = b^\top q + U\, \mathbb{E}_\epsilon \Big[\min_{x \in \mathcal{R}_0} \left\{ \bar{c}_x(q) - \tau z_x^\top \epsilon \right\} \Big]. \tag{2}$$

Computing $x(q, \epsilon)$ changes only the arc costs without modifying the solver. The perturbations alter the scores of columns but never their feasibility, and columns that share arcs receive correlated shifts.

Proposition 1 (Gradient from perturbed dual)**.** *Assume that ϵ has zero mean, a density, and a finite first moment, and that vectors z_x, $x \in \mathcal{R}_0$, are pairwise distinct, so that no column shares the representation $z_0 = 0$ of the empty column, and that every perturbed pricing problem is solved exactly over $\mathcal{R}_0$. Then, for every q, the perturbed winner $x(q, \epsilon)$ is almost surely unique, D_τ is concave and differentiable, and*

$$\nabla D_\tau(q) = b - u_\tau(q), \qquad u_\tau(q) = U\, \mathbb{E}_\epsilon \big[a_{x(q,\epsilon)} \big]. \tag{3}$$

The proof is given in Appendix B.1. We call $u_\tau(q)$ the *mean coupling response*, the expected row activity obtained when the perturbed pricing winner carries weight U. The gradient $b - u_\tau(q)$ measures the mismatch between the required and expected row activity, and varies continuously with q (Figure 2, B2). Each exact perturbed pricing response therefore supplies an unbiased gradient estimate $b - U a_{x(q,\epsilon)}$. Smoothing preserves the lower bound $D_\tau(q) \leq D(q) \leq P$, with $D(q) - D_\tau(q) = O(\tau)$ (Appendix B.3).

2.3 Learning self-consistent dual prices

To address obstacle (iii), at any given RMP iteration t, we seek a self-consistent dual price (SCDP) $q_t^\star$ whose own pricing response accounts for the move from y_t. This self-consistency requires

$$q_t^\star - y_t = \eta \big[b - u_\tau(q_t^\star) \big] = \eta\, \nabla D_\tau(q_t^\star), \tag{4}$$

where $\eta > 0$ is a step size. Since $q_t^\star$ appears on both sides, (4) is a fixed-point condition. Rearranging it gives $(q_t^\star - y_t)/\eta - \nabla D_\tau(q_t^\star) = 0$, which is precisely the stationarity condition of the loss

$$\ell_t(q) = -D_\tau(q) + \frac{1}{2\eta} \left\| q - y_t \right\|^2, \qquad \nabla \ell_t(q) = \frac{1}{\eta} \Big[(q - y_t) - \eta \big(b - u_\tau(q) \big) \Big]. \tag{5}$$

Because D_τ is concave, ℓ_t is $(1/\eta)$-strongly convex, so its unique minimizer is the unique solution of (4). The SCDP is therefore the proximal point of $-D_\tau$ at y_t. Its closest methodological antecedent is proximal stabilization in CG (Amor et al., 2009), which regularizes a restricted-master model built from previously generated columns. In contrast, the SCDP proximal objective is defined by the smoothed full Lagrangian dual over the entire column family. By (4), the displacement from y_t is determined by the full-pool pricing response induced at the minimizer itself. Crucially, optimizing this full-pool objective does not require enumerating its columns, because perturbed pricing provides

unbiased stochastic estimates of its gradient directly from the full column family. This turns dual prediction from matching a precomputed reference dual into learning an implicit, state-dependent target directly from pricing feedback.

This structure is reflected directly in the gradient of (5), which compares the proposed move $q - y_t$ with the move $\eta(b - u_\tau(q))$ induced by the pricing response at q. It is precisely the self-consistency residual scaled by $1/\eta$. The residual vanishes at $q_t^\star$, and its norm controls the distance to this target (Appendix C.2). Concavity of D_τ further yields the following sufficient-ascent guarantee.

Theorem 1 (Sufficient ascent of SCDP)**.** *Suppose that D_τ is concave and differentiable, and let $\eta > 0$. For any RMP price y_t, the self-consistency condition* (4) *has a unique solution $q_t^\star$ satisfying*

$$D_\tau(q_t^\star) - D_\tau(y_t) \geq \frac{\|q_t^\star - y_t\|^2}{\eta}. \tag{6}$$

Consequently, the SCDP strictly improves D_τ unless y_t already maximizes D_τ, in which case $q_t^\star = y_t$.

The proof is given in Appendix C.1. For any nonzero exact SCDP step, the lower bound in (6) is strictly positive, so the theorem guarantees dual-bound improvement. With D_τ the lower bound on MP and the current RMP objective the primal upper bound, this improvement narrows the primal–dual gap at the current CG iteration, even when the RMP stalls.

Unsupervised learning from pricing feedback. The loss in (5) provides a direct objective for learning the SCDP. Consequently, we train a network parameterized by θ to predict SCDP $q_\theta(s_t)$ from the CG state s_t, comprising the instance, $\mathcal{R}_t$, and y_t. Using recorded CG states, we minimize

$$\min_\theta \ \mathbb{E}_{s_t}\big[\ell_t\big(q_\theta(s_t)\big)\big]. \tag{7}$$

Following Proposition 1, perturbed pricing at the predicted price supplies an unbiased estimate of the loss gradient. The network can therefore be trained from pricing feedback without computing $q_t^\star$ as a label or differentiating through the pricing solver. Learning this dependence offline therefore breaks the reciprocal dependence issue when deployed during CG iterations, allowing the network to predict an approximate SCDP before generating the columns it induces. Section 3 next develops efficient estimators of this training gradient.

3 Scalable Training with Unbiased Gradients

While perturbed pricing enables gradient-based training, a single PP response can yield a noisy gradient estimate. The variance can be effectively reduced by averaging over J independent responses, but direct computation requires J exact pricing solves per training state. Since pricing is NP-hard in many CG applications, these repeated PP solves can make training prohibitively expensive. We address this issue by exploiting the ability of a pricing search to return multiple feasible columns: we retain these columns in a pool and re-evaluate their costs under each perturbation. A single pool-building PP solve can then supply all J samples when the pool satisfies the certificate in Theorem 2, preserving unbiasedness while reducing variance. When the pool cannot be certified, only one additional exact perturbed pricing solve is needed to restore unbiasedness (Proposition 2). This replaces J separate pricing solves with one pool-building search and at most one additional solve, making SCDP applicable for broader CG applications with expensive pricing oracles.

Certified column pools for unbiased gradient estimates. Write $\bar{c}_x(q) = c_x - q^\top a_x$ for the reduced cost of column x at q. The perturbed winner $x(q, \epsilon)$ in (2) then minimizes $\bar{c}_x(q) - \tau z_x^\top \epsilon$ over $\mathcal{R}_0$. With independent perturbations $\epsilon_1, \ldots, \epsilon_J$, Proposition 1 gives the sample-average estimate

$$\hat{g} = \frac{q - y_t}{\eta} - b + \frac{U}{J}\sum_{j=1}^{J} a_{x(q,\epsilon_j)}, \qquad \mathbb{E}[\hat{g}] = \nabla \ell_t(q), \qquad \mathrm{Cov}(\hat{g}) = \tfrac{1}{J}\,\mathrm{Cov}\big(U a_{x(q,\epsilon)}\big). \tag{8}$$

We introduce the *certified replay technique*: one search can serve all J perturbations. Specifically, a certified enumeration (CE) lists columns in nondecreasing order of reduced cost at q, up to a depth limit L. It returns a pool $\mathcal{P} \subseteq \mathcal{R}_0$ containing x_q, a column of minimum reduced cost at q, together with a lower bound $\Delta \geq 0$ on how much worse every omitted column is:

$$\bar{c}_x(q) \geq \bar{c}_{x_q}(q) + \Delta \quad \text{for every } x \in \mathcal{R}_0 \setminus \mathcal{P}, \qquad \hat{x}(q, \epsilon) \in \operatorname*{arg\,min}_{x \in \mathcal{P}} \big\{\bar{c}_x(q) - \tau z_x^\top \epsilon\big\}. \tag{9}$$

The pool is fixed before drawing the perturbations. For each new perturbation, we re-price only the pooled columns and take the *pool winner* $\hat{x}(q, \epsilon)$ instead of solving (PP) again. We call this step a *replay*. Replacing $x(q, \epsilon_j)$ by $\hat{x}(q, \epsilon_j)$ in (8) gives the replay estimator $\hat{g}_{\mathcal{P}}$.

To certify exact replay, we must rule out any omitted column becoming the perturbed winner. By (9), every omitted column starts at least Δ behind x_q, and a perturbation can reduce this gap by at most $\tau \left\| z_x - z_{x_q} \right\|_1 \|\epsilon\|_\infty$. If Δ exceeds this quantity for every omitted column, each remains worse than $x_q \in \mathcal{P}$, so the perturbed winner must lie in the pool. This observation gives the following sufficient condition for exact replay and the proof is supplied in Appendix D.1.

Theorem 2 (Exact pool replay)**.** *Under the assumptions of Proposition 1, suppose that* $\|\epsilon\|_\infty \leq 1$ *almost surely. Let* $\mathcal{P}$ *and* Δ *be as in* (9)*, and let* $R \geq 0$ *be a known bound satisfying* $\left\| z_x - z_{x_q} \right\|_1 \leq R$ *for every* $x \in \mathcal{R}_0 \setminus \mathcal{P}$*. If* $\Delta > \tau R$*, then* $\hat{x}(q, \epsilon) = x(q, \epsilon)$ *almost surely.*

For bounded perturbations, the normalization $\|\epsilon\|_\infty \leq 1$ can be imposed by absorbing their scale into τ, and a uniform noise on $[-1, 1]^d$ satisfies it directly.

Unbiased correction for uncertified column pools. On large instances, CE may be too expensive. In vehicle routing, for example, (PP) is accelerated by imposing *exact dominance* (ED) or *heuristic dominance* (HD) rules. ED retains optimal solution of PP at q but does not guarantee a global top-L list, and HD may eliminate the optimal solution. Neither guarantees that the pool contains the perturbed winner $x(q, \epsilon)$, so replay may be biased. Fortunately, we can still correct this bias with only one additional exact perturbed pricing solve per training state. More generally, this correction applies to any nonempty pool $\mathcal{P} \subseteq \mathcal{R}_0$ fixed before drawing the J independent perturbations.

Write $C_j = a_{\hat{x}(q,\epsilon_j)}$ and $F_j = a_{x(q,\epsilon_j)}$. Choose I uniformly from $\{1, \ldots, J\}$, independently of the perturbations, solve the perturbed pricing problem exactly at (q, ϵ_I), and update the gradient:

$$\widehat{g}_{\mathrm{corr}} = \frac{q - y_t}{\eta} - b + U \left(\frac{1}{J} \sum_{j=1}^{J} C_j + F_I - C_I \right). \tag{10}$$

Proposition 2 (Unbiased correction of pool replay)**.** *Under the assumptions of Proposition 1 and the uniform sampling procedure above, the corrected estimator* $\widehat{g}_{\mathrm{corr}}$ *is unbiased.*

The proof can be found in Appendix D.4, and the detailed training and deployment procedures for learning SCDP with pool replay are given in Algorithm 1 in Appendix E.

4 Experiments

We evaluate the learned SCDP on three classical applications of CG: the capacitated vehicle routing problem (CVRP), the CVRP with time window constraints (CVRPTW), and the one-dimensional cutting stock problem (CSP). These problems test SCDP across different pricing structures, with resource-constrained route search in CVRP and CVRPTW and bounded knapsack in CSP.

Baseline. We compare SCDP with standard CG, four dual-stabilization methods, and two learning-based methods. Stabilization baselines comprise minimum-norm dual stabilization (`QP`), which prices at the minimum-norm point of the RMP's optimal dual face, interior point stabilization (`IPS`) (Rousseau et al., 2007), automated LP-based stabilization (`ACLPS`) (Pessoa et al., 2018), and primal-dual column generation (`PDCGM`) (Gondzio et al., 2016). Learning-based baselines are `SPARSE` (Kraul et al., 2023) and `ASCG` (Shen et al., 2024). All learned methods use the same training instances and are trained until their loss stabilizes. We summarize the details of the implementations, architecture, and hyperparameters in Appendix E.

Instances. CVRP training states come from 35-customer instances generated following Queiroga et al. (2021), and testing uses the 20 X-benchmark instances with 100–200 customers (Uchoa et al., 2017). For CVRPTW, we add time windows to the training generator and test on 116 instances: 56 Solomon instances (Solomon, 1987) and 60 instances from the 200-customer Gehring–Homberger set (Homberger & Gehring, 1999), retaining only the first 100 customer nodes of each latter instance. CSP training uses CUTGEN1 (Gau & Wäscher, 1995), and testing uses 45 BPPLIB instances from AI/ANI, SSTriples, and GI (Delorme et al., 2018), families accounting for over 95% of standard CG's

runtime on the library. Test instances are excluded from training, and routing models trained at $n = 35$ are deployed without retraining, so all results are reflective of out-of-distribution performances.

Solvers for pricing subproblems. We use the state-of-the-art solvers for our PP. RouteOpt (You & Yang, 2026) solves routing problems with exact-dominance labeling (ED) at $K = 1000$. CSP uses bounded-knapsack pricing via Gurobi 12.0.2 (Gurobi Optimization, LLC, 2026) with $K = 50$. All methods share the same PP solver and column cap within each comparison.

4.1 Main results

Table 1: CG performance. Competitor denotes the fastest non-SCDP baseline by total time: SPARSE for CVRP and CVRPTW, and PDCGM for CSP. The complete results can be found in the Appendix E.4.

	Per-instance runtime (s): mean (std)			Total reduction (competitor / SCDP)			
Benchmark (K)	Standard CG	Competitor	SCDP	CG time	RMP	PP	RMP iter.
CVRP (1000)	97.4 (217.3)	51.6 (116.9)	**35.8** (81.4)	48% / **63%**	65% / **85%**	28% / **42%**	49% / **69%**
CVRPTW (1000)	124.6 (512.8)	8.1 (12.8)	**7.8** (12.6)	**94%** / **94%**	**98%** / **98%**	76% / **78%**	69% / **73%**
CSP (50)	4.0 (4.0)	3.2 (3.2)	**2.1** (2.6)	18% / **47%**	28% / **52%**	16% / **31%**	22% / **49%**

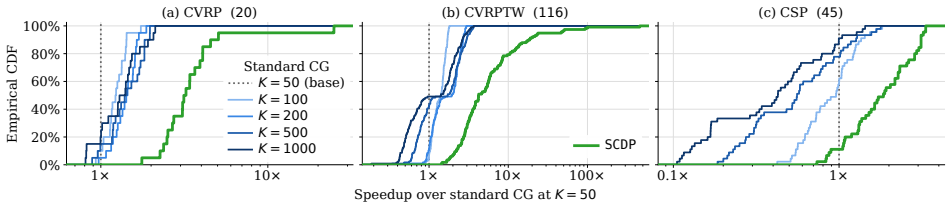


Figure 3: Distribution of per-instance CG runtimes on (a) CVRP, (b) CVRPTW, and (c) CSP. From (a) to (c), SCDP uses $K = 1000, 1000, 50$, respectively. Parentheses give test-set sizes.

SCDP reduces total CG time by 63%, 94%, and 47% on CVRP, CVRPTW, and CSP, and its mean runtime is lower than that of the strongest baseline on each benchmark (Table 1). Figure 3 shows broader runtime improvements on three benchmarks compared to standard CG with multiple K configurations. These results demonstrate strong adaptability of SCDP in two respects: the method across different problem types and the pretrained model across different problem scales.

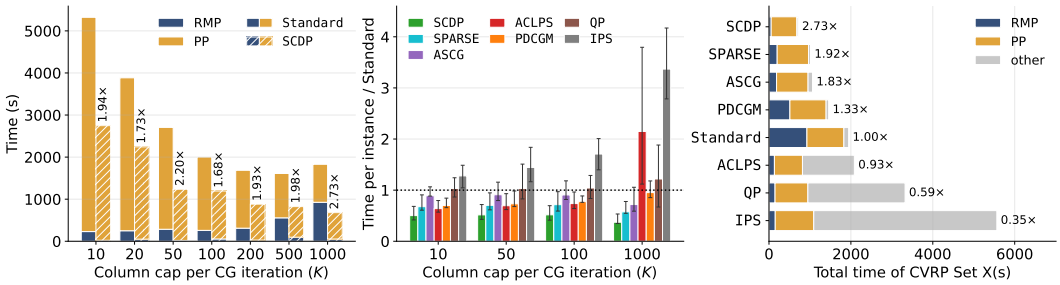


Figure 4: Effect of the column cap on CVRP with ED pricing. **Left:** Total RMP and PP time as K varies, with SCDP speedups above each pair. **Middle:** Per-instance runtime relative to standard CG. **Right:** Total CG time at $K = 1000$, with speedups over standard CG.

SCDP remains effective when each pricing call admits many columns (Figure 4). Increasing K reduces standard CG's pricing time but increases its RMP cost at large caps. SCDP reduces both components and achieves its largest speedup at $K = 1000$, motivating this cap for the routing experiments. At this setting, its total speedup is 2.73×, compared with 1.92× for SPARSE. The stabilization baselines QP, ACLPS, and IPS take longer than standard CG. The reason is that

stabilization widens the variety of returned columns, which is also what a larger column cap does directly. The largest runtime reduction occurs on CVRPTW, where standard CG spends 80% of its time solving the RMP. `SCDP` reduces RMP time by 98% and PP time by 78%, yielding a $15.95\times$ total speedup. `SPARSE` reaches $15.42\times$ (Figure 5, left). CVRPTW appears to be predominantly stabilization-limited: its highly degenerate CG trajectory leaves substantial room for any informative future-dual prediction to suppress redundant dual movement, making `SPARSE` highly effective.

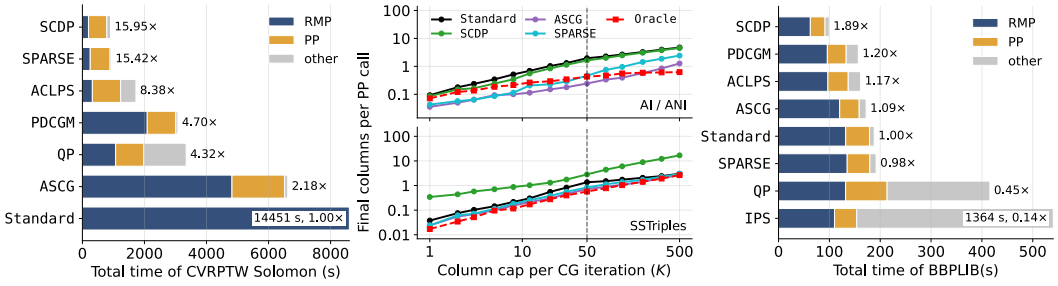


Figure 5: Runtime and column usefulness on CVRPTW and CSP. **Left:** CVRPTW runtime at $K = 1000$. **Middle:** Effect of the search price and cap on columns retained in the final LP solution per pricing call, where `Oracle` uses an optimal Lagrange dual. **Right:** CSP runtime at $K = 50$.

An optimal full dual does not necessarily generate the most useful columns for the current RMP. On AI/ANI and SSTriples, `Oracle` generates fewer columns retained in the final LP solution per pricing call than standard CG (Figure 5, middle). SCDP approximately matches standard CG on AI/ANI and exceeds all compared methods on SSTriples. This supports using the current state and the full-pool response to select search prices in the proximal region of the current dual, as in Eq.(5). Across the three CSP families, SCDP achieves a $1.89\times$ speedup, compared with $1.20\times$ for `PDCGM` and $0.98\times$ for `SPARSE`.

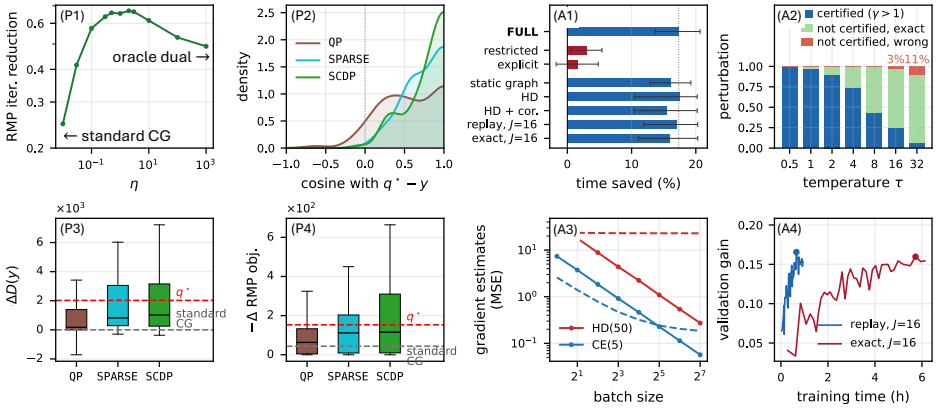


Figure 6: SCDP property analyses (**P**) and ablation studies (**A**) on 35-customer CVRP instances. **(P1)** Effect of η in the training loss on CG iteration reduction. **(P2)** Cosine similarity between $q_t - y_t$ and $q_t^\star - y_t$. **(P3)** Lagrange dual improvement. **(P4)** RMP objective decreases after adding columns and re-solving. **(A1)** Effect of the full dual objective, endpoint consistency, graph representation, and gradient-estimation variants on CG time saved. **(A2)** Coverage of the replay certificate as τ increases. **(A3)** Effect of gradient correction on estimation error averaging batch size (dashed: uncorrected, solid: corrected). **(A4)** compares replay with exact perturb-and-MAP at $J = 16$.

SCDP property analyses. Learned SCDP displacements align more closely with $q_t^\star - y_t$ than those of `QP` and `SPARSE`, with cosine similarity concentrated near one (Figure 6, P2). Theorem 1 guarantees ascent on D_τ at the exact SCDP. For learned prices, Eq.(12) relates this guarantee to the self-consistency residual. Empirically, `SCDP` yields positive median gains in the unsmoothed Lagrange-dual bound D (P3) and produces larger median RMP objective decreases than standard CG

after one round of column addition (P4). These results support using the Lagrange-pool response to guide column discovery and make practical progress toward the primal optimum. Intermediate η gives the largest iteration reductions (P1): small values constrain prices near y_t, while large values weaken the dependence on the current RMP.

Ablation. The full-pool objective and endpoint consistency account for most of the acceleration (Figure 6, A1). Restricting the objective to the RMP leaves y_t optimal, removing the incentive to change the search price while an explicit step evaluates the response at y_t instead of its endpoint. Both ablations sharply reduce runtime savings. In contrast, using the same neural network as SPARSE preserves most of the savings, with error bars overlapping those of the RMP bipartite representation. This supports attributing SCDP's advantage primarily to its training framework and objective. The replay certificate is sufficient but conservative (A2): coverage decreases with τ, yet most uncertified replays remain exact. The HD and correction variants yield similar runtime savings (A1), because the gradient correction is unbiased and errors continue decreasing with batch size. Likewise, pool replay and exact perturbed pricing at $J = 16$ yield similar CG savings. A4 shows that replay reaches comparable validation performance to perturb-and-MAP in substantially less training time.

5 Related Work

Dual stabilization and learning-augmented column generation. Dual stabilization modifies the prices used for pricing when the RMP dual is an unreliable guide to the full master, as can occur under degeneracy. Box and penalty methods restrict or penalize deviations from a stability center (Du Merle et al., 1999; Amor et al., 2009), while smoothing methods query convex combinations of the RMP dual price and a center (Wentges, 1997; Neame, 2000). Stabilization parameters can also be adjusted automatically (Pessoa et al., 2018). More recently, learning has been used throughout CG, including dual prediction for stabilization or warm starts (Kraul et al., 2023; Shen et al., 2024; Sugishita et al., 2024), pricing-bound prediction (Václavík et al., 2018), learned pricing heuristics (Shen et al., 2022), and column selection (Morabit et al., 2021; Chi et al., 2022; Yuan et al., 2024). These methods either learn from the restricted master or act on generated columns, whereas we predict dual prices directly through the response of the full column family.

Learning from discrete pricing responses. Defining the dual prices through full-family pricing responses makes the learning target implicit rather than directly available as a supervised label. Perturbed optimizers provide differentiable signals from discrete solvers by randomizing their costs and differentiating expected responses (Berthet et al., 2020), with related formulations based on convex regularization, Fenchel–Young losses, and shifted-cost optimization (Blondel et al., 2020; Vlastelica et al., 2020; Niepert et al., 2021; Paulus et al., 2020). In CG, however, perturbing individual column scores is impractical because the column family is exponentially large. We instead perturb primitive costs, allowing the pricing solver to provide gradient over the full column family.

Reusing pricing computation across perturbations. Estimating gradients from pricing responses requires solving the pricing problem under multiple perturbations, which can be costly for combinatorial pricing. Classical CG reuses search through multiple-column pricing and column pools (Lübbecke & Desrosiers, 2005), while labeling and exact dominance retain optimal columns for a nominal pricing objective (Irnich & Desaulniers, 2005; Righini & Salani, 2006). These mechanisms, however, do not guarantee validity across perturbed objectives. Our certified replay identifies when a single pricing pool exactly recovers the responses to multiple perturbations, avoiding repeated combinatorial searches.

6 Conclusion

Pricing responses provide a useful learning signal even when the RMP objective stalls. Across routing and cutting stock, using this signal to learn self-consistent dual prices reduces both CG iterations and runtime. The ablations suggest that the main gains come from the Lagrange dual objective and self consistency condition, with limited sensitivity to the graph representation. Replay achieves comparable training outcomes at substantially lower computational cost. Future work should examine how these gains translate to complete branch-and-price search, where pricing problems evolve along the search tree

AI USE STATEMENT

In this work, we used generative AI tools to assist in implementing methods (coding assistance) and to assist in the writing of proofs, limited to checking the correctness of proofs that were derived and written by the authors. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, or interpret results, and cleaning and reformatting datasets and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to edit the paper to improve readability, including polishing the language and correcting grammar. We have manually reviewed all language edits suggested by the LLM to ensure that they did not alter the meaning of the text. AI-assisted code was reviewed by the authors and validated through testing and by checking that the reported results are reproducible. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our research, we open-source our code at `https://anonymous.4open.science/r/SCDP_CG-FC18/README.md`. A detailed README.md file offers step-by-step instructions for setting up the computational environment and executing the experiments described in this paper. Our full evaluation methodology, experimental setup, datasets, and computational environment are described in Appendix E.

REFERENCES

Hatem MT Ben Amor, Jacques Desrosiers, and Antonio Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009.

Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

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

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

Cheng Chi, Amine Aboussalah, Elias Khalil, Juyoung Wang, and Zoha Sherkat-Masoumi. A deep reinforcement learning framework for column generation. *Advances in Neural Information Processing Systems*, 35:9633–9644, 2022.

George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

Maxence Delorme, Manuel Iori, and Silvano Martello. Bpplib: a library for bin packing and cutting stock problems. *Optimization Letters*, 12(2):235–250, 2018.

Jacques Desrosiers and Marco E Lübbecke. A primer in column generation. In *Column generation*, pp. 1–32. Springer, 2005.

Olivier Du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

Issmail Elhallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. Dynamic aggregation of set-partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005.

T Gau and G Wäscher. Cutgen1: A problem generator for the standard one-dimensional cutting stock problem. *European journal of operational research*, 84(3):572–579, 1995.

Paul C Gilmore and Ralph E Gomory. A linear programming approach to the cutting-stock problem. *Operations research*, 9(6):849–859, 1961.

Jacek Gondzio, Pablo González-Brevis, and Pedro Munari. Large-scale optimization with the primal-dual column generation method. *Mathematical Programming Computation*, 8(1):47–82, 2016.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL `https://www.gurobi.com`.

Jörg Homberger and Hermann Gehring. Two evolutionary metaheuristics for the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 37(3): 297–318, 1999.

Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In *Column generation*, pp. 33–65. Springer, 2005.

Sebastian Kraul, Markus Seizinger, and Jens O Brunner. Machine learning–supported prediction of dual variables for the cutting stock problem with an application in stabilized column generation. *INFORMS Journal on Computing*, 35(3):692–709, 2023.

Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

Mouad Morabit, Guy Desaulniers, and Andrea Lodi. Machine-learning–based column selection for column generation. *Transportation Science*, 55(4):815–831, 2021.

Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

Philip James Neame. *Nonsmooth dual methods in integer programming*. PhD thesis, University of Melbourne, Department of Mathematics and Statistics, 2000.

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

Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris Maddison. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems*, 33:5691–5704, 2020.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

Eduardo Queiroga, Ruslan Sadykov, Eduardo Uchoa, and Thibaut Vidal. 10,000 optimal CVRP solutions for testing machine learning based heuristics. In *AAAI-22 workshop on machine learning for operations research (ML4OR)*, 2021.

Giovanni Righini and Matteo Salani. Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3): 255–273, 2006.

R Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Louis-Martin Rousseau, Michel Gendreau, and Dominique Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.

Yunzhuang Shen, Yuan Sun, Xiaodong Li, Andrew Eberhard, and Andreas Ernst. Enhancing column generation by a machine-learning-based pricing heuristic for graph coloring. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 9926–9934, 2022.

Yunzhuang Shen, Yuan Sun, Xiaodong Li, Zhiguang Cao, Andrew Eberhard, and Guangquan Zhang. Adaptive stabilization based on machine learning for column generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 44741–44758, 2024.

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

Nagisa Sugishita, Andreas Grothey, and Ken McKinnon. Use of machine learning models to warmstart column generation for unit commitment. *INFORMS Journal on Computing*, 36(4):1129–1146, 2024.

Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Roman Václavík, Antonín Novák, Přemysl Šůcha, and Zdeněk Hanzálek. Accelerating the branch-and-price algorithm using machine learning. *European Journal of Operational Research*, 271(3): 1055–1069, 2018.

Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

Paul Wentges. Weighted dantzig-wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162, 1997.

Zhengzhong You and Yu Yang. Routeopt: An open-source modular exact solver for vehicle routing problems. *INFORMS Journal on Computing*, 2026.

Haofeng Yuan, Lichang Fang, and Shiji Song. A reinforcement-learning-based multiple-column selection strategy for column generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8209–8216, 2024.

A THE NORMALIZED MASTER AND THE FULL DUAL

This appendix supports Section 2.1. It states what the weight bound U has to satisfy, proves the properties of the full dual D used there, and gives two settings in which a valid bound is easy to find.

A.1 DERIVATION OF THE LAGRANGE DUAL FUNCTION

Relaxing the coupling constraints of (NMP) using dual prices q, while retaining nonnegativity and the weight constraint, gives

$$D(q) = b^\top q + \min_{\substack{w \geq 0 \\ \sum_{x \in \mathcal{R}_0} w_x = U}} \sum_{x \in \mathcal{R}_0} (c_x - q^\top a_x) w_x.$$

For fixed q, the inner problem minimizes a linear function over a scaled simplex. A minimizer assigns all weight U to a column of minimum reduced cost, yielding (1). Since $\mathcal{R}_0$ is finite, D is the minimum of finitely many affine functions of q and is therefore finite, concave, and piecewise affine

A.2 A VALID WEIGHT BOUND

Section 2.1 asks for a bound $U > 0$ with two properties:

(a) adding $\sum_{x \in \mathcal{R}} w_x \leq U$ to (MP) leaves the optimal value P unchanged;
(b) every RMP solved by the algorithm stays feasible under the same constraint.

Property (a) is what makes D a valid bound, and it is all that Section 2 uses. Property (b) matters only if CG is run on the normalized master, that is, on the RMP with the empty column and the weight row added.

Proposition A.1 (Normalized master and full dual)**.** *Let U satisfy (a). Then:*

(i) setting $w_0 = U - \sum_{x \in \mathcal{R}} w_x$ maps the feasible solutions of (MP) *that obey $\sum_{x \in \mathcal{R}} w_x \leq U$ one-to-one onto the feasible solutions of* (NMP)*, keeps every objective value, and makes the optimal value of* (NMP) *equal to P;*

(ii) D is finite, concave, and piecewise affine, and $b - U a_{x_q}$ is a subgradient of D at q whenever x_q attains the minimum in (1)*;*

(iii) $\max_q D(q) = P$. In particular $D(q) \leq P$ at every price, so one pricing call returns a valid lower bound on P.

Proof. (i) Take $w \geq 0$ with $\sum_{x \in \mathcal{R}} a_x w_x = b$ and $\sum_{x \in \mathcal{R}} w_x \leq U$, and set $w_0 = U - \sum_{x \in \mathcal{R}} w_x \geq 0$. The empty column has $a_0 = 0$ and $c_0 = 0$, so the coupling rows and the objective are unchanged, while the weight row now holds with equality. Going back, dropping w_0 from a feasible solution of (NMP) leaves a solution of (MP) with $\sum_{x \in \mathcal{R}} w_x = U - w_0 \leq U$ and the same objective value. The two maps invert each other, so the optimal value of (NMP) is the optimal value of (MP) under the extra bound, which is P by (a).

(ii) Each column contributes an affine function $q \mapsto b^\top q + U(c_x - q^\top a_x)$, and D is the smallest of finitely many of them, hence finite, concave, and piecewise affine. If x_q attains the minimum at q, then for any q',

$$D(q') \;\leq\; b^\top q' + U\big(c_{x_q} - q'^\top a_{x_q}\big) \;=\; D(q) + \big(b - U a_{x_q}\big)^\top (q' - q),$$

which is the subgradient inequality for the concave function D.

(iii) Given the coupling rows of (NMP) the multipliers q and the weight row the multiplier ν, the LP dual is

$$\max_{q,\nu} \Big\{ b^\top q + U\nu :\; c_x - q^\top a_x \geq \nu \;\text{ for all } x \in \mathcal{R}_0 \Big\}.$$

For a fixed q, the best choice is $\nu = \min_{x \in \mathcal{R}_0}(c_x - q^\top a_x)$, and substituting it turns the dual into $\max_q D(q)$. By (i) the problem (NMP) is feasible with finite optimal value P, so strong duality gives $\max_q D(q) = P$. □

Nothing in the proof uses the full family: the same statement holds for any subfamily $\mathcal{S} \subseteq \mathcal{R}$, with P replaced by the optimal value of the master over $\mathcal{S}$. This is what property (b) is about. If the bound is strictly redundant, meaning that every feasible solution of the RMP satisfies $\sum_x w_x < U$, then the empty column always carries weight, complementary slackness makes the dual of the weight row zero, and the two restricted masters have the same dual prices. CG then produces the same y_t whichever of the two it solves.

A.3 Two settings with an easy bound

Set-partitioning masters. Let $a_x \in \{0,1\}^m$, let every column cover at least one row, and let $b = \mathbf{1}$. Every feasible w, for the full master and for any restricted master, satisfies

$$\sum_x w_x \;\le\; \sum_x \big(\mathbf{1}^\top a_x\big) w_x \;=\; \mathbf{1}^\top b \;=\; m.$$

Any $U \ge m$ therefore has both properties, and any $U > m$ is strictly redundant, so the remark above applies at every iteration.

Masters with positive costs. Let $c_x \ge c_{\min} > 0$ for every column, and let $\bar{P}$ be the optimal value of the initial RMP. A solution with objective value v has $\sum_x w_x \le v/c_{\min}$. RMP values do not increase as columns are added, so every RMP, and (MP) itself, has an optimal solution with $\sum_x w_x \le \bar{P}/c_{\min}$. Any $U \ge \bar{P}/c_{\min}$ therefore has both properties. Such a bound need not be strictly redundant, and for other masters a valid bound has to be checked separately.

B Perturbed Pricing

Throughout, $\bar{c}_x(q) = c_x - q^\top a_x$ is the reduced cost of column x at q, and

$$\phi(q,\epsilon) = \min_{x \in \mathcal{R}_0} \big\{\bar{c}_x(q) - \tau z_x^\top \epsilon\big\}, \qquad \text{so that} \qquad D_\tau(q) = b^\top q + U\,\mathbb{E}_\epsilon\big[\phi(q,\epsilon)\big].$$

We write $A_{\max} = \max_{x \in \mathcal{R}_0} \|a_x\|_2$ for the largest norm of a coefficient vector. Beyond the assumptions listed in Proposition 1, we assume that ϵ has a density and a finite first moment. The uniform law on $[-1,1]^d$ used in the experiments has both.

B.1 Proof of Proposition 1

Proof. Fix q. Two distinct columns $x \ne x'$ tie when

$$\tau\,(z_x - z_{x'})^\top \epsilon = \bar{c}_x(q) - \bar{c}_{x'}(q).$$

The representations are distinct, so the left-hand side is a nonzero linear function of ϵ and the tie happens only on a hyperplane in $\mathbb{R}^d$. A hyperplane has probability zero under a density, and there are finitely many pairs of columns, so with probability one no two columns tie. The same argument applies inside any subfamily, which is what makes the pool winner unique as well.

For each ϵ, the map $q \mapsto \phi(q,\epsilon)$ is the smallest of finitely many affine functions, hence concave, and $|\phi(q,\epsilon)| \le \max_x |\bar{c}_x(q)| + \tau \max_x \|z_x\|_2 \, \|\epsilon\|_2$, which is integrable. An expectation of concave functions is concave.

Every $\phi(\cdot,\epsilon)$ is Lipschitz with constant $A_{\max}$, because the affine pieces have slopes $-a_x$. When the winner at (q,ϵ) is unique it stays the winner in a neighbourhood of q, where $\phi(\cdot,\epsilon)$ is affine with gradient $-a_{x(q,\epsilon)}$. Fix a direction h. As $s \downarrow 0$ the quotients $[\phi(q+sh,\epsilon) - \phi(q,\epsilon)]/s$ converge almost surely to $-a_{x(q,\epsilon)}^\top h$ and never exceed $A_{\max}\|h\|_2$, so dominated convergence gives the directional derivative $\big(b - U\,\mathbb{E}[a_{x(q,\epsilon)}]\big)^\top h$ for D_τ at q. This is linear in h, and a finite concave function whose directional derivatives are linear at a point is differentiable there. Hence $\nabla D_\tau(q) = b - u_\tau(q)$. □

A differentiable concave function on $\mathbb{R}^m$ has a continuous gradient, so the mean coupling response $u_\tau = b - \nabla D_\tau$ is continuous in q, as Figure 2, B2 draws.

B.2 THE SAMPLE-AVERAGE ESTIMATE

Let $\epsilon_1, \dots, \epsilon_J$ be independent copies of ϵ. The vectors $Ua_{x(q,\epsilon_j)}$ are then independent, have mean $u_\tau(q)$ by Proposition 1, and are bounded in norm by $UA_{\max}$. For the estimate $\hat{g}$ of (8) this gives

$$\mathbb{E}[\hat{g}] = \frac{q - y_t}{\eta} - b + u_\tau(q) = \nabla \ell_t(q), \qquad \mathrm{Cov}(\hat{g}) = \frac{1}{J}\mathrm{Cov}\big(Ua_{x(q,\epsilon)}\big),$$
$$\mathbb{E}\,\|\hat{g} - \nabla \ell_t(q)\|_2^2 \le \frac{U^2 A_{\max}^2}{J}.$$

When an update is formed, the recorded state s_t and the parameters θ are fixed, and no gradient runs through the process that produced the state. The price $q = q_\theta(s_t)$ and the Jacobian $\partial q_\theta(s_t)/\partial\theta$ are then deterministic, so

$$\mathbb{E}\Big[\big(\partial q_\theta(s_t)/\partial\theta\big)^\top \hat{g}\Big] = \big(\partial q_\theta(s_t)/\partial\theta\big)^\top \nabla \ell_t\big(q_\theta(s_t)\big) = \nabla_\theta\, \ell_t\big(q_\theta(s_t)\big),$$

and averaging over the finitely many recorded states gives an unbiased estimate of the gradient of the training objective ℓ_t. The same holds for any other unbiased estimate of $\nabla \ell_t(q)$, in particular for the replay estimate of Theorem 2 and the corrected estimate of Proposition 2.

B.3 WHAT SMOOTHING COSTS

Write $W = \mathbb{E}\big[\max_{x \in \mathcal{R}_0} z_x^\top \epsilon\big]$ for the width of the perturbation over the column family. Since the empty column has $z_0 = 0$, $W \ge 0$, and W is finite because ϵ is integrable. Under the normalization $\|\epsilon\|_\infty \le 1$, we have $W \le \max_{x \in \mathcal{R}} \|z_x\|_1$.

Proposition B.1 (Smoothing error)**.** *For every price q,*

$$D(q) - \tau UW \;\le\; D_\tau(q) \;\le\; D(q),$$

so $D_\tau(q) \le P$ as well, and $P - \tau UW \le \sup_q D_\tau(q) \le P$.

Proof. Fix q and let x_q minimize $\bar{c}_x(q)$ over $\mathcal{R}_0$. For the upper bound, $\phi(q,\epsilon) \le \bar{c}_{x_q}(q) - \tau z_{x_q}^\top \epsilon$ for every ϵ, and ϵ has zero mean, so $\mathbb{E}\phi(q,\epsilon) \le \bar{c}_{x_q}(q)$. For the lower bound, $\phi(q,\epsilon) \ge \bar{c}_{x_q}(q) - \tau \max_x z_x^\top \epsilon$, and taking expectations subtracts τW. Multiplying by U and adding $b^\top q$ gives the two inequalities, and Proposition A.1(iii) turns them into the statement about P. □

Every value $D_\tau(q)$ is therefore still a lower bound on P, and maximizing D_τ instead of D gives away at most τUW. This is the cost proportional to τ.

C SELF-CONSISTENT DUAL PRICES

This appendix proves Theorem 1 and the two statements made there about a learned price, which meets the self-consistency condition only approximately. Throughout, D_τ is concave and differentiable, $\eta > 0$, and ℓ_t is the loss of Eq.(5), so that

$$\nabla \ell_t(q) = \frac{q - y_t}{\eta} - \nabla D_\tau(q).$$

Concavity gives the inequality used in both proofs,

$$D_\tau(q') \;\le\; D_\tau(q) + \nabla D_\tau(q)^\top (q' - q) \qquad \text{for all } q, q' \in \mathbb{R}^m. \tag{11}$$

C.1 PROOF OF THEOREM 1

Proof. The loss ℓ_t adds the convex function $-D_\tau$ to the $(1/\eta)$-strongly convex quadratic $\|q - y_t\|^2/(2\eta)$, so it is $(1/\eta)$-strongly convex, finite, and coercive, and it has exactly one minimizer. It is also convex and differentiable, so a price minimizes ℓ_t exactly when $\nabla \ell_t(q) = 0$, which is the self-consistency condition (4). That condition therefore has the single solution $q_t^\star$.

Apply (11) at $q = q_t^\star$ and $q' = y_t$. Self-consistency gives $\nabla D_\tau(q_t^\star) = (q_t^\star - y_t)/\eta$, so

$$D_\tau(y_t) \;\le\; D_\tau(q_t^\star) + \frac{1}{\eta}(q_t^\star - y_t)^\top (y_t - q_t^\star) \;=\; D_\tau(q_t^\star) - \frac{\|q_t^\star - y_t\|^2}{\eta},$$

which is (6). If $q_t^\star = y_t$ then (4) gives $\nabla D_\tau(y_t) = 0$; conversely, if $\nabla D_\tau(y_t) = 0$ then y_t solves (4) and is the only solution. For a differentiable concave function, $\nabla D_\tau(y_t) = 0$ says exactly that y_t maximizes D_τ. So the ascent in (6) is strictly positive unless y_t already maximizes D_τ. □

The SCDP is the proximal point of $-D_\tau$ at y_t with parameter η (Moreau, 1965; Rockafellar, 1976). Being the maximizer of $D_\tau(q) - \|q - y_t\|^2/(2\eta)$ already gives $D_\tau(q_t^\star) - D_\tau(y_t) \ge \|q_t^\star - y_t\|^2/(2\eta)$, using the gradient at $q_t^\star$ doubles it.

The ascent also carries over to the unsmoothed bound D. Theorem 1 with Proposition B.1 gives

$$D(q_t^\star) \;\ge\; D_\tau(q_t^\star) \;\ge\; D_\tau(y_t) + \frac{\|q_t^\star - y_t\|^2}{\eta} \;\ge\; D(y_t) + \frac{\|q_t^\star - y_t\|^2}{\eta} - \tau UW,$$

so the SCDP improves the Lagrangian bound whenever it moves far enough, $\|q_t^\star - y_t\|^2 > \eta\tau UW$.

C.2 Learned prices

A learned price q_θ solves (4) only approximately, and the residual $\nabla \ell_t(q_\theta)$ controls both how far it is from $q_t^\star$ and how much improvement it achieves.

Proposition C.1 (Inexact self-consistency)**.** *For every price q,*

(i) $\|q - q_t^\star\| \le \eta\, \|\nabla \ell_t(q)\|$;

(ii) $D_\tau(q) - D_\tau(y_t) \;\ge\; \dfrac{\|q - y_t\|^2}{\eta} - \nabla \ell_t(q)^\top (q - y_t) \;\ge\; \|q - y_t\| \Big(\dfrac{\|q - y_t\|}{\eta} - \|\nabla \ell_t(q)\| \Big)$.

Proof. (i) Strong convexity of ℓ_t with modulus $1/\eta$ gives $\big(\nabla \ell_t(q) - \nabla \ell_t(q_t^\star)\big)^\top (q - q_t^\star) \ge \|q - q_t^\star\|^2/\eta$. The residual vanishes at $q_t^\star$, so the Cauchy–Schwarz inequality leaves $\|\nabla \ell_t(q)\|\, \|q - q_t^\star\| \ge \|q - q_t^\star\|^2/\eta$, which is the claim when $q \ne q_t^\star$ and is trivial otherwise.

(ii) By the definition of the loss, $\nabla D_\tau(q) = (q - y_t)/\eta - \nabla \ell_t(q)$. Putting this into (11) with $q' = y_t$ gives

$$D_\tau(y_t) \;\le\; D_\tau(q) - \frac{\|q - y_t\|^2}{\eta} + \nabla \ell_t(q)^\top (q - y_t),$$

which rearranges to the first inequality; Cauchy–Schwarz gives the second. □

With $q = q_\theta$, item (ii) becomes

$$D_\tau(q_\theta) - D_\tau(y_t) \ge \frac{\|q_\theta - y_t\|^2}{\eta} - \nabla \ell_t(q_\theta)^\top (q_\theta - y_t). \tag{12}$$

The second form shows that $\eta\, \|\nabla \ell_t(q_\theta)\| < \|q_\theta - y_t\|$ is enough for strict ascent. Item (i) is the distance bound. At $q = q_t^\star$, the residual vanishes and (ii) becomes (6).

In a run, $\nabla \ell_t(q)$ is known only through an estimate such as $\hat{g}$ in (8). By Jensen's inequality, $\mathbb{E}\,\|\hat{g}\| \ge \|\nabla \ell_t(q)\|$, so a sample residual overstates the true one on average. Checking the ascent condition from a single sample would need a variance estimate as well.

D Pool Replay

This appendix proves Theorem 2, explains how the two quantities in its condition are obtained during a search, and proves Proposition 2 together with the covariance that the correction pays for.

Throughout, the price q and the RMP dual price y_t are fixed, we abbreviate $\bar{c}_x = \bar{c}_x(q)$, and the pool $\mathcal{P} \subseteq \mathcal{R}_0$ is fixed before any perturbation is drawn. For a perturbation ϵ, the two winners are

$$x(\epsilon) \in \arg\min_{x \in \mathcal{R}_0} \left\{ \bar{c}_x - \tau z_x^\top \epsilon \right\} \qquad \text{and} \qquad \hat{x}(\epsilon) \in \arg\min_{x \in \mathcal{P}} \left\{ \bar{c}_x - \tau z_x^\top \epsilon \right\},$$

and both are unique with probability one (Appendix B.1). We keep the normalization $\|\epsilon\|_\infty \le 1$.

D.1 Proof of Theorem 2

Everything follows from one deterministic statement: if no omitted column can close the gap Δ under a given perturbation, the two winners agree for that perturbation.

Lemma D.1 (Coincidence). *Let $\mathcal{P}$, $x_q \in \mathcal{P}$, and Δ be as in (9), and let ϵ satisfy $\tau\,(z_x - z_{x_q})^\top \epsilon < \Delta$ for every $x \notin \mathcal{P}$. Then no column outside $\mathcal{P}$ attains the minimum over $\mathcal{R}_0$, so $\hat{x}(\epsilon) = x(\epsilon)$ whenever that minimizer is unique. If $\mathcal{P} = \mathcal{R}_0$, the two winners agree for every ϵ.*

Proof. If $\mathcal{P} = \mathcal{R}_0$ the two problems are the same. Otherwise take $x \notin \mathcal{P}$. The pool bound (9) gives $\bar{c}_x \ge \bar{c}_{x_q} + \Delta$, so

$$\left(\bar{c}_x - \tau z_x^\top \epsilon\right) - \left(\bar{c}_{x_q} - \tau z_{x_q}^\top \epsilon\right) \;\ge\; \Delta - \tau\,(z_x - z_{x_q})^\top \epsilon \;>\; 0.$$

Every omitted column is therefore strictly worse than x_q, which lies in the pool, so the minimum over $\mathcal{R}_0$ is attained inside $\mathcal{P}$, where it is the minimum over $\mathcal{P}$. □

Proof of Theorem 2. For an omitted column x and any ϵ with $\|\epsilon\|_\infty \le 1$, Hölder's inequality gives

$$(z_x - z_{x_q})^\top \epsilon \;\le\; \left\|z_x - z_{x_q}\right\|_1 \|\epsilon\|_\infty \;\le\; R.$$

If $\Delta > \tau R$, the hypothesis of Lemma D.1 holds for every perturbation that can occur, so $\hat{x}(\epsilon) = x(\epsilon)$ almost surely. □

Because the pool is fixed before sampling, the replays $\hat{x}(\epsilon_1), \ldots, \hat{x}(\epsilon_J)$ are functions of independent perturbations, hence independent. Under the condition of the theorem each one equals $x(\epsilon_j)$ almost surely, so the replay estimator $\hat{g}_{\mathcal{P}}$ coincides with $\hat{g}$ of (8), and it has the mean and the covariance computed in Appendix B.2. One search then does the work of J.

D.2 Getting the gap and the radius

Other perturbation laws. Hölder's inequality holds for any norm and its dual norm. If $\|\epsilon\|_* \le B$ almost surely and R bounds $\left\|z_x - z_{x_q}\right\|$ over the omitted columns, the same proof gives exact replay under $\Delta > \tau B R$. Replacing (ϵ, τ) by $(\epsilon/B, \tau B)$ changes no perturbed reduced cost.

The radius. R bounds the distance from x_q to the columns *outside* the pool, and the largest distance seen inside the pool cannot take its place. For binary representations,

$$\left\|z_x - z_{x_q}\right\|_1 = \mathbf{1}^\top z_x + \mathbf{1}^\top z_{x_q} - 2 z_{x_q}^\top z_x \;\le\; \mathbf{1}^\top z_x + \mathbf{1}^\top z_{x_q},$$

so any bound on the number of primitives a column may use gives a valid R. The exact maximum over $\mathcal{R}$ is itself a pricing problem with modified primitive costs, and any upper bound on it may be used instead. Putting the empty column in the pool, which is free, removes the term $\left\|z_0 - z_{x_q}\right\|_1 = \mathbf{1}^\top z_{x_q}$ from the maximum.

A stopping rule. Let $\mathcal{P}_1 \subseteq \mathcal{P}_2 \subseteq \cdots$ be the pools after successive enumeration depths, with gaps $\Delta_1, \Delta_2, \ldots$ that satisfy (9) for their own pool. A bound that holds for every column outside $\mathcal{P}_k$ also holds outside any larger pool, so the running maximum $\max_{k' \le k} \Delta_{k'}$ is a valid gap at depth k and never decreases. The radius may likewise be kept from an earlier depth. Once $\Delta > \tau R$ holds it keeps holding, so the enumeration can test it as it goes and stop at the first depth that passes. A search that reaches the depth limit L first leaves the pool uncertified, and the gradient correction takes over.

D.3 What an uncertified pool costs

Write $\Gamma = \max_{x,x' \in \mathcal{R}_0} \|a_x - a_{x'}\|_2$ for the largest distance between two coefficient vectors, and

$$\rho_{\mathcal{P}} = \mathbb{P}\big(a_{\hat{x}(\epsilon)} \neq a_{x(\epsilon)}\big)$$

for the chance that replay and full pricing return different coefficient vectors, which is at most the chance that they return different columns. Under the condition of Theorem 2, $\rho_{\mathcal{P}} = 0$, for ED and HD pools nothing bounds it.

Lemma D.2 (Bias of uncorrected replay)**.** *For a fixed pool $\mathcal{P}$, the replay estimator satisfies*

$$\mathbb{E}[\hat{g}_{\mathcal{P}}] - \nabla \ell_t(q) = U\,\mathbb{E}\big[a_{\hat{x}(\epsilon)} - a_{x(\epsilon)}\big], \qquad \|\mathbb{E}[\hat{g}_{\mathcal{P}}] - \nabla \ell_t(q)\|_2 \;\le\; U\,\Gamma\,\rho_{\mathcal{P}},$$

and neither expression depends on J.

Proof. Taking expectations in the definition of $\hat{g}_{\mathcal{P}}$ gives $\mathbb{E}[\hat{g}_{\mathcal{P}}] = (q - y_t)/\eta - b + U\,\mathbb{E}[a_{\hat{x}(\epsilon)}]$ for every J, while Proposition 1 gives $\nabla \ell_t(q) = (q - y_t)/\eta - b + U\,\mathbb{E}[a_{x(\epsilon)}]$. Subtracting the two leaves the identity. For the bound, move the norm inside the expectation; driven by the same ϵ, the difference is zero unless the two coefficient vectors differ, and at most Γ when they do. □

More replays shrink the sampling error but not this bias, which is the gap that gradient correction closes. For a fixed pool, $\hat{g}_{\mathcal{P}}$ is the exact gradient of the loss obtained by replacing $\mathcal{R}_0$ with $\mathcal{P}$ in D_τ. Rebuilding the pool at every price would make the update directions gradients of different restricted losses, and in general of no single objective, which is another reason to keep the full-family loss ℓ_t as the target.

D.4 Proof of Proposition 2

Proof of Proposition 2. Write $\mathcal{E} = (\epsilon_1, \ldots, \epsilon_J)$ and

$$V = \frac{1}{J}\sum_{j=1}^{J} C_j + F_I - C_I, \qquad \text{so that} \qquad \widehat{g}_{\mathrm{corr}} = \frac{q - y_t}{\eta} - b + UV.$$

The pool is fixed before sampling, so C_j and F_j depend on ϵ_j alone and the pairs (F_j, C_j) are independent copies of one pair (F, C). The index I is uniform and independent of $\mathcal{E}$, so

$$\mathbb{E}[V \mid \mathcal{E}] = \frac{1}{J}\sum_{j=1}^{J} C_j + \frac{1}{J}\sum_{j=1}^{J}(F_j - C_j) = \frac{1}{J}\sum_{j=1}^{J} F_j.$$

By Proposition 1, $U\,\mathbb{E}[F_j] = u_\tau(q)$, hence $\mathbb{E}[\widehat{g}_{\mathrm{corr}}] = (q - y_t)/\eta - b + u_\tau(q) = \nabla \ell_t(q)$. □

The proof does not rely no optimality property of the pool and no bound on the omitted columns. It dependents on two things: the exact solve at (q, ϵ_I) must run on the same column family, the same primitive representation, and the same perturbation law that define D_τ, since otherwise $U\,\mathbb{E}[F] \neq u_\tau(q)$, and the pool must not depend on the perturbations, which is why any enlargement of it waits until the estimate has been formed. What the correction costs is variance.

Proposition D.3 (Covariance of the corrected estimate)**.** *Let F and C be the two winners under one fresh perturbation. Then*

$$\mathrm{Cov}\big(\widehat{g}_{\mathrm{corr}}\big) = U^2\left[\frac{1}{J}\mathrm{Cov}(F) + \Big(1 - \frac{1}{J}\Big)\mathrm{Cov}(F - C)\right], \qquad \mathrm{tr}\,\mathrm{Cov}(F - C) \;\le\; \Gamma^2 \rho_{\mathcal{P}}.$$

Proof. Keep the notation of the previous proof and write $H_j = F_j - C_j$ with average $\bar{H}$. Given $\mathcal{E}$, the vector V is a constant plus H_I with I uniform, so its conditional covariance is the empirical covariance of $H_1, \ldots, H_J$ with denominator J, whose expectation over independent copies is $(1 - 1/J)\mathrm{Cov}(F - C)$. The conditional mean $J^{-1}\sum_j F_j$ averages J independent copies of F, so its covariance is $\mathrm{Cov}(F)/J$. Adding the two, as the law of total covariance prescribes, and multiplying by U^2 gives the formula. For the trace bound, $\mathrm{tr}\,\mathrm{Cov}(F - C) \le \mathbb{E}\,\|F - C\|_2^2$, and $F - C$ is zero unless the two coefficient vectors differ, which happens with probability $\rho_{\mathcal{P}}$ and costs at most Γ^2. □

Algorithm 1 Learning SCDPs with pool replay

Require: CG states, pricer $\in \{\text{CE}, \text{ED}, \text{HD}\}$, temperature τ, step size η, perturbations J, pool depth L

▷ **Training**
for each recorded state s_t, with RMP dual y_t **do**
  $q \leftarrow q_\theta(s_t)$
  $\mathcal{P} \leftarrow \textsc{BuildPool}(q)$
  draw J perturbations and re-price the pool
  estimate gradient $\hat{g} \leftarrow (q - y_t)/\eta - b + \frac{U}{J}\sum_j a_{\hat{x}(q,\epsilon_j)}$
  if pricer $\neq$ CE or the pool is uncertified **then**
    draw I uniformly from $\{1, \dots, J\}$, solve (PP) at (q, ϵ_I)
    correct gradient $\hat{g} \leftarrow \hat{g} + U\left(a_{x(q,\epsilon_I)} - a_{\hat{x}(q,\epsilon_I)}\right)$
  end if
  update θ by backpropagating $\hat{g}$ through q_θ
end for
▷ **Deploying**
solve the RMP over $\mathcal{R}_t$ for y_t, predict $q \leftarrow q_\theta(s_t)$, and solve (PP) at q, add the returned columns
if there are none column with negative reduced cost at y_t, price exactly at y_t and either add columns or stop with an optimal LP solution

The first term is the variance the estimate would have with exact perturbed pricing, and it falls as $1/J$. The second is what the pool adds, and it is governed by the same $\rho_{\mathcal{P}}$ that governs the bias in Lemma D.2: a pool closer to the full family adds less. Under the condition of Theorem 2 it vanishes, $F = C$ almost surely, the correction term disappears, and the estimator is recovered without the extra solve. For an uncertified pool the second term does not shrink with J, which is the price of one exact solve per update instead of J.

E Experiment Details

This appendix supports Sections 2.3 and 4. Algorithm 1 collects the training loop of Section 2.3 and one guided CG iteration. At iteration t the RMP over $\mathcal{R}_t$ is solved exactly and returns a primal solution w^t, a dual price y_t, and the value P_t. The network proposes $q = q_\theta(s_t)$, and pricing at q, exact or heuristic, returns a set of columns. If the iteration is mis-priced, that is, if no returned column with $\bar{c}_x(y_t) < 0$, pricing is repeated exactly at y_t to avoid CG stuck. The re-pricing either returns a column with $\bar{c}_x(y_t) < 0$, which is added, or reports $\min_{x \in \mathcal{R}} \bar{c}_x(y_t) \geq 0$, and CG stops.

E.1 Baselines

Every baseline runs inside this same loop, with the same pricer, the same column cap per CG iteration K, so the only difference is the price at which pricing is called.

Adaptive dual stabilization. `QP` prices at the minimum-norm point of the optimal dual face of the RMP, which removes the dependence on the vertex the simplex method happens to return; that point comes from a quadratic program over the face. `IPS` (Rousseau et al., 2007) prices at an interior point of the optimal dual face, built from several extreme points of the face. `ACLPS` (Pessoa et al., 2018) combines smoothing with a directional penalty and tunes the stabilization parameters during the run. `PDCGM` (Gondzio et al., 2016) prices at a well-centered suboptimal primal–dual solution of the RMP produced by an interior point method, and tightens its tolerance as the bound gap closes. If the original author reported parameters, we use those parameters. Otherwise, we conduct a grid search to find the most suitable parameters on the validation set. The specific parameter configurations are as follows:

1. `QP` has no parameter of its own: the dual face is read off the current basis and the quadratic program is solved once per iteration.
2. `IPS` averages 20 extreme points, obtained by maximizing and minimizing ten linear functions whose coefficients are drawn uniformly from $[0, 1]$, which is the number of points the original uses.

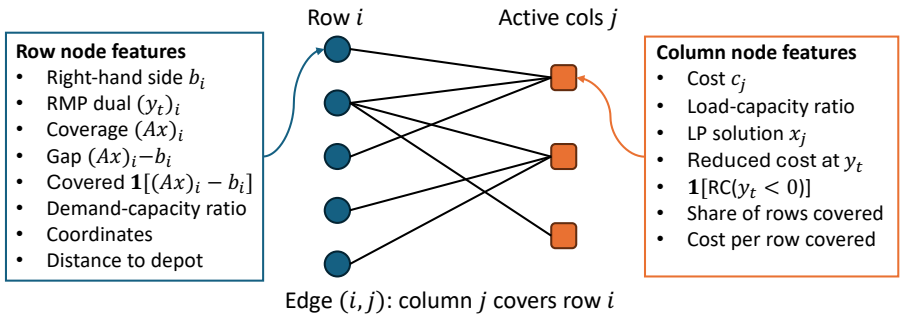


Figure 7: RMP state as a bipartite graph

3. `ACLPS` starts at one half and moves by $1/10$ after every iteration, in the direction the subgradient of the last separation point indicates, and is capped at 0.95.
4. `PDCGM` solves the master by a barrier method without crossover to a relative tolerance of $\min\{0.5,\ \text{gap}/10\}$, the values the authors recommend, and hands over to standard CG once the valid gap falls below 10^{-6}.

Learning-based baselines. `SPARSE` (Kraul et al., 2023) predicts dual values with a supervised model and stabilizes CG with them. `ASCG` (Shen et al., 2024) learns to adjust the stabilization parameter along the run. Both are trained on the same instances as our network and are trained until their loss stabilizes.

`SPARSE` keeps the original's network shape. Its inputs describe the instance alone. The original's five features for CSP (rank and size of an item, its demand share, and the extreme sizes) have no routing counterpart, so for CVRP and CVRPTW we we use ten: the rank of a customer's depot distance and that distance relative to the largest, demand over capacity and demand share, the smallest and the average depot distance relative to the largest, a fleet density, and the average distance to the one, three and five nearest customers relative to the depot distance. The original's monotone post-processing is kept for CSP, where item sizes order the duals, so we dropped it for CVRP and CVRPTW, where no such order exists. Labels are the optimal dual of the last certified iteration of a standard CG run, as the Gurobi solver returns it. We keep the center of a box of width 0.1, and the pricer receives the penalized master's dual. The width is kept while a call adds a new column and halved when it does not, and the box is removed once it has been halved three times for routing and five times for cutting stock, as the authors recommend. After removing the box, `SPARSE` runs as standard CG.

`ASCG` keeps the original's network, trained with Adam at step size 10^{-4} for up to 1,000 epochs and stopped after 100 epochs without improvement on a validation tenth. Its original inputs are statistics of sampled independent sets in a graph coloring master and have no routing counterpart, so it reads the same features as `SPARSE`. The label is the one the authors use: each training instance is solved ten times with a different solver seed and a different set of starting columns, every run certifying the same LP value, and the ten optimal duals are averaged, so that the target is an interior point of the optimal dual face. At run time the box is centered on the prediction, starts at width 1, and is recomputed after every call from the ratio the authors derive, which assumes columns of unit cost. Below a width of 0.01 the box is removed and the run finishes as standard CG.

E.2 Network and Training Setup

State. The state is the RMP itself, encorded as a bipartite graph as shown in Figure 7: one node per row, one node per active column that the current LP solution uses, and an edge wherever the column covers the row. Keeping only the active columns makes the graph's size independent of how long CG has been running. Lengths are divided by the average distance between two locations, so the features do not carry the instance's units. A row node holds nine features: its right-hand side, the anchor price, the coverage the current solution gives it, the difference between the two, an indicator that the row

is exactly covered, the customer's demand over the vehicle capacity, its two coordinates rescaled to the unit square, and its distance to the depot. A column node holds eight: its cost, its load over the capacity, its value in the current solution, an indicator that this value is positive, its reduced cost at the anchor, an indicator that this reduced cost is negative, the share of rows it covers, and its cost per row covered. For cutting stock the rows are item types and the columns patterns, the geometric features are replaced by widths and demands, and each edge carries how many times the pattern cuts the item.

SPARSE and ASCG retain their original prediction settings and therefore use instance-level features instead of intermediate RMP states. To separate the effect of architecture from that of the learning objective, Figure 6(A1) additionally evaluates SCDP with the baseline network: most of the improvement is retained, indicating that the principal gain arises from the self-consistent full-pool objective rather than the particular graph architecture.

Network. Two small feed-forward networks embed the row and the column features into 64 dimensions. Three rounds of message passing follow, each updating the columns from their rows and then the rows from their columns, with the messages summarized by their mean and their maximum and added to the previous embedding. A two-layer head reads one number per row. The network has about 113,000 weights for CVRP and CVRPTW, and about 163,000 for CSP, where the messages also include the edge counts.

Training. We use Adam at step size 10^{-3}, batches of eight states accumulated before each update, gradients clipped at 1, at most 150 epochs, and early stopping after twelve epochs without improvement on a fifth of the instances held out for validation, with the best of those epochs kept. The gradient is with temperature $\tau = 4$, $J = 64$ perturbations, and pools of depth $L = 50$ for CVRP and CVRPTW at 35-node instances. For CSP, the temperature is a hundredth of the average dual and pools hold 200 patterns. The proximal step size is set to the price scale, and we sweep $\eta \in \{0.5, 1, 2\}$ to find the candidate. A perturbation draws one uniform number in $[-1, 1]$ for every primitive. Each training state costs one pricing call to build ED replay pool used in the main experiment. All J draws are then scored against that frozen pool, and then we update the gradient estimator according to Eq. (10). Validation reports the gain in the smoothed dual D_τ at the predicted price q_θ, with the same perturbations used at both CE and HD prices. We round up total demand over vehicle capacity as the weight bound in CVRP and CVRPTW. For CSP, every pattern costs one roll, so $\sum_x w_x = \sum_x c_x w_x$ holds for every feasible solution, and every optimal solution has total weight exactly P.

E.3 Experimental Environment

Experiments were conducted on a workstation equipped with an AMD Ryzen Threadripper PRO 7955WX v16 CPU and 1 NVIDIA RTX 5090 (32GB vRAM), running Ubuntu 22.04 LTS. The models were implemented using Python 3.10 and PyTorch 2.7.0, with additional libraries including torch-scatter 2.1.2, torch-geometric 2.6.1 and Guorbi 12.0.2. The master problems are solved by Gurobi and the routing pricers are those of RouteOpt (You & Yang, 2026).

E.4 Additional Experiment Results

Table 2: Per-instance comparison of the two fastest methods at $K = 1000$ on the CVRP X Set.

	Standard CG				Change vs. standard CG							
					SCDP				SPARSE			
Instance	CG (s)	Iter.	RMP (s)	PP (s)	CG	Iter.	RMP	PP	CG	Iter.	RMP	PP
X-n101-k25	1.12	23	0.16	0.58	**−67%**	**−57%**	**−84%**	**−58%**	−46%	−26%	−69%	−25%
X-n106-k14	9.14	52	2.73	5.42	**−41%**	**−52%**	**−76%**	**−22%**	−17%	−33%	−59%	+8%
X-n110-k13	6.60	55	2.61	2.82	**−63%**	**−64%**	**−95%**	**−28%**	−55%	−60%	−91%	−16%
X-n115-k10	243.07	464	160.90	68.31	**−96%**	**−95%**	**−99%**	**−87%**	−94%	−91%	−99%	−81%
X-n120-k6	87.32	184	61.52	21.71	**−82%**	**−70%**	**−95%**	**−48%**	−56%	−37%	−73%	−8%
X-n125-k30	6.87	51	1.25	4.33	**−52%**	**−63%**	**−82%**	**−38%**	−9%	−24%	−44%	+10%
X-n129-k18	10.49	78	4.76	4.07	**−64%**	**−67%**	**−92%**	**−28%**	−42%	−45%	−71%	−7%
X-n139-k10	45.69	112	17.27	25.36	**−58%**	**−74%**	**−97%**	**−30%**	−44%	−51%	−85%	−14%
X-n148-k46	1.11	26	0.22	0.47	**−55%**	**−50%**	**−87%**	**−21%**	−44%	−23%	−70%	−10%
X-n153-k22	46.93	183	14.67	26.90	**−62%**	**−77%**	**−93%**	**−42%**	−40%	−45%	−69%	−22%
X-n157-k13	18.61	105	11.97	4.36	**−54%**	**−35%**	**−75%**	**−12%**	−31%	−22%	−43%	+1%
X-n162-k11	115.16	152	34.25	76.51	**−48%**	**−74%**	**−97%**	**−24%**	−24%	−47%	−78%	+2%
X-n167-k10	186.95	196	85.72	95.84	**−46%**	**−53%**	**−89%**	−6%	−34%	−44%	−62%	**−8%**
X-n172-k51	5.47	39	0.67	3.86	**−44%**	**−44%**	**−69%**	**−40%**	−22%	−23%	−52%	−8%
X-n176-k26	62.06	157	25.68	31.69	**−75%**	**−80%**	**−97%**	**−56%**	−17%	−22%	−46%	+9%
X-n181-k23	8.73	64	5.11	2.06	**−69%**	**−52%**	**−86%**	**−33%**	−55%	−41%	−72%	−13%
X-n186-k15	99.80	136	30.12	65.68	**−23%**	**−49%**	**−85%**	+7%	**−23%**	**−49%**	−74%	**+3%**
X-n190-k8	976.04	375	448.16	514.24	**−63%**	**−65%**	**−76%**	**−52%**	−46%	−51%	−50%	−40%
X-n195-k51	4.95	40	0.70	3.35	**−58%**	**−60%**	**−91%**	**−46%**	−30%	−38%	−81%	−12%
X-n200-k36	12.34	54	3.02	7.98	**−38%**	**−48%**	**−76%**	**−23%**	−2%	−17%	−44%	+18%
Total	1948.44	2546	911.48	965.55	**−63%**	**−69%**	**−85%**	**−42%**	−48%	−49%	−65%	−28%
Geo. mean	23.01	91	7.51	10.88	**−58%**	**−61%**	**−87%**	**−34%**	−37%	−39%	−67%	−11%

Table 3: Per-instance comparison of the two fastest methods at $K = 1000$ on the Solomon CVRPTW (Family C)

	Standard CG				Change vs. standard CG							
					SCDP				SPARSE			
Instance	CG (s)	Iter.	RMP (s)	PP (s)	CG	Iter.	RMP	PP	CG	Iter.	RMP	PP
C101	0.63	55	0.25	0.22	**−74%**	**−71%**	**−96%**	**−68%**	−66%	−64%	−94%	−63%
C102	1.53	80	0.72	0.59	−67%	−66%	−95%	−49%	**−83%**	**−76%**	**−97%**	**−67%**
C103	4.54	121	1.50	2.49	−74%	−70%	−95%	−67%	**−81%**	**−77%**	**−96%**	**−71%**
C104	7.54	102	1.80	5.22	−53%	−52%	−92%	−42%	**−67%**	**−67%**	**−95%**	**−57%**
C105	0.77	54	0.36	0.24	−63%	**−69%**	**−96%**	−63%	**−83%**	−67%	−95%	**−69%**
C106	1.15	74	0.51	0.41	−73%	−68%	−96%	−63%	**−90%**	**−80%**	**−99%**	**−81%**
C107	0.73	57	0.35	0.24	−59%	−60%	**−94%**	−39%	**−75%**	**−68%**	**−94%**	**−56%**
C108	1.46	75	0.60	0.59	−69%	−61%	−95%	−55%	**−88%**	**−81%**	**−98%**	**−78%**
C109	3.68	113	1.49	1.68	−72%	−65%	−96%	−56%	**−86%**	**−79%**	**−98%**	**−76%**
C201	12.32	256	9.72	1.61	**−81%**	**−63%**	**−94%**	**−54%**	−79%	−50%	−87%	−51%
C202	72.04	772	58.75	11.07	**−95%**	**−87%**	**−98%**	**−82%**	−87%	−68%	−93%	−61%
C203	98.01	541	72.66	22.81	**−91%**	**−74%**	**−98%**	**−75%**	−89%	−68%	−96%	−69%
C204	390.19	835	278.76	106.26	**−95%**	**−87%**	**−99%**	−84%	**−95%**	−86%	**−99%**	**−86%**
C205	112.46	759	102.61	6.19	**−98%**	**−91%**	**−99.6%**	**−88%**	−97%	−79%	−98%	−72%
C206	223.52	986	204.75	13.75	**−99%**	**−92%**	**−99.7%**	**−90%**	−97%	−76%	−98%	−79%
C207	216.44	947	188.38	23.44	**−98%**	**−92%**	**−99.7%**	**−90%**	−97%	−78%	−99%	−81%
C208	1126.19	2168	1057.95	56.16	**−99.8%**	**−97%**	**−99.96%**	**−97%**	−99.6%	−95%	−99.9%	−95%
C1_2_1	0.29	35	0.10	0.10	−2%	**−26%**	**−58%**	−17%	**−31%**	−23%	−48%	**−20%**
C1_2_2	0.80	58	0.27	0.41	−2%	**−29%**	**−55%**	**−11%**	**−18%**	−14%	−50%	+3%
C1_2_3	2.31	64	0.65	1.44	−16%	**−22%**	**−66%**	**−7%**	**−23%**	−16%	−63%	−2%
C1_2_4	6.35	87	1.09	4.86	−29%	−44%	−75%	−24%	**−41%**	**−47%**	**−79%**	**−33%**
C1_2_5	0.42	36	0.19	0.13	−22%	**−28%**	**−69%**	−14%	**−39%**	−22%	−59%	**−18%**
C1_2_6	0.50	39	0.20	0.19	−1%	**−18%**	**−60%**	+5%	**−33%**	−15%	−51%	**−18%**
C1_2_7	0.66	41	0.29	0.25	+4%	−20%	−60%	+33%	**−45%**	**−24%**	**−65%**	**−24%**
C1_2_8	0.97	45	0.40	0.40	−30%	**−33%**	**−78%**	**−29%**	**−42%**	−29%	−68%	−26%
C1_2_9	1.93	59	0.59	1.01	**−47%**	−32%	**−81%**	**−47%**	−41%	**−37%**	−67%	−20%
C1_2_10	2.83	63	0.74	1.85	−29%	**−30%**	**−75%**	−21%	**−44%**	−25%	−73%	**−37%**
C2_2_1	4.85	160	3.44	0.84	**−86%**	**−75%**	**−95%**	**−77%**	−84%	−67%	−90%	−72%
C2_2_2	16.48	257	12.00	3.28	−87%	**−76%**	**−95%**	−73%	**−89%**	−67%	−94%	**−75%**
C2_2_3	22.92	235	13.60	8.17	**−79%**	**−68%**	**−92%**	**−64%**	−73%	−51%	−88%	−51%
C2_2_4	77.91	341	37.16	38.76	−76%	**−67%**	**−94%**	−62%	**−78%**	**−67%**	−93%	**−63%**
C2_2_5	13.98	242	10.86	2.05	**−92%**	**−80%**	**−98%**	**−83%**	**−92%**	−79%	−97%	−80%
C2_2_6	19.96	264	15.74	2.74	−90%	−75%	−96%	−75%	**−94%**	**−78%**	**−98%**	**−78%**
C2_2_7	38.70	383	31.79	5.13	−95%	**−84%**	−98%	**−84%**	**−96%**	−81%	**−99%**	−83%
C2_2_8	46.87	371	39.12	5.79	−94%	**−84%**	**−98%**	**−83%**	**−95%**	**−84%**	**−98%**	−80%
C2_2_9	85.33	455	71.23	11.46	−95%	−80%	**−99%**	−81%	**−97%**	**−85%**	**−99%**	**−88%**
C2_2_10	50.31	322	38.40	10.14	−91%	**−76%**	**−97%**	**−78%**	**−92%**	**−76%**	**−97%**	−77%
Total	2667.58	11552	2259.04	351.94	**−96%**	**−82%**	**−99%**	**−80%**	**−96%**	−77%	**−99%**	−78%
Geo. mean	8.44	163	4.40	2.19	−82%	**−70%**	**−95%**	**−66%**	**−85%**	−67%	−94%	**−66%**

Table 4: Per-instance comparison of the two fastest methods at $K = 1000$ on the Solomon CVRPTW (Family R)

	Standard CG				Change vs. standard CG							
					SCDP				SPARSE			
Instance	CG (s)	Iter.	RMP (s)	PP (s)	CG	Iter.	RMP	PP	CG	Iter.	RMP	PP
R101	0.19	23	0.07	0.05	+43%	**−4%**	**−37%**	+75%	**−1%**	**−4%**	−27%	**+31%**
R102	0.59	37	0.14	0.37	−1%	**−5%**	**−18%**	**−35%**	**−16%**	**−5%**	−8%	−24%
R103	1.28	42	0.34	0.80	+36%	+14%	**−11%**	+31%	**−2%**	**+2%**	+8%	**−11%**
R104	3.37	45	0.71	2.30	+2%	+2%	−44%	+14%	**−18%**	**−4%**	**−52%**	**0%**
R105	0.44	27	0.18	0.15	+48%	+7%	+1%	+24%	**0%**	**0%**	**−6%**	**+10%**
R106	1.01	37	0.37	0.48	**−6%**	**0%**	**−44%**	**−4%**	+26%	+22%	−28%	+86%
R107	1.81	42	0.55	1.11	+30%	+10%	−34%	+24%	**−8%**	**+2%**	**−45%**	**+13%**
R108	3.58	46	0.73	2.55	**0%**	**−7%**	**−45%**	**+2%**	**0%**	+4%	−42%	+18%
R109	0.80	31	0.25	0.30	+6%	+26%	**−30%**	+33%	**−15%**	**+23%**	−24%	**+29%**
R110	1.72	38	0.50	1.04	+2%	**−11%**	−54%	+16%	**−31%**	−5%	**−57%**	**−16%**
R111	1.64	42	0.45	1.07	−10%	−10%	−44%	−11%	**−26%**	**−14%**	**−50%**	**−17%**
R112	3.36	49	0.79	2.36	−17%	−27%	−69%	**−6%**	**−21%**	**−31%**	**−70%**	−1%
R201	13.51	177	10.91	1.86	**−74%**	**−50%**	**−83%**	**−58%**	−73%	−41%	−79%	−55%
R202	21.91	208	15.24	5.51	**−66%**	**−50%**	**−80%**	**−47%**	−62%	−41%	−72%	−39%
R203	52.55	255	29.37	21.50	**−69%**	**−55%**	**−82%**	**−59%**	−62%	−49%	−77%	−42%
R204	147.64	333	73.63	71.48	−76%	−65%	−92%	−63%	**−80%**	**−66%**	**−93%**	**−66%**
R205	34.48	219	25.04	7.77	**−80%**	**−55%**	**−89%**	**−62%**	−75%	−43%	−83%	−51%
R206	64.48	264	40.04	22.67	**−78%**	**−55%**	**−89%**	**−65%**	−73%	−48%	−84%	−56%
R207	95.67	267	49.43	44.08	**−68%**	**−52%**	**−82%**	**−56%**	−67%	−45%	**−82%**	−51%
R208	165.97	326	66.98	96.64	**−73%**	**−63%**	**−92%**	**−61%**	−71%	**−63%**	**−92%**	−57%
R209	38.31	236	24.85	12.09	**−78%**	**−59%**	**−89%**	**−64%**	−75%	−52%	−86%	−52%
R210	40.37	209	23.90	14.95	**−70%**	**−51%**	**−84%**	**−56%**	−66%	−45%	−81%	−43%
R211	144.84	366	75.47	66.64	**−88%**	**−77%**	**−96%**	**−82%**	−84%	−67%	−95%	−72%
R1_2_1	0.49	37	0.25	0.12	+15%	**−16%**	**−31%**	**−5%**	**−19%**	+16%	**−31%**	+12%
R1_2_2	1.06	59	0.55	0.35	+11%	**−27%**	−38%	+9%	**−25%**	−22%	**−43%**	**+1%**
R1_2_3	3.98	84	1.55	1.96	−29%	−27%	−55%	−24%	**−46%**	**−30%**	**−64%**	**−28%**
R1_2_4	12.94	100	3.91	8.39	−10%	**−24%**	−47%	**+1%**	**−17%**	−18%	**−55%**	**+1%**
R1_2_5	1.12	46	0.59	0.37	+13%	+26%	−20%	**−28%**	**−22%**	**+13%**	**−44%**	−24%
R1_2_6	2.25	70	0.96	0.92	−42%	**−40%**	**−64%**	**−50%**	**−46%**	−20%	−51%	−37%
R1_2_7	4.51	90	1.78	2.32	−14%	−22%	−35%	−18%	**−38%**	**−40%**	**−49%**	**−30%**
R1_2_8	15.59	104	4.62	10.32	−29%	−26%	−62%	−19%	**−41%**	**−32%**	**−73%**	**−26%**
R1_2_9	1.57	61	0.84	0.40	−25%	−13%	−46%	**−24%**	**−28%**	**−16%**	**−48%**	+38%
R1_2_10	3.66	88	1.96	1.40	−48%	**−42%**	**−71%**	**−43%**	**−50%**	−35%	−70%	−35%
R2_2_1	18.61	232	15.99	1.69	**−88%**	**−71%**	**−93%**	**−78%**	−87%	−63%	−91%	−70%
R2_2_2	34.31	306	28.26	4.52	−79%	**−61%**	**−88%**	**−65%**	**−81%**	−59%	−86%	−56%
R2_2_3	151.58	643	124.10	24.26	**−88%**	**−74%**	**−94%**	**−66%**	**−88%**	−70%	**−94%**	−59%
R2_2_4	1849.70	1683	1549.48	286.63	**−98%**	**−88%**	**−99%**	**−90%**	−96%	−86%	**−99%**	−83%
R2_2_5	73.94	361	67.28	4.28	−93%	−72%	**−96%**	**−77%**	**−94%**	**−73%**	**−96%**	−74%
R2_2_6	84.23	364	71.28	10.71	−85%	−58%	**−91%**	**−64%**	**−87%**	**−60%**	**−91%**	−61%
R2_2_7	192.13	502	146.30	42.69	**−87%**	**−63%**	**−95%**	**−67%**	−84%	−62%	−92%	−59%
R2_2_8	5031.73	2885	4341.64	668.90	**−99%**	**−94%**	**−99.9%**	**−96%**	**−99%**	**−94%**	−99.7%	−94%
R2_2_9	170.50	524	157.60	9.71	**−95%**	**−77%**	**−98%**	**−78%**	**−95%**	−64%	−97%	−73%
R2_2_10	258.07	622	234.14	19.88	**−96%**	**−77%**	**−98%**	**−84%**	**−96%**	−71%	**−98%**	−83%
Total	8751.50	12180	7193.03	1477.60	**−95%**	**−70%**	**−98%**	**−83%**	**−95%**	−67%	**−98%**	−78%
Geo. mean	12.65	134	6.16	3.92	−64%	**−47%**	**−81%**	**−48%**	**−66%**	−42%	−79%	−42%

Table 5: Per-instance comparison of the two fastest methods at $K = 1000$ on the Solomon CVRPTW (Family RC)

	Standard CG				Change vs. standard CG							
					SCDP				SPARSE			
Instance	CG (s)	Iter.	RMP (s)	PP (s)	CG	Iter.	RMP	PP	CG	Iter.	RMP	PP
RC101	0.32	26	0.12	0.12	−3%	−12%	**−58%**	+5%	**−44%**	**−15%**	−56%	**−33%**
RC102	0.94	33	0.33	0.43	+2%	+15%	**−63%**	+36%	**−32%**	**−9%**	−48%	**−22%**
RC103	1.40	37	0.34	0.72	−19%	+3%	−54%	+6%	**−33%**	**−5%**	**−60%**	**+1%**
RC104	4.15	52	0.77	2.96	**−19%**	**−29%**	**−69%**	**−5%**	−17%	−13%	−53%	−1%
RC105	0.64	27	0.14	0.23	−3%	+11%	−13%	+18%	**−47%**	**+7%**	**−37%**	**−14%**
RC106	0.68	31	0.22	0.35	+48%	+6%	**−44%**	+43%	**+5%**	**−10%**	−10%	**+8%**
RC107	1.72	37	0.57	0.95	−32%	**−16%**	**−73%**	**−37%**	**−47%**	**−16%**	**−73%**	−31%
RC108	2.55	40	0.46	1.91	+2%	**−12%**	**−65%**	**+12%**	**−3%**	−7%	−62%	+14%
RC201	8.62	129	5.75	1.89	−71%	**−56%**	**−86%**	−46%	**−74%**	−55%	−81%	**−55%**
RC202	13.81	185	9.34	3.55	**−66%**	**−54%**	**−83%**	**−43%**	−63%	−50%	−78%	−28%
RC203	73.27	273	44.52	26.77	**−78%**	**−62%**	**−92%**	**−61%**	**−78%**	−55%	−91%	−56%
RC204	206.43	559	107.53	95.57	−82%	**−84%**	**−98%**	−65%	**−86%**	−82%	**−98%**	**−73%**
RC205	14.63	188	9.60	4.05	−64%	−48%	−82%	−45%	**−74%**	**−52%**	**−85%**	**−56%**
RC206	23.12	200	15.00	6.89	−77%	**−62%**	**−91%**	−61%	**−79%**	−54%	−87%	**−65%**
RC207	18.04	171	9.75	7.53	**−65%**	**−50%**	**−83%**	**−51%**	−64%	−42%	−80%	−46%
RC208	258.97	558	156.96	98.49	**−91%**	**−85%**	**−99%**	**−81%**	**−91%**	−81%	**−99%**	−79%
RC1_2_1	0.75	47	0.37	0.21	+13%	−13%	**−53%**	+53%	**−34%**	**−17%**	−52%	**+9%**
RC1_2_2	1.71	56	0.71	0.79	−17%	**−12%**	−59%	−13%	**−43%**	−11%	**−61%**	**−28%**
RC1_2_3	4.66	83	1.90	2.37	−15%	**−20%**	**−59%**	+7%	**−29%**	−12%	**−59%**	**0%**
RC1_2_4	14.43	123	4.38	9.36	−29%	−39%	**−70%**	−15%	**−44%**	**−46%**	**−70%**	**−31%**
RC1_2_5	2.12	62	0.86	0.87	**−51%**	**−27%**	**−72%**	**−44%**	−3%	−6%	−38%	+56%
RC1_2_6	1.48	52	0.66	0.63	−25%	**−19%**	−62%	**−16%**	**−40%**	−8%	**−63%**	−15%
RC1_2_7	3.18	69	1.43	1.37	−23%	−16%	−66%	**−4%**	**−40%**	**−19%**	**−70%**	−1%
RC1_2_8	3.95	72	1.52	2.05	**−19%**	**−25%**	**−65%**	**+6%**	−14%	−19%	−53%	+22%
RC1_2_9	4.53	71	1.75	2.34	−13%	**−21%**	−65%	+21%	**−33%**	**−21%**	**−80%**	**+4%**
RC1_2_10	5.95	84	1.95	3.75	−8%	−29%	−76%	+22%	**−40%**	**−51%**	**−88%**	**−13%**
RC2_2_1	12.74	279	10.10	1.77	−70%	−63%	**−85%**	**−56%**	**−79%**	**−65%**	**−85%**	−51%
RC2_2_2	21.65	301	16.31	4.06	**−72%**	**−59%**	**−86%**	**−45%**	−42%	−27%	−66%	+56%
RC2_2_3	151.55	517	109.01	39.15	−82%	−59%	**−94%**	−55%	**−86%**	**−62%**	**−94%**	**−65%**
RC2_2_4	983.54	1205	705.13	269.02	−91%	**−80%**	**−98%**	−76%	**−93%**	−76%	**−98%**	**−80%**
RC2_2_5	42.17	371	31.52	9.11	**−79%**	**−68%**	**−92%**	**−46%**	−78%	−60%	−88%	**−46%**
RC2_2_6	61.00	447	50.86	7.96	**−90%**	**−69%**	**−96%**	**−66%**	−88%	−62%	−94%	−55%
RC2_2_7	86.81	441	71.90	12.56	−86%	**−60%**	**−95%**	−47%	**−90%**	−59%	**−95%**	**−62%**
RC2_2_8	221.56	643	175.31	42.20	**−90%**	**−75%**	**−97%**	**−64%**	−89%	−66%	−96%	−61%
RC2_2_9	338.97	754	276.98	56.88	−93%	**−83%**	**−99%**	−71%	**−94%**	−80%	**−99%**	**−72%**
RC2_2_10	439.95	857	339.14	95.22	−92%	**−79%**	**−98%**	−70%	**−93%**	−77%	**−98%**	**−73%**
Total	3031.99	9080	2163.15	814.11	−88%	**−66%**	**−97%**	−67%	**−89%**	−62%	−96%	**−70%**
Geo. mean	11.67	138	5.62	4.00	−61%	**−49%**	**−85%**	−36%	**−66%**	−45%	−84%	**−38%**
Total, all	14451.06	32812	11615.22	2643.65	**−94%**	**−73%**	**−98%**	**−78%**	**−94%**	−69%	**−98%**	−76%
Geo. mean, all	10.84	144	5.38	3.28	−71%	**−56%**	**−89%**	**−52%**	**−74%**	−53%	−87%	−50%

Table 6: Per-instance comparison of the two fastest methods at $K = 50$ on the AI/ANI, SSTriples and GI families of BPPLIB.

	Standard CG				Change vs. standard CG							
					SCDP				PDCGM			
Instance	CG (s)	Iter.	RMP (s)	PP (s)	CG	Iter.	RMP	PP	CG	Iter.	RMP	PP
AI/ANI ($\|\mathcal{I}\| = 15$)												
201_2500_DI_31	0.62	78	0.46	0.11	**+15%**	**−14%**	**−29%**	**−6%**	+18%	+5%	−11%	+13%
201_2500_DI_44	0.64	78	0.51	0.09	+20%	**−6%**	+4%	**−2%**	**+6%**	−4%	**−23%**	+11%
201_2500_NR_0	0.49	72	0.36	0.09	**+36%**	**−4%**	+23%	**+1%**	+47%	+12%	**+19%**	+32%
201_2500_NR_1	0.66	78	0.52	0.10	**−7%**	**−6%**	**−30%**	**−3%**	+18%	−1%	−6%	+15%
201_2500_NR_16	0.54	81	0.40	0.10	**+12%**	**−11%**	**+5%**	**+3%**	+45%	+9%	+22%	+24%
201_2500_NR_38	0.66	82	0.50	0.11	**+19%**	**−3%**	**−3%**	**−1%**	+21%	+4%	−2%	+3%
402_10000_DI_6	2.30	138	1.79	0.40	**−6%**	−1%	**−12%**	**+2%**	+36%	**−5%**	+31%	+3%
402_10000_DI_40	5.03	157	4.32	0.57	**−42%**	**−11%**	**−47%**	**−2%**	+8%	+2%	0%	+20%
402_10000_NR_11	3.22	140	2.59	0.51	**−36%**	**−8%**	**−42%**	**−5%**	+27%	+1%	+22%	+7%
600_20000_DI_24	8.11	198	6.36	1.53	**−24%**	**−10%**	**−24%**	**−9%**	+18%	−4%	+16%	+2%
600_20000_DI_48	11.55	215	9.69	1.61	**−25%**	+1%	**−31%**	+5%	−22%	**−19%**	−28%	**−13%**
600_20000_DI_49	10.46	203	8.63	1.51	**−34%**	+6%	**−37%**	+3%	−10%	**−11%**	−16%	**−6%**
600_20000_NR_5	11.75	214	9.74	1.66	**−41%**	+3%	**−44%**	+6%	−19%	**−12%**	−23%	**−15%**
600_20000_NR_37	16.17	233	14.02	1.88	**−31%**	−9%	−34%	**−6%**	−28%	**−10%**	**−35%**	+5%
600_20000_NR_48	13.75	220	11.74	1.68	**−39%**	+1%	**−42%**	**−1%**	−8%	**−8%**	−17%	+34%
Total	85.95	2187	71.64	11.95	**−33%**	**−7%**	**−39%**	**−2%**	−8%	−6%	−16%	+3%
SSTriples ($\|\mathcal{I}\| = 15$))												
N216W10000-005	1.16	135	0.61	0.35	**−54%**	**−64%**	**−63%**	**−37%**	+2%	−16%	−11%	−6%
N216W10000-010	0.91	124	0.53	0.30	**−57%**	**−72%**	**−70%**	**−43%**	+14%	−24%	−24%	−12%
N216W10000-024	0.91	117	0.53	0.31	**−56%**	**−72%**	**−69%**	**−42%**	+11%	−15%	−22%	+1%
N216W10000-029	0.75	111	0.44	0.25	**−46%**	**−61%**	**−64%**	**−32%**	+22%	−10%	−6%	−2%
N216W10000-040	0.94	124	0.55	0.31	**−51%**	**−57%**	**−69%**	**−33%**	+10%	−19%	−17%	−8%
N216W10000-043	0.82	112	0.46	0.29	**−57%**	**−74%**	**−69%**	**−48%**	+17%	−14%	−15%	−2%
N216W10000-047	0.98	125	0.57	0.33	**−48%**	**−74%**	**−71%**	**−50%**	−1%	−20%	−24%	−15%
N405W15000-003	6.15	292	4.16	1.72	**−68%**	**−72%**	**−77%**	**−44%**	−18%	−24%	−30%	−21%
N405W15000-009	4.90	251	2.94	1.61	**−61%**	**−65%**	**−70%**	**−46%**	−16%	−19%	−28%	−24%
N405W15000-010	5.80	275	3.69	1.81	**−65%**	**−64%**	**−75%**	**−47%**	−13%	−18%	−24%	−24%
N405W15000-015	5.76	283	3.69	1.79	**−65%**	**−69%**	**−75%**	**−47%**	−22%	−27%	−34%	−31%
N405W15000-016	5.96	299	3.67	1.89	**−69%**	**−71%**	**−76%**	**−54%**	−30%	−32%	−41%	−35%
N405W15000-018	5.64	281	3.49	1.76	**−67%**	**−72%**	**−75%**	**−52%**	−26%	−26%	−40%	−30%
N405W15000-024	7.79	341	5.23	2.12	**−70%**	**−68%**	**−81%**	**−45%**	−44%	−40%	−56%	−39%
N405W15000-030	5.13	267	3.29	1.59	**−62%**	**−65%**	**−73%**	**−42%**	−20%	−25%	−33%	−26%
Total	53.58	3137	33.83	16.42	**−65%**	**−68%**	**−75%**	**−46%**	−21%	−24%	−35%	−26%
GI ($\|\mathcal{I}\| = 15$)												
csAA125_4	0.28	38	0.05	0.22	**−39%**	**−55%**	**−36%**	**−21%**	+4%	−21%	−30%	−6%
csAA125_5	0.30	39	0.04	0.23	−5%	**−56%**	**−43%**	−19%	**−14%**	−31%	−42%	**−26%**
csAA125_17	0.30	41	0.06	0.23	**−47%**	**−61%**	−52%	**−31%**	−18%	−44%	**−56%**	−22%
csAA250_11	1.04	93	0.34	0.64	**−54%**	**−67%**	**−62%**	**−41%**	−18%	−29%	−42%	−22%
csAA250_18	1.07	97	0.34	0.65	**−56%**	**−67%**	**−64%**	**−44%**	−32%	−38%	−59%	−33%
csAA500_7	5.34	246	3.01	2.07	**−68%**	**−75%**	**−75%**	**−54%**	−43%	−48%	−63%	−25%
csAA500_13	5.24	260	2.95	2.04	**−68%**	**−75%**	**−76%**	**−56%**	−44%	−47%	−61%	−31%
csAA500_15	5.93	261	3.44	2.23	**−68%**	**−75%**	**−75%**	**−55%**	−52%	−50%	−65%	−42%
csAB125_7	0.36	40	0.08	0.25	**−16%**	**−32%**	**−28%**	**+12%**	+36%	+18%	−2%	+28%
csAB250_5	1.23	83	0.55	0.61	**−8%**	**−30%**	−16%	+11%	−6%	−8%	**−30%**	**−1%**
csAB500_15	5.04	157	3.34	1.53	**−27%**	**−46%**	−34%	**−9%**	−15%	−16%	**−40%**	+23%
csBA250_6	1.44	117	0.48	0.78	**−66%**	**−71%**	**−75%**	**−50%**	−43%	−45%	−65%	−36%
csBA500_8	5.24	272	3.04	1.97	**−70%**	**−76%**	**−79%**	**−58%**	−55%	−54%	−68%	−47%
csBB250_8	1.03	77	0.48	0.50	**−23%**	**−42%**	**−32%**	**−1%**	−1%	−4%	−28%	+4%
csBB500_1	5.15	164	3.46	1.55	**−24%**	**−44%**	−32%	**−3%**	−20%	−19%	**−40%**	+8%
Total	38.99	1985	21.65	15.49	**−52%**	**−64%**	**−59%**	**−38%**	−35%	−37%	−54%	−21%