FROM ALGORITHMS TO REASONING: OPERATIONS RESEARCH AS PROCESS SUPERVISION FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous combinatorial optimization remains challenging for language models, while operations research provides well-established algorithms for these problems, and internalizing these algorithms as learned solution strategies could enhance the models' optimization capabilities. We propose a training framework that uses operations research as verifiable process supervision, translating algorithmic structures and mathematical evaluation criteria into learning signals for intermediate decisions. The model constructs and revises candidate solutions through natural-language reasoning, while a verifier used during training evaluates and compares alternative decisions from the same state using constraint satisfaction, objective values, and algorithm-specific quality criteria. Under shared process-quality criteria, the framework combines local preference learning weighted by quality gaps with reinforcement learning over complete trajectories to jointly train local decision-making and global coordination. Experiments across six classes of combinatorial optimization problems demonstrate improvements in feasibility rates and solution quality. This approach to algorithm internalization could strengthen language models' reasoning over long-horizon tasks in real-world settings, providing a methodological foundation for agent decision-making.

1 INTRODUCTION

Large language models (LLMs) are increasingly studied as agents that reason and act in interactive environments (Yao et al., 2023; Shinn et al., 2023; Liu et al., 2024c). Practical tasks such as travel planning (Xie et al., 2024) and robotic three-dimensional bin packing (Song et al., 2023) require translating task understanding into feasible plans. Such tasks often involve combinatorial optimization, where constraints determine feasibility and an objective distinguishes better solutions. Training LLMs to construct such solutions directly through their own reasoning abilities could therefore support timely, autonomous decision-making even in less structured settings (Li et al., 2026).

This need has motivated an emerging line of research on training LLMs as direct combinatorial optimization solvers. FOARL (Jiang et al., 2025a) and Forge (Li et al., 2026) use reinforcement learning with rewards based on constraint satisfaction and objective values. However, these approaches derive their optimization feedback primarily from completed solutions, leaving the intermediate solving process without explicit supervision. The algorithmic and theoretical foundations of operations research (OR), including decomposition algorithms, mathematical proofs, and criteria for evaluating intermediate progress, remain underutilized as a basis for supervising solution construction and refinement.

Column generation and branch-and-price provide a principled framework for optimization problems that admit a suitable decomposition (Lübbecke & Desrosiers, 2005; Barnhart et al., 1998). Column generation alternates between generating candidate components, or columns, and optimizing their combination in a restricted master problem. Branch-and-price adds branching to address integrality. Their mathematical derivations and correctness proofs are grounded in linear programming duality and branch-and-bound theory, yielding explicit conditions for certifying optimality. Reduced costs assess columns relative to the current dual solution, while feasibility checks, objective values, and valid bounds quantify solution quality and intermediate progress. Together, these properties con-

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.

nect a structured solving process with mathematically justified evaluation criteria, providing explicit signals for learning to construct and refine solutions.

We propose a training framework that translates OR solution strategies into verifiable process supervision for autonomous combinatorial optimization. At its core is CG-CoT, a block-structured chain of thought that organizes solution construction around the decomposition structure of column generation. Each block couples natural-language reasoning with an explicit optimization decision, making intermediate solutions accessible to mathematical evaluation. Solver-derived CG-CoT demonstrations initialize the policy through supervised fine-tuning. The same OR evaluation criteria then guide direct preference optimization over local blocks and reinforcement learning over complete trajectories. Preference weights reflect mathematical quality differences between intermediate decisions, while trajectory rewards combine process quality with final feasibility and objective value. At inference, the LLM serves as an end-to-end solver, generating a solution within a single response without external solver calls. Its solution quality therefore reflects its own learned optimization capability.

Our main contributions are summarized below.

- We introduce CG-CoT, which formalizes chain-of-thought as a sequence of optimization problems. It decomposes a full optimization task into smaller subproblems within LLM reasoning, allowing learned optimization intuitions to guide column generation and integer-plan assembly.
- We develop an OR-driven joint RDO–RLOO algorithm that trains LLM reasoning on small combinatorial optimization subproblems and coordinates these local capabilities to solve larger optimization problems.
- To our knowledge, we are the first to use mathematical criteria from column generation and branch-and-price to supervise the reasoning process of LLMs that directly construct combinatorial optimization solutions. These signals guide local preference learning and reinforcement learning over complete trajectories.

This framework offers a setting for studying algorithm internalization in LLMs and may inform long-horizon reasoning in real-world tasks, including understanding and planning in embodied systems.

2 Related Work

LLMs for Optimization. LLMs generate formulations and solver programs through agent coordination (Xiao et al., 2024; Ahmaditeshnizi et al., 2024), supervised training (Huang et al., 2025a; Yang et al., 2025), and constraint-document retrieval (Jiang et al., 2025b). Algorithm-design methods combine generated programs or heuristic descriptions with execution-based feedback, including FunSearch (Romera-Paredes et al., 2024), EoH (Liu et al., 2024a), ReEvo (Ye et al., 2024), LLaMEA (van Stein & Bäck, 2025), and MCTS-AHD (Zheng et al., 2025). OPRO (Yang et al., 2024) guides proposals using previous candidates and objective values. We use OR evaluation during training to internalize solution construction within a single model response.

Learning for Combinatorial Optimization. Learning supports both direct construction and components of mathematical solvers (Bengio et al., 2021). Neural approaches use pointer networks, graph representations, attention, and local rewriting (Vinyals et al., 2015; Khalil et al., 2017; Kool et al., 2019; Kwon et al., 2020; Chen & Tian, 2019). Learned solver components address branching (Gasse et al., 2019), pricing (Shen et al., 2022; Abouelrous et al., 2025), column selection (Morabit et al., 2021; Chi et al., 2022; Yuan et al., 2024; Hu et al., 2025), pricing order (Koutecká et al., 2025), and dual stabilization (Shen et al., 2024). Our framework transfers OR evaluation criteria to textual optimization trajectories generated by a pretrained LLM.

Process Supervision for Language Models. Post-training combines human feedback (Ouyang et al., 2022), DPO (Rafailov et al., 2023), verifiable rewards (Guo et al., 2025), and rationale bootstrapping (Zelikman et al., 2022). Intermediate supervision uses annotations (Lightman et al., 2024), sampled outcomes (Wang et al., 2024), progress estimates (Setlur et al., 2025), generative verification (Zhao et al., 2026), implicit rewards (Cui et al., 2026), and stepwise preferences (Setlur et al.,

2024; Lu et al., 2025). OR-PRM learns a modeling critic (Wang et al., 2026), while StepORLM uses process judgments and solver verification for modeling DPO (Zhou et al., 2026). We mathematically evaluate explicit columns and plans, weighting local preferences by quality gaps before training fresh complete responses.

End-to-End LLMs for Combinatorial Optimization. Graph encodings and reasoning benchmarks characterize direct solution capabilities (Fatemi et al., 2024; Fan et al., 2024; Tang et al., 2025; Jiang et al., 2026). Searchformer, search-stream learning, and Dualformer train on serialized or selectively shortened search traces (Lehnert et al., 2024; Gandhi et al., 2024; Su et al., 2025). GraphWiz combines graph demonstrations with DPO (Chen et al., 2024), and GraphThought constructs heuristic- and solver-guided reasoning trajectories (Huang et al., 2025b). FOARL (Jiang et al., 2025a) and Forge (Li et al., 2026) primarily reward completed solutions. Our column-generation decomposition additionally exposes intermediate decisions to OR-based preference and reinforcement supervision.

3 PRELIMINARIES

We consider configuration masters that combine locally feasible columns through linking constraints. In a Dantzig–Wolfe formulation (Dantzig & Wolfe, 1960), group k has configurations $\mathcal{P}_k$ and constraint $\sum_{p\in\mathcal{P}_k} \lambda_{kp} = m_k$, with $m_k = 1$ in the basic case. Let $z_L^\star, z_I^\star$ denote the full LP and integer optima.

Column generation optimizes a restricted master problem (RMP) over pools $\mathcal{C}_k \subseteq \mathcal{P}_k$ (Lübbecke & Desrosiers, 2005). For cost c_{kp}, linking coefficients a_{kp}, e_{kp}, and dual multipliers $y \geq 0, \nu, \sigma_k$, pricing minimizes the reduced cost $r_{kp} = c_{kp} - a_{kp}^\top y - e_{kp}^\top \nu - \sigma_k$ over $\mathcal{P}_k$. Branch-and-price embeds this procedure in branch-and-bound to address integrality (Barnhart et al., 1998). Appendix A gives the formulations, dual-variable definitions, and bound scope.

4 METHODOLOGY

4.1 OVERVIEW

Our framework trains LLMs to construct integer solutions through *CG-CoT*, using solver-derived demonstrations for supervised fine-tuning (SFT), followed by alternating OR-reward-weighted direct preference optimization (RDO) and REINFORCE leave-one-out (RLOO) updates (Figure 1). A shared OR evaluator supplies local preferences and process rewards for complete responses.

4.2 BLOCK-STRUCTURED COLUMN GENERATION CHAIN-OF-THOUGHT

CG-CoT represents a response $y_{1:T}$ to instance I as autoregressive blocks $B_t = y_{a_t:b_t}$, each combining natural-language reasoning with a typed operation and closing delimiter. Conditioned on the preceding text $h_t = (I, y_{<a_t})$, the block has probability

$$p_\theta(B_t \mid h_t) = \prod_{j=a_t}^{b_t} \pi_\theta(y_j \mid I, y_{<j}). \tag{1}$$

Completed `COLUMNS` blocks expand a pool $\mathcal{C}_t$ that subsequent `MASTER` blocks use to construct integer plans, with `FINAL` submitting the solution. The model conditions on its generated text, while the training evaluator performs LP and dual computations.

A pricing-stage `COLUMNS` block proposes a nonempty batch $\Delta\mathcal{C}_t \subseteq \mathcal{P}_k \setminus \mathcal{C}_{t,k}$ of new columns for group k. Under the prefix's fixed dual, the reduced costs $r_{t,kp}$ defined in Section 3 give the pricing target and batch objective

$$\rho_{t,k}^\star = \min_{p\in\mathcal{P}_k} r_{t,kp}, \qquad \bar{r}_{h_t}(B_t) = \frac{1}{|\Delta\mathcal{C}_t|} \sum_{p\in\Delta\mathcal{C}_t} r_{t,kp}. \tag{2}$$

A `MASTER` block targets the integer problem restricted to the accumulated pool. Let $\mathcal{X}_I$ be the original feasible set, including auxiliary variables, and let $\lambda(x)$ identify the columns used by plan x.

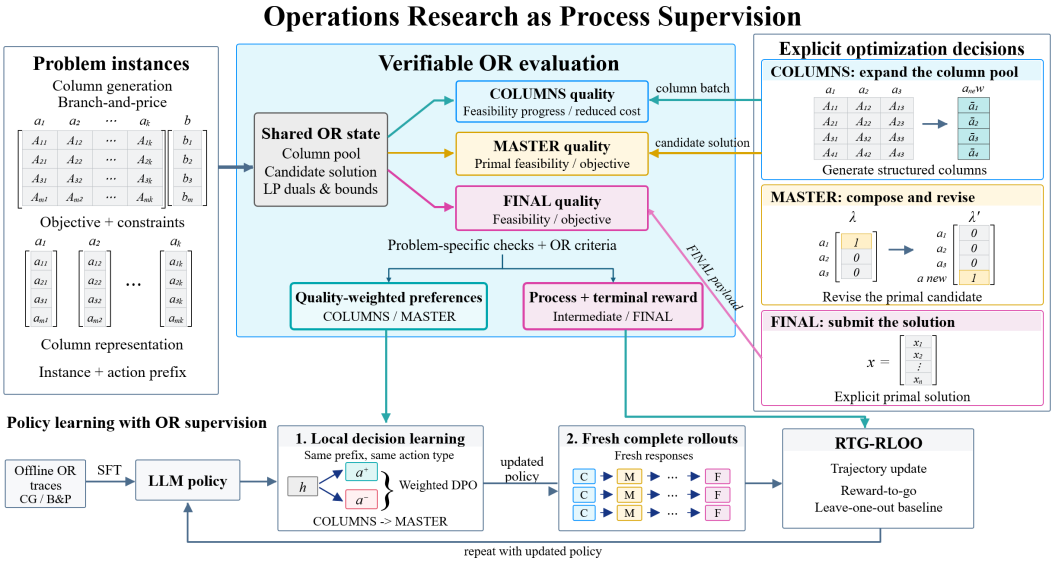


Figure 1: OR supervision for CG-CoT. RDO updates precede fresh-response RLOO, with a shared training evaluator. At inference, the policy constructs the solution within its own response.

Then

$$\begin{aligned} \mathcal{C}_{t+1} &= \mathcal{C}_t \cup \Delta\mathcal{C}_t, & \mathcal{X}_I(\mathcal{C}_t) &= \{x \in \mathcal{X}_I : \operatorname{supp}\lambda(x) \subseteq \mathcal{C}_t\}, \\ v_I(\mathcal{C}_t) &= \min_{x \in \mathcal{X}_I(\mathcal{C}_t)} f(x), & z_I^\star &\le v_I(\mathcal{C}_{t+1}) \le v_I(\mathcal{C}_t). \end{aligned} \tag{3}$$

Here $\Delta\mathcal{C}_t$ is empty for blocks that add no columns. Although this cumulative integer formulation differs from standard branch-and-price, it retains non-increasing restricted optima and theoretical solution bounds. For any feasible proposal of cost U_t, a global lower bound L_t gives $0 \le U_t - z_I^\star \le U_t - L_t$, without requiring the proposal to attain the restricted optimum (Appendix B).

To construct SFT demonstrations, we start from a feasible pool and serialize chronological solver discoveries as `COLUMNS` blocks. An offline integer master reassembles each expanded pool under the original constraints, excluding node-specific branch restrictions. The better feasible plan between the new solution and the preceding incumbent becomes the next `MASTER`, accompanied by an explanation of feasibility or objective changes. The trace ends with `FINAL`, yielding non-increasing teacher objectives even with time-limited solves. Appendix B gives the proofs and examples for all six tasks.

4.3 Blockwise RDO–RLOO Training

Although SFT teacher plans match the cumulative integer formulation, their columns originate from LP-based pricing along retained branch-and-price search paths, limiting supervision for alternative decisions. Blockwise RDO–RLOO therefore trains on model-generated alternatives, combining local quality comparisons with reinforcement learning over complete solutions.

Blockwise preference optimization. RDO learns to distinguish operations by sampling multiple blocks $B \sim p_\theta(\cdot \mid h)$ at selected `COLUMNS` and `MASTER` prefixes from current-policy responses. The OR evaluator compares candidates of the same type and prefix using validity, feasibility, and the quality criteria in Section 4.4, yielding a preferred block B^+, a rejected block B^-, and weight w_h. With the SFT policy frozen as p_{ref}, the weighted DPO objective (Rafailov et al., 2023) is

$$\mathcal{L}_{\text{pref}} = -\mathbb{E}\big[w_h \log \operatorname{sigmoid}\big(\beta\,[g_\theta(B^+ \mid h) - g_\theta(B^- \mid h)]\big)\big], \quad g_\theta = \log \frac{p_\theta}{p_{\text{ref}}}. \tag{4}$$

Quality-gap weights are capped within each comparison type. After collecting both block types, we update `COLUMNS` and then `MASTER` preferences (Appendix D.2).

Reinforcement learning over complete solutions. After RDO, RLOO trains global coordination using $K \geq 2$ independently sampled fresh responses τ_i per instance from the updated policy. Each receives $R_i = R_F(\tau_i) + \lambda_P P(\tau_i)$, combining final solution quality with the process score in Section 4.4. The token factorization in Equation (1) lets us compute block gradients by summing token log-probability gradients (Appendix D.3). With a leave-one-out baseline (Ahmadian et al., 2024), the update is

$$\widehat{A}_{i,t} = R_i - \frac{1}{K-1}\sum_{j\neq i} R_j + \eta q(h_{i,t}),$$
$$\widehat{g}_{\mathrm{RLOO}} = \frac{1}{K}\sum_{i=1}^{K}\sum_{t\in\mathcal{S}_i} \widehat{A}_{i,t}\nabla_\theta \log p_\theta(B_{i,t} \mid h_{i,t}). \tag{5}$$

Here $\mathcal{S}_i$ partitions all generated tokens into blocks and residual text, including unsuccessful outputs. The history term $q(h_{i,t})$ uses preceding process losses and preserves the expected complete-reward gradient under the conditions in Appendix D.3, with $\eta = 0$ recovering ordinary RLOO. New local candidates are collected after each RLOO update.

4.4 Process Supervision Driven by Operations Research Theory

We introduce OR evaluation signals into process supervision without requiring an exact solution to potentially NP-hard pricing or integer master problems for each candidate. Using reduced costs and tractable relaxation bounds, we evaluate column proposals by pricing quality and integer plans by their objective gap. Let $\underline{\rho}_h$ bound the pricing optimum from below under the prefix's dual, and let $L_M \leq z_I^\star$ be a task-specific global lower bound. For a valid column block B and feasible plan M with cost $U(M)$, we define

$$\ell_C(B \mid h) = \frac{\bar{r}_h(B) - \underline{\rho}_h}{s_C}, \qquad \ell_M(M \mid h) = \frac{U(M) - L_M}{s_M + U(M) - L_M}, \tag{6}$$

where $s_C, s_M > 0$. Feasibility-restoration blocks instead use a constraint-violation score (Appendix E).

This reward design is motivated by dual-feasibility correction (Gondzio et al., 2013), which converts subproblem bounds into a global lower bound without requiring exact pricing or an optimal RMP solution.

Proposition 1 (Pricing bounds and master error)**.** *For the finite DW master in Section 3, let u_C, d_C be RMP primal–dual feasible objectives and $\underline{\rho}_k \leq \min_{p\in\mathcal{P}_k} r_{kp}$ under that dual. Setting $\delta_k = \max(0, -\underline{\rho}_k)$ gives*

$$L_{\mathrm{DW}} = d_C - \sum_k m_k\delta_k \leq z_L^\star \leq z_I^\star, \qquad 0 \leq u_C - z_L^\star \leq u_C - L_{\mathrm{DW}}. \tag{7}$$

The bound combines restricted-master error with pricing uncertainty and, when used as L_M, makes $U(M) - L_M$ an upper bound on integer suboptimality. It thus supports evaluating integer plans alongside LP-based pricing feedback, with the proof and error decomposition in Appendix C.

For RLOO, weighted block losses and step/error costs form $D(\tau)$ (Appendix D). Responses with at least one scored block receive a bounded process reward that complements final feasibility and objective quality through

$$P(\tau) = \mathbf{1}_{\mathrm{eligible}}\frac{s_P}{s_P + D(\tau)}, \qquad R(\tau) = R_F(\tau) + \lambda_P P(\tau). \tag{8}$$

5 Experiments

5.1 Experiment Settings

We evaluate six problem families also studied in NLCO (Jiang et al., 2026): bin packing (BPP), job-shop scheduling (JSSP), capacitated vehicle routing (CVRP), traveling salesman with time windows

Table 1: Performance across six tasks.

Method	BPP				JSSP				CVRP				TSPTW				CSP				GCP			
	FR↑	Gap↓	Opt.↑	Q_5 ↑	FR↑	Gap↓	Opt.↑	Q_5 ↑	FR↑	Gap↓	Opt.↑	Q_5 ↑	FR↑	Gap↓	Opt.↑	Q_5 ↑	FR↑	Gap↓	Opt.↑	Q_5 ↑	FR↑	Gap↓	Opt.↑	Q_5 ↑
Conventional optimization																								
SCIP	100.00%	0.00%	100.00%	100.00%	100.00%	0.95%	83.33%	91.33%	100.00%	71.19%	67.79%	71.14%	68.00%	1.10%	60.00%	64.00%	94.67%	0.31%	88.67%	92.00%	100.00%	0.00%	100.00%	100.00%
Direct prompting																								
qwen 3.5-9B	1.33%	0.00%	1.33%	1.33%	0.00%	n/a	0.00%	0.00%	27.52%	35.93%	0.00%	0.00%	11.33%	12.32%	2.67%	3.33%	0.00%	n/a	0.00%	0.00%	0.00%	n/a	0.00%	0.00%
Qwen-3.8-max	96.67%	0.37%	94.67%	94.67%	32.00%	3.67%	26.00%	26.67%	77.18%	22.84%	10.07%	21.48%	48.67%	9.27%	15.33%	24.67%	43.33%	1.14%	39.33%	39.33%	90.67%	2.19%	88.00%	88.00%
External-feedback methods (non-end-to-end)																								
OPRO (Yang et al., 2024)	72.00%	4.38%	54.67%	54.67%	65.00%	83.35%	—	—	21.00%	37.03%	—	—	24.67%	5.26%	14.67%	19.33%	9.33%	4.55%	7.33%	7.33%	76.00%	8.08%	56.00%	56.00%
LMEA (Liu et al., 2024b)	72.00%	7.46%	46.67%	46.67%	44.00%	83.19%	—	—	24.00%	61.24%	—	—	28.00%	6.75%	12.00%	19.33%	8.00%	4.34%	6.67%	6.67%	66.67%	10.23%	49.33%	49.33%
LLMs as end-to-end solvers																								
SFT	70.67%	3.25%	59.33%	59.33%	2.67%	13.63%	0.67%	1.33%	11.41%	9.21%	2.01%	5.37%	36.00%	3.30%	19.33%	28.00%	42.00%	6.98%	31.74%	31.74%	41.33%	1.61%	39.33%	39.33%
CG-CoT SFT	68.75%	3.27%	62.50%	62.50%	34.38%	15.24%	18.75%	34.38%	50.00%	12.89%	18.75%	29.69%	56.25%	4.51%	18.75%	18.75%	37.50%	6.46%	18.75%	32.81%	31.33%	1.92%	31.25%	31.33%
FOARL (Jiang et al., 2025a)	78.67%	4.24%	62.67%	62.67%	13.67%	11.24%	9.53%	9.53%	12.75%	9.93%	1.34%	5.37%	34.67%	3.31%	17.33%	26.00%	76.67%	6.21%	32.67%	37.33%	46.25%	2.56%	39.33%	41.33%
Our Method	87.50%	0.00%	87.50%	87.50%	62.50%	7.29%	29.69%	45.31%	71.88%	11.68%	15.63%	34.38%	59.38%	4.18%	33.33%	37.50%	73.44%	5.25%	43.75%	65.63%	65.60%	1.35%	50.00%	58.33%
Our Method+BoN	100.00%	2.25%	93.75%	93.75%	76.56%	6.84%	35.94%	46.88%	87.50%	9.34%	15.63%	39.06%	67.19%	3.22%	37.50%	43.75%	81.25%	5.78%	43.75%	75.00%	71.88%	1.02%	54.69%	64.58%

(TSPTW), cutting stock (CSP), and graph coloring (GCP). All problems can be solved exactly using a standard branch-and-price (B&P) framework. Their native constraints and formulations, together with details of the corresponding B&P algorithms, are provided in Appendix G.

We apply exact optimization to all benchmark instances and record certified optimal objective values for approximately 99% of them. For the remaining 1%, we record the best available feasible upper bound. For solution cost U and reference objective $U^{\mathrm{ref}} > 0$, we report four metrics:

- **FR:** Percentage of all instances with feasible submissions.
- **Gap:** Mean signed $100(U - U^{\mathrm{ref}})/U^{\mathrm{ref}}$ over feasible outputs (undefined if none).
- **Opt.:** Percentage of all instances attaining the reference with feasible $U \leq U^{\mathrm{ref}}$.
- Q_5**:** Percentage of all instances with feasible $U \leq \lceil 1.05U^{\mathrm{ref}} \rceil$.

Base uses Qwen3.5-9B, with SFT, CG-CoT SFT, and FOARL fine-tuned from it. Qwen3.8-Max is used for large-model evaluation, OPRO (Yang et al., 2024), and LMEA (Liu et al., 2024b).

Evaluation uses 128 randomly sampled problems from NLCO (Jiang et al., 2026), approximately balanced across scales. OPRO and LMEA use all 150 benchmark instances. We adapt prompts using FOARL's (Jiang et al., 2025a) task-specific heuristic features and cap generation at 16,384 tokens for all methods. We additionally report best-of-N sampling results ($N = 8$) for our method.

5.2 Main Results

OR supervision improves both feasibility and solution quality. Table 1 shows that our method achieves the highest reported Q_5 on all six tasks among the single-response solvers trained from Qwen3.5-9B, outperforming SFT, CG-CoT SFT, and FOARL. Its task-level macro-average FR and Q_5 reach 70.05% and 54.77%, compared with 46.37% and 34.91% for CG-CoT SFT. The improvement extends beyond producing valid outputs. On CSP, FOARL has a slightly higher FR (76.67% versus 73.44%), but our method raises Q_5 from 37.33% to 65.63%. Against Qwen3.8-Max, our method has higher Q_5 on JSSP, CVRP, TSPTW, and CSP. These results indicate that task-specific OR supervision can improve the optimization capability of a smaller model, with benefits that depend on the problem structure.

Learning the algorithm requires more than imitating its traces. The contrast between SFT and CG-CoT SFT suggests that an explicit decomposition is particularly useful for coordinating interdependent decisions. Their reported FR rises from 2.67% to 34.38% on JSSP and from 11.41% to 50.00% on CVRP, but falls on BPP, CSP, and GCP. Thus, serializing solver trajectories does not by itself ensure reliable autonomous execution. Our subsequent RDO–RLOO training improves both FR and Q_5 over CG-CoT SFT on every task. On TSPTW, FR changes modestly from 56.25% to 59.38%, while Q_5 doubles from 18.75% to 37.50%. This pattern is consistent with the complementary roles of our training objectives: RDO compares alternative column proposals and integer plans at the same prefix, while RLOO trains their coordination across the complete response. The aggregate results support this interpretation of the combined procedure, with component comparisons provided in Section 5.3.

Table 2: CSP ablation results by problem size.

Method	Scale	FR↑	Gap↓	Opt.↑	Q_5 ↑
CG-CoT SFT	Small	59.09%	4.53%	40.91%	50.00%
	Medium	42.86%	7.74%	9.52%	28.57%
	Large	9.52%	17.24%	4.76%	19.05%
CG-CoT SFT + RDO	Small	63.64%	2.98%	54.55%	54.55%
	Medium	52.38%	0.00%	52.38%	52.38%
	Large	14.29%	14.58%	0.00%	0.00%
CG-CoT SFT + OR-RLOO + RTG	Small	72.73%	1.46%	68.18%	72.73%
	Medium	47.62%	5.82%	14.29%	23.81%
	Large	4.76%	19.31%	0.00%	0.00%
Our Method	Small	95.45%	1.15%	90.91%	95.45%
	Medium	85.71%	2.39%	33.33%	76.19%
	Large	38.10%	10.50%	4.76%	23.81%

Best-of-eight sampling improves coverage, with task-dependent quality gains. BoN raises macro-average FR to 80.73% and Q_5 to 60.50%, with improvements in both metrics on every task. BPP reaches 100.00% FR and 93.75% Q_5, while CVRP reaches 87.50% FR but only 39.06% Q_5. Additional candidates therefore help recover feasible solutions without eliminating the difficulty of finding high-quality ones. The higher BPP Gap under BoN (2.25%) is compatible with this coverage gain, since newly feasible submissions enter the average. Each candidate is generated autonomously, with additional inference computation and selection for BoN.

5.3 Ablation Studies

Table 2 examines RDO and OR-RLOO+RTG on CSP across three scales. RDO raises overall FR from 37.50% to 43.75% across 128 instances and reduces conditional Gap at every scale, reaching 0.00% on medium instances. These gains are consistent with same-prefix OR comparisons improving column construction and plan assembly. However, large-scale Q_5 remains 0.00%, indicating that local improvements alone do not ensure reliable coordination. OR-RLOO+RTG raises small-instance FR and Q_5 to 72.73%, but reaches only 4.76% FR and 19.31% Gap on large instances. This pattern suggests that complete-response rewards help coordinate shorter trajectories, while longer ones remain difficult to optimize through trajectory feedback alone. Together, the results motivate combining local operation supervision with complete-response learning. The variants assess whole training stages and do not isolate RTG or individual reward terms.

Alternating RDO and RLOO raises overall FR from 37.50% to 73.44%, exceeding the stronger single-stage variant by 29.69 percentage points, and achieves an overall Q_5 of 65.63%. These gains support complementary roles: RDO refines local column construction and plan assembly, while RLOO trains their coordination over complete responses. Recollection then refreshes local comparisons under the updated policy. The full method also lowers conditional Gap at every scale relative to CG-CoT SFT. RDO alone's zero medium-scale Gap is conditional on only 52.38% feasible outputs, whereas the joint method reaches 85.71% FR and 76.19% Q_5. The remaining large-scale FR of 38.10% suggests that coordinating local operations over longer trajectories remains an important challenge.

5.4 Illustrative Optimization Trajectories

Figure 2 compares generated cost U_t with the restricted integer optimum $v_I(\mathcal{C}_t)$ defined in Equation (3). Their separation measures improvement available through better assembly. Arrows compare next-batch columns with the pricing optimum under the same current LP dual: downward arrows indicate negative reduced costs and diamonds indicate zero. Appendix F.2 details the offline computations and arrow scaling.

These trajectories show how the learned policy can turn local column proposals into improved complete solutions. Negative reduced costs identify columns favored by the current dual prices. Realizing an objective reduction also requires the model to combine columns into a feasible integer plan.

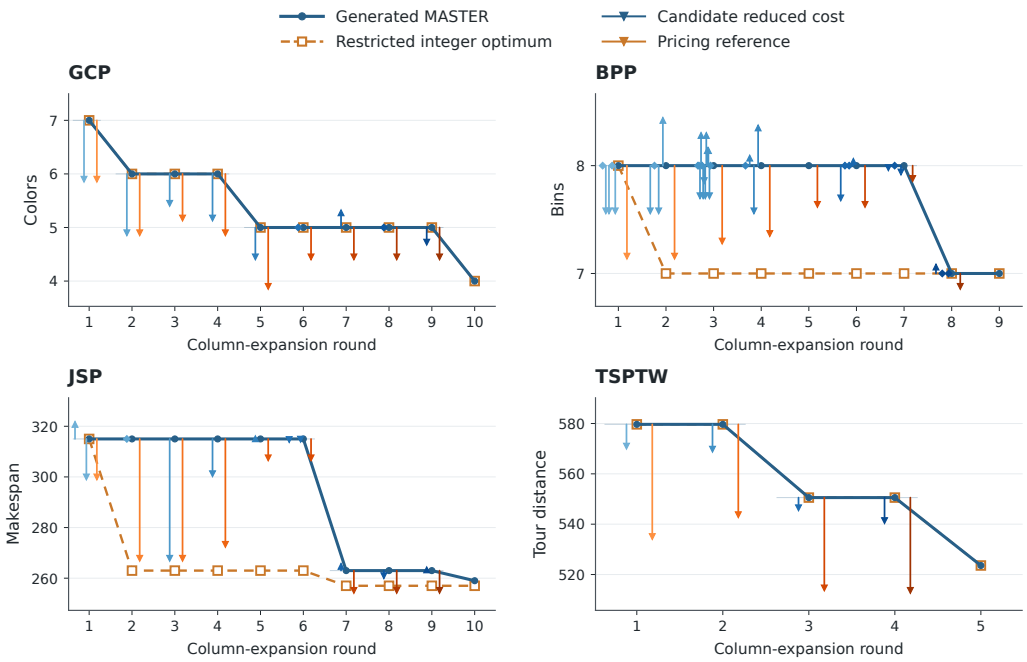


Figure 2: Single-response optimization trajectories. The solid blue curve shows generated MASTER cost U_t, and the dashed orange curve shows the restricted integer optimum $v_I(\mathcal{C}_t)$. Each blue arrow represents one accepted next-batch column. The orange arrow represents the full-domain pricing minimum under the same current-prefix dual. Darker shades denote later rounds.

Because pool expansion changes the dual prices, the arrows assess proposals at their respective decision points, while the curves track actual integer progress. In GCP and TSPTW, six of nine and all four new columns have negative reduced costs, respectively, and the coinciding curves show that the model fully realizes the best solution available in each successive pool. BPP reveals why column discovery alone is insufficient: the pool supports seven bins at round 2, but the model constructs that solution only at round 8. The remaining separation in JSP likewise identifies room to improve assembly even with the existing columns. These observations support the complementary supervision in our method: RDO refines both column proposals and plan assembly at fixed prefixes, while RLOO trains the policy to coordinate these operations across a complete response.

5.5 Learning Signals from Process Rewards

Figure 3 examines how process rewards distinguish responses with identical terminal rewards. Across 60 TSPTW training cycles, we analyze 480 groups, each containing four responses to one instance. Of the 35 groups with equal terminal rewards, process feedback yields nonzero sequence advantages in 27, covering 77.1% of these groups and 5.6% of all groups. Panel (a) measures this distinction by the root-mean-square advantage within each group. The signal occurs in every ten-cycle window in panel (b), and 21 of the 27 groups contain four infeasible final solutions. Thus, process feedback supplies training information even when terminal rewards cannot distinguish unsuccessful responses.

Panels (c,d) trace this signal from intermediate behavior to the sequence advantages used in training. Responses B and C each contain 12 operations and use 1,306 and 1,302 tokens, respectively, yet their valid `COLUMNS` counts are 6/6 and 0/6. Their step costs are identical, while C incurs 18 units of invalid-column penalties, giving total process losses of 3.43 for B and 21.44 for C. All four responses in this group receive the same terminal contribution of -0.4, so their terminal advantages are zero. With process feedback, B and C receive recorded sequence advantages of $+0.0387$ and -0.0415, respectively. This contrast is driven primarily by local validity and cannot be explained by action count alone. It illustrates how our process supervision allows RLOO to favor valid intermediate

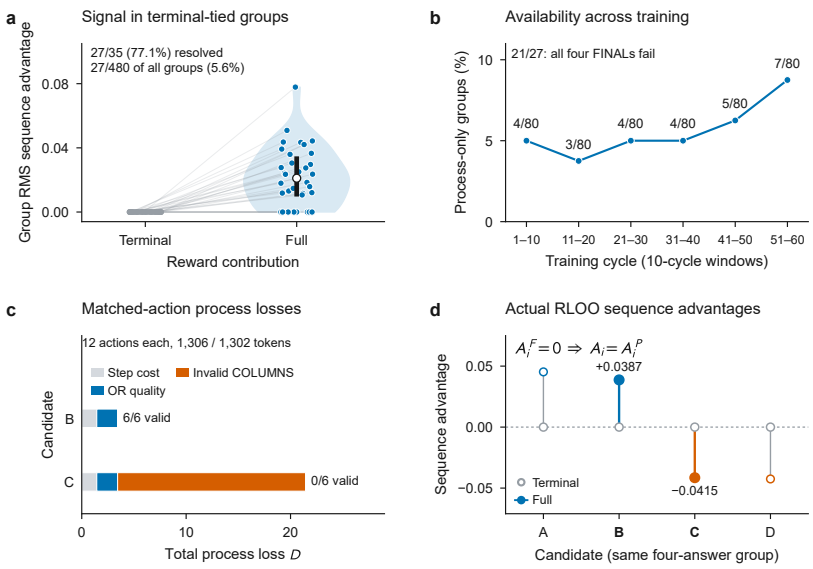


Figure 3: Process rewards in TSPTW. (a) RMS advantages for 35 groups with equal terminal rewards. (b) Process-only discrimination in 80-group windows. (c) Loss components for two responses with equal action counts. (d) RLOO advantages in their four-response group. Terminal and Full compare reward components on the same responses.

constructions even when every final solution fails. Appendix F.3 provides the reward decomposition and calculation details.

6 Conclusion

We presented an OR-supervised framework that uses column generation and branch-and-price structures to decompose combinatorial optimization into learnable column proposals and integer-plan assembly. CG-CoT and joint RDO–RLOO training connect local mathematical feedback with complete-solution learning.

Experiments across six task families demonstrate improved autonomous solution quality. Trajectory and reward analyses illustrate how intermediate optimization decisions support solution refinement and provide additional learning signals, offering a direction for long-horizon task decomposition and decision-making in agents.

AI use statement

Generative AI tools were used to assist with revising mathematical propositions and proofs, developing code, translating text, and polishing the language of the manuscript. They were not used to generate or rewrite corpus, or to clean or reformat datasets. No other generative AI assistance was used in conducting or reporting this research beyond the activities disclosed above.

The authors independently reviewed and verified all AI-assisted contributions, including revisions to mathematical propositions and proofs, code, translations, and edited text. The authors take full responsibility for the accuracy, originality, and integrity of the manuscript and accompanying research artifacts, including all AI-assisted content.

REFERENCES

Abdo Abouelrous, Laurens Bliek, Adriana F. Gabor, Yaoxin Wu, and Yingqian Zhang. Reinforcement learning for solving the pricing problem in column generation for routing. *Operations Research Perspectives*, 15:100364, 2025. doi: 10.1016/j.orp.2025.100364. URL http://dx.doi.org/10.1016/j.orp.2025.100364.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 577–596. PMLR, 2024. URL https://proceedings.mlr.press/v235/ahmaditeshnizi24a.html.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. doi: 10.1287/opre.46.3.316. URL https://doi.org/10.1287/opre.46.3.316.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021. doi: 10.1016/j.ejor.2020.07.063. URL http://dx.doi.org/10.1016/j.ejor.2020.07.063.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. GraphWiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 353–364. ACM, 2024. doi: 10.1145/3637528.3672010. URL http://dx.doi.org/10.1145/3637528.3672010.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/131f383b434fdf48079bff1e44e2d9a5-Paper.pdf.

Cheng Chi, Amine Mohamed Aboussalah, Elias B. Khalil, Juyoung Wang, and Zoha Sherkat-Masoumi. A deep reinforcement learning framework for column generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9633–9644. Curran Associates, Inc., 2022. doi: 10.52202/068431-0700. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3ecfe5c632afb7d96a2337b18ff99b1f-Paper-Conference.pdf.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=9SkkifLopZ.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi: 10.1287/opre.8.1.101. URL https://pubsonline.informs.org/doi/10.1287/opre.8.1.101.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, and Yongfeng Zhang. NPHardEval: Dynamic benchmark on reasoning ability of large language models via complexity classes. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pp. 4092–4114. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.225. URL https://aclanthology.org/2024.acl-long.225/.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations*, pp. 43909–43934, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/bf72f65f30eedf5d48da6980ee02b589-Paper-Conference.pdf.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (SoS): Learning to search in language. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=2cop2jmQVL.

Maxime Gasse, Didier Chetelat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/d14c2267d848abeb81fd590f371d39bd-Paper.pdf.

Jacek Gondzio, Pablo Gonzalez-Brevis, and Pedro Munari. New developments in the primal-dual column generation technique. *European Journal of Operational Research*, 224(1):41–51, January 2013. doi: 10.1016/j.ejor.2012.07.024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Yi-Xiang Hu, Feng Wu, Shaoang Li, Yifang Zhao, and Xiang-Yang Li. FFCG: Effective and fast family column generation for solving large-scale linear program. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 11238–11245. Association for the Advancement of Artificial Intelligence (AAAI), 2025. doi: 10.1609/aaai.v39i11.33222. URL http://dx.doi.org/10.1609/aaai.v39i11.33222.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. ORLM: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 73(6):2986–3009, 2025a. doi: 10.1287/opre.2024.1233. URL http://dx.doi.org/10.1287/opre.2024.1233.

Zixiao Huang, Lifeng Guo, Wenhao Li, Junjie Sheng, Chuyun Shen, Haosheng Chen, Bo Jin, Changhong Lu, and Xiangfeng Wang. GraphThought: Graph combinatorial optimization with thought generation. *arXiv preprint arXiv:2502.11607*, 2025b. doi: 10.48550/arxiv.2502.11607. URL https://arxiv.org/abs/2502.11607.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. Large language models as end-to-end combinatorial optimization solvers. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 164787–164826. Curran Associates, Inc., 2025a. doi: 10.52202/085713-5495. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f0c6c93e1dbff84daf6082fef7e1a094-Paper-Conference.pdf.

Xia Jiang, Yaoxin Wu, Chenhao Zhang, and Yingqian Zhang. DRoC: Elevating large language models for complex vehicle routing via decomposed retrieval of constraints. In *International Conference on Learning Representations*, pp. 46731–46752, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/73d6c3e4b214deebbbf8256e26d2cf45-Paper-Conference.pdf.

Xia Jiang, Jing Chen, Cong Zhang, Jie Gao, Chengpeng Hu, Chenhao Zhang, Yaoxin Wu, and Yingqian Zhang. Reasoning in a combinatorial and constrained world: Benchmarking LLMs on natural-language combinatorial optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 30592–30648. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1529. URL https://aclanthology.org/2026.findings-acl.1529/.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d9896106ca98d3d05b8cbdf4fd8b13a1-Paper.pdf.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Pavlína Koutecká, Přemysl Šůcha, Jan Hůla, and Broos Maenhout. A machine learning approach to rank pricing problems in branch-and-price. *European Journal of Operational Research*, 320(2): 328–342, 2025. doi: 10.1016/j.ejor.2024.07.029. URL http://dx.doi.org/10.1016/j.ejor.2024.07.029.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/f231f2107df69eab0a3862d50018a9b2-Paper.pdf.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul McVay, Michael Rabbat, and Yuandong Tian. Beyond A*: Better planning with transformers via search dynamics bootstrapping. In *Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=SGoVIC0u0f.

Xiaozhe Li, Xinyu Fang, Shengyuan Ding, Yang Li, Linyang Li, Haodong Duan, Qingwen Liu, and Kai Chen. Forge: Quality-aware reinforcement learning for NP-hard optimization in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 28351–28368. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1413. URL https://aclanthology.org/2026.findings-acl.1413/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223. PMLR, 2024a. URL https://proceedings.mlr.press/v235/liu24bs.html.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, 2024b. doi: 10.1109/CEC60901.2024.10611913. URL https://ieeexplore.ieee.org/document/10611913.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, pp. 52989–53046, 2024c. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Yunqiao Yang, Junting Pan, Mingjie Zhan, and Hongsheng Li. Step-controlled DPO: Leveraging stepwise errors for enhancing mathematical reasoning of language models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=jp1AdIcKTj.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234. URL https://doi.org/10.1287/opre.1050.0234.

Mouad Morabit, Guy Desaulniers, and Andrea Lodi. Machine-learning–based column selection for column generation. *Transportation Science*, 55(4):815–831, 2021. doi: 10.1287/trsc.2021.1045. URL http://dx.doi.org/10.1287/trsc.2021.1045.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL http://dx.doi.org/10.1038/s41586-023-06924-6.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. RL on incorrect synthetic data scales the efficiency of LLM math reasoning by eight-fold. In *Advances in Neural Information Processing Systems*, volume 37, pp. 43000–43031. Curran Associates, Inc., 2024. doi: 10.52202/079017-1361. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4b77d5b896c321a29277524a98a50215-Paper-Conference.pdf.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, pp. 60808–60838, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/98711dea460bdefe0e651ca23ec98ba2-Paper-Conference.pdf.

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. Association for the Advancement of Artificial Intelligence (AAAI), 2022. doi: 10.1609/aaai.v36i9.21230. URL http://dx.doi.org/10.1609/aaai.v36i9.21230.

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 *Proceedings of Machine Learning Research*, pp. 44741–44758. PMLR, 2024. URL https://proceedings.mlr.press/v235/shen24e.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Shuai Song, Shuo Yang, Ran Song, Shilei Chu, Yibin Li, and Wei Zhang. Towards online 3D bin packing: Learning synergies between packing and unpacking via DRL. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1136–1145. PMLR, 2023. URL https://proceedings.mlr.press/v205/song23a.html.

Andy (DiJia) Su, Sainbayar Sukhbaatar, Michael Rabbat, Yuandong Tian, and Qinqing Zheng. Dualformer: Controllable fast and slow thinking by learning with randomized reasoning traces. In *International Conference on Learning Representations*, pp. 95080–95117, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/ed45d6a03de84cc650cae0655f699356-Paper-Conference.pdf.

Jianheng Tang, Qifan Zhang, Yuhan Li, Nuo Chen, and Jia Li. GraphArena: Evaluating and exploring large language models on graph computation. In *International Conference on Learning Representations*, pp. 48118–48145, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/77fa8253adfc8b33209639f3e9985741-Paper-Conference.pdf.

Niki van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29 (2):331–345, 2025. doi: 10.1109/tevc.2024.3497793. URL http://dx.doi.org/10.1109/TEVC.2024.3497793.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yilin Wang, Heng Zhou, Dongxing Mao, Linjie Li, Jingru Tan, Haochen Han, Zhengyuan Yang, Alex Jinpeng Wang, and Min Li. OR-PRM: A process reward model for algorithmic problem in operations research. In *International Conference on Learning Representations*, pp. 42343–42369,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4624803ce92f6c44d8cabe085d8d9d96-Paper-Conference.pdf.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *International Conference on Learning Representations*, pp. 48519–48537, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/d45ee77826332c100a1e15f7765b99ff-Paper-Conference.pdf.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54590–54613. PMLR, 2024. URL https://proceedings.mlr.press/v235/xie24j.html.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, pp. 12028–12068, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3339f19c5fcee3ad74502947a32be9e6-Paper-Conference.pdf.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. In *International Conference on Learning Representations*, pp. 24726–24759, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/3deb687c44d3687ace0729e5db3b4efd-Paper-Conference.pdf.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 43571–43608. Curran Associates, Inc., 2024. doi: 10.52202/079017-1381. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/4ced59d480e07d290b6f29fc8798f195-Paper-Conference.pdf.

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. Association for the Advancement of Artificial Intelligence (AAAI), 2024. doi: 10.1609/aaai.v38i8.28661. URL http://dx.doi.org/10.1609/aaai.v38i8.28661.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488. Curran Associates, Inc., 2022. doi: 10.52202/068431-1126. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/639a9a172c044fbb64175b5fad42e9a5-Paper-Conference.pdf.

Jian Zhao, Runze Liu, Kaiyan Zhang, Zhimu Zhou, Junqi Gao, Dong Li, Jiafei Lyu, Zhouyi Qian, Biqing Qi, Xiu Li, and Bowen Zhou. GenPRM: Scaling test-time compute of process reward models via generative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34932–34940. Association for the Advancement of Artificial Intelligence (AAAI), 2026. doi: 10.1609/aaai.v40i41.40797. URL http://dx.doi.org/10.1609/aaai.v40i41.40797.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte Carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78338–78373. PMLR, 2025. URL https://proceedings.mlr.press/v267/zheng25o.html.

Chenyu Zhou, Tianyi Xu, Jianghao Lin, and Dongdong Ge. StepORLM: A self-evolving framework with generative process supervision for operations research language models. In *International Conference on Learning Representations*, pp. 6914–6940, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0bcfb525c8f8f07ae10a93d0b2a40e00-Paper-Conference.pdf.

A Background on Column Generation and Branch-and-Price

A.1 Decomposable problems and configuration masters

Consider a bounded, feasible integer optimization problem

$$\min_{x_k \in X_k} \sum_{k=1}^{K} c_k^\top x_k \quad \text{subject to} \quad \sum_k A_k x_k \geq b, \qquad \sum_k E_k x_k = f, \tag{A.1}$$

where each X_k is a finite set of locally feasible integer configurations. The linking constraints coordinate the local choices. For $p \in \mathcal{P}_k$, let $x_{kp} \in X_k$, $a_{kp} = A_k x_{kp}$, $e_{kp} = E_k x_{kp}$, and $c_{kp} = c_k^\top x_{kp}$. The configuration LP is

$$\begin{aligned} z_L^\star = \min_{\lambda \geq 0} \quad & \sum_{k,p} c_{kp} \lambda_{kp} \\ \text{subject to} \quad & \sum_{k,p} a_{kp} \lambda_{kp} \geq b, \qquad \sum_{k,p} e_{kp} \lambda_{kp} = f, \\ & \sum_{p \in \mathcal{P}_k} \lambda_{kp} = m_k, \qquad k = 1, \ldots, K. \end{aligned} \tag{A.2}$$

For Eq. (A.1), $m_k = 1$. More general nonnegative group masses m_k are allowed when the task formulation supplies the corresponding interpretation. With all legal configurations and $m_k = 1$, imposing binary λ_{kp} recovers Eq. (A.1). We denote the optimum under the task's integer restrictions by $z_I^\star$.

A.2 Restricted masters, dual variables, and pricing

An RMP restricts Eq. (A.2) to finite pools $\mathcal{C}_k \subseteq \mathcal{P}_k$. Its primal solutions extend to the full master by setting omitted coefficients to zero. Let $y \geq 0$, ν, and σ_k be the dual multipliers for the inequality, equality, and group rows. The full dual maximizes

$$d(y, \nu, \sigma) = b^\top y + f^\top \nu + \sum_k m_k \sigma_k \quad \text{subject to} \quad a_{kp}^\top y + e_{kp}^\top \nu + \sigma_k \leq c_{kp} \quad \forall k, p. \tag{A.3}$$

The equality and group multipliers are unrestricted in sign. The reduced cost of a configuration and the pricing optimum are

$$r_{kp} = c_{kp} - a_{kp}^\top y - e_{kp}^\top \nu - \sigma_k, \qquad \rho_k^\star = \min_{p \in \mathcal{P}_k} r_{kp}. \tag{A.4}$$

An RMP-feasible dual satisfies these inequalities only for its restricted columns. A negative reduced cost identifies a violated full-dual constraint. Column generation adds such configurations and reoptimizes the RMP.

A feasible pricing candidate supplies an *upper* bound on the minimum $\rho_k^\star$. Certification requires a lower bound $\underline{\rho}_k \leq \rho_k^\star$, obtained, for example, by a valid relaxation. If an RMP primal–dual pair is optimal and pricing certifies nonnegative reduced costs over every full domain, that pair is optimal for the full LP.

A.3 Branch-and-price and bound scope

Branch-and-price applies column generation within branch-and-bound (Barnhart et al., 1998). Branch restrictions must be represented consistently in the node master and pricing domains. A certified node lower bound applies to that node's feasible region. To obtain a lower bound for the original integer problem, the open-node frontier must cover the unresolved feasible region and all pruning decisions must be valid. With a global feasible incumbent U, one can then aggregate the frontier bounds as

$$L_{\text{global}} = \min\big(\{U\} \cup \{L_n : n \text{ is open}\}\big). \tag{A.5}$$

CG-CoT evaluation uses the fixed original formulation.

B CG-CoT Representation and Demonstration Construction

B.1 Nested integer masters and certificate interpretation

Let $\mathcal{P}$ denote the union of all legal configuration groups. For a complete plan x, $\lambda(x)$ denotes its configuration coefficients, and any auxiliary variables remain part of x. Fix the original feasible set $\mathcal{X}_I$ and objective f throughout the trajectory. The restricted feasible set $\mathcal{X}_I(\mathcal{C})$ and optimum $v_I(\mathcal{C})$ are defined in Eq. (3). For scheduling and time-window problems, compatibility and timing constraints are retained in $\mathcal{X}_I$. They are not replaced by local column checks.

Proposition 2 (Nested-pool monotonicity)**.** *If $\mathcal{C}_t \subseteq \mathcal{C}_{t+1} \subseteq \mathcal{P}$ and the initial restricted problem is feasible with a finite optimum, then*

$$z_I^\star \le v_I(\mathcal{C}_{t+1}) \le v_I(\mathcal{C}_t). \tag{B.1}$$

If $\mathcal{C}_T$ contains the support of some globally optimal plan, then $v_I(\mathcal{C}_T) = z_I^\star$.

Proof. Every feasible plan using $\mathcal{C}_t$ remains feasible using $\mathcal{C}_{t+1}$. In coefficient form, omitted column variables are set to zero, with all preceding coefficients and auxiliary variables unchanged. Thus $\mathcal{X}_I(\mathcal{C}_t) \subseteq \mathcal{X}_I(\mathcal{C}_{t+1}) \subseteq \mathcal{X}_I$, which proves the inequalities for minimization. If a global optimum belongs to $\mathcal{X}_I(\mathcal{C}_T)$, minimizing over that set gives $v_I(\mathcal{C}_T) \le z_I^\star$. The opposite inequality follows from restriction. □

This argument applies to integer reassembly over a growing pool. Changing branch constraints would also invalidate the required feasible-set inclusion. The discovery tree therefore supplies columns to the demonstrations, but its node restrictions are not transferred to successive `MASTER` problems.

Time-limited teacher solves. The actual teacher need not attain $v_I(\mathcal{C}_t)$ at every round. If U_t is its retained feasible objective and U_{new} is a subsequent feasible proposal over $\mathcal{C}_{t+1}$, the retention rule gives

$$U_{t+1} = \min\{U_t, U_{\text{new}}\} \le U_t, \qquad U_t \ge v_I(\mathcal{C}_t). \tag{B.2}$$

When no new feasible proposal is found, the previous plan is retained. For the teacher output itself to attain $z_I^\star$, the final restricted solve must be exact or the retained witness must independently certify that value.

What a trajectory certifies. A verified feasible plan certifies an upper bound U_t on the original minimization problem. For any valid *global* lower bound L_t and feasible proposal of cost U_t in the current pool, weak ordering gives

$$L_t \le z_I^\star \le v_I(\mathcal{C}_t) \le U_t, \qquad 0 \le U_t - z_I^\star \le U_t - L_t. \tag{B.3}$$

If $U_t = L_t$, the plan is globally optimal. Otherwise $U_t - L_t$ bounds its absolute suboptimality. The role of CG-CoT is to expose checkable primal objects and their nested search spaces.

B.2 WORKED EXAMPLES ACROSS THE SIX PROBLEM FAMILIES

Each box pairs a task instruction and a small constructed instance with a complete two-round CG-CoT response. Natural-language reasoning accompanies explicit column declarations and integer plans. A later column batch permits revision of the initial plan, followed by a task-native final solution. The following examples illustrate CG-CoT.

CG-CoT Example: Cutting Stock (CSP)

Instruction. Meet every item demand using as few stock bars as possible. Return one cutting pattern per used bar.

Input. Stock length: 10. Item lengths: $w_A = 6$, $w_B = 4$. Demands: $d_A = d_B = 2$. Overproduction is permitted.

CG-CoT Response.

<think>

<OR:COLUMNS>

I initialize two reusable patterns. Their used lengths are 6 and 8, both at most 10.
{"columns":[{"id":"p1","pattern":{"A":1}},{"id":"p2","pattern":{"B":2}}]}
</OR:COLUMNS>

<OR:MASTER>

I use the first pattern twice and the second once. Three bars produce two units of each item, meeting both demands.
{"solution":[{"A":1},{"A":1},{"B":2}],"total_stock_used":3}
</OR:MASTER>

<OR:COLUMNS>

A mixed pattern uses 6 + 4 = 10 units of length and can be reused to satisfy the order.
{"columns":[{"id":"p3","pattern":{"A":1,"B":1}}]}
</OR:COLUMNS>

<OR:MASTER>

I replace the three-bar plan with two copies of the mixed pattern. Production is unchanged, and the bar count decreases to two.
{"solution":[{"A":1,"B":1},{"A":1,"B":1}],"total_stock_used":2}
</OR:MASTER>

</think>

<OR:FINAL>

I submit the revised two-bar plan. Total required length is 20, so two bars also match the volume lower bound.
{"solution":[{"A":1,"B":1},{"A":1,"B":1}],"total_stock_used":2}
</OR:FINAL>

CG-CoT Example: Bin Packing (BPP)

Instruction. Pack each item exactly once without exceeding bin capacity, and minimize the number of bins.

Input. Bin capacity: 10. Distinct items (A, B, C, D) have weights $(6, 6, 4, 4)$.

CG-CoT Response.

<think>

<OR:COLUMNS>

I initialize one singleton pattern for each item. Every pattern fits within the bin capacity.

```
{"columns":[
  {"id":"p1","pattern":["A"]},
  {"id":"p2","pattern":["B"]},
  {"id":"p3","pattern":["C"]},
  {"id":"p4","pattern":["D"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

The four singleton bins cover every item exactly once.

```
{"solution":[["A"],["B"],["C"],["D"]],"total_bins":4}
</OR:MASTER>
```

<OR:COLUMNS>

I add two paired patterns. Each has weight 6 + 4 = 10 and is therefore a legal bin.

```
{"columns":[
  {"id":"p5","pattern":["A","C"]},
  {"id":"p6","pattern":["B","D"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I select the two paired patterns instead of the singleton bins. Every item still occurs exactly once.

```
{"solution":[["A","C"],["B","D"]],"total_bins":2}
</OR:MASTER>
```

</think>

<OR:FINAL>

I submit the two-bin packing. Its bin count matches the total-weight lower bound of 20 / 10 = 2.

```
{"solution":[["A","C"],["B","D"]],"total_bins":2}
</OR:FINAL>
```

CG-CoT Example: Vehicle Routing (CVRP)

Instruction. Serve each customer exactly once using capacity-feasible depot-to-depot routes, minimizing total Euclidean distance.

Input. Depot: $0 = (0, 0)$. Customers: $A = (1, 0)$, $B = (2, 0)$, $C = (0, 1)$, $D = (0, 2)$. Each demand is 1, and vehicle capacity is 2.

CG-CoT Response.

<think>

<OR:COLUMNS>

I initialize a route for each customer. Every route starts and ends at the depot and has load one.

```
{"columns":[
  {"id":"p1","route":[0,"A",0]},
  {"id":"p2","route":[0,"B",0]},
  {"id":"p3","route":[0,"C",0]},
  {"id":"p4","route":[0,"D",0]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I select all four routes. Each customer is served once, with total distance 2 + 4 + 2 + 4 = 12.

```
{"solution":[[0,"A",0],[0,"B",0],[0,"C",0],[0,"D",0]]}
</OR:MASTER>
```

<OR:COLUMNS>

I add one route through A and B, and another through C and D. Each route has load two and length four.

```
{"columns":[
  {"id":"p5","route":[0,"A","B",0]},
  {"id":"p6","route":[0,"C","D",0]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I replace the four singleton routes with these two routes. Coverage is unchanged and total distance decreases to eight.

```
{"solution":[[0,"A","B",0],[0,"C","D",0]]}
</OR:MASTER>
```

</think>

<OR:FINAL>

I submit the revised complete route set.

```
{"solution":[[0,"A","B",0],[0,"C","D",0]]}
</OR:FINAL>
```

CG-CoT Example: Time-Window Routing (TSPTW)

Instruction. Visit each non-depot node exactly once and return to the depot within all time windows, minimizing Euclidean tour length.

Input. Nodes $(0, 1, 2, 3)$: $(0, 0), (3, 0), (3, 4), (0, 4)$. Time windows: $[0, 30], [0, 15], [0, 16], [0, 24]$. Travel time equals distance, service time is zero, and waiting is allowed. Source blocks: $S_0 = (0, 1)$ and $S_1 = (2, 3)$.

CG-CoT Response.

<think>

<OR:COLUMNS>

I declare the source partition and one successor pattern per block. Together they form the tour 0, 2, 1, 3, 0.

```
{"blocks":[[0,1],[2,3]],
 "columns":[
  {"id":"p1","block":0,"successors":[2,3]},
  {"id":"p2","block":1,"successors":[1,0]}]}
</OR:COLUMNS>
```

<OR:MASTER>

The complete tour visits every node and respects the time windows. Its distance is 5 + 4 + 5 + 4 = 18.

```
{"solution":[0,2,1,3,0],"total_distance":18}
</OR:MASTER>
```

<OR:COLUMNS>

I add two successor patterns that connect the nodes around the rectangle.

```
{"columns":[
  {"id":"p3","block":0,"successors":[1,2]},
  {"id":"p4","block":1,"successors":[3,0]}]}
</OR:COLUMNS>
```

<OR:MASTER>

Selecting the new patterns gives arrival times 0, 3, 7, 10, 14. All windows hold, including the depot return, and distance decreases to 14.

```
{"solution":[0,1,2,3,0],"total_distance":14}
</OR:MASTER>
```

</think>

<OR:FINAL>

I submit the complete tour induced by the revised pattern selection.

```
{"solution":[0,1,2,3,0],"total_distance":14}
</OR:FINAL>
```

CG-CoT Example: Graph Coloring (GCP)

Instruction. Partition all vertices into independent color classes, minimizing the number of colors.

Input. Vertices: $\{A, B, C, D\}$. Edges: $\{AB, BC, CD, DA\}$. Every vertex must receive exactly one color.

CG-CoT Response.

<think>

<OR:COLUMNS>

I initialize singleton independent sets. Each set is a legal candidate color class.

```
{"columns":[
  {"id":"p1","pattern":["A"]},
  {"id":"p2","pattern":["B"]},
  {"id":"p3","pattern":["C"]},
  {"id":"p4","pattern":["D"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I assign one singleton class to each vertex, obtaining a feasible four-color partition.

```
{"solution":[["A"],["B"],["C"],["D"]],"total_colors":4}
</OR:MASTER>
```

<OR:COLUMNS>

A and C are nonadjacent, as are B and D. I add these two independent sets as alternative color classes.

```
{"columns":[
  {"id":"p5","pattern":["A","C"]},
  {"id":"p6","pattern":["B","D"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I replace the singleton classes with the two new sets. Every vertex appears once and all adjacent vertices have different colors.

```
{"solution":[["A","C"],["B","D"]],"total_colors":2}
</OR:MASTER>
```

</think>

<OR:FINAL>

I submit the two-color partition. Since the graph contains an edge, at least two colors are necessary.

```
{"solution":[["A","C"],["B","D"]],"total_colors":2}
</OR:FINAL>
```

CG-CoT Example: Job-Shop Scheduling (JSSP)

Instruction. Schedule every operation without violating job precedences or machine non-overlap, minimizing makespan.

Input. Jobs: $A : (M_1, 2) \to (M_2, 1)$ and $B : (M_2, 2) \to (M_1, 1)$, where pairs denote machine and duration. Machine identifiers are $1, 2$. Each job has task indices $0, 1$.

CG-CoT Response.

<think>

<OR:COLUMNS>

I initialize one order per machine, placing A before B. The combined precedence graph is acyclic.

```
{"columns":[
  {"id":"p1","machine":1,"job_order":["A","B"]},
  {"id":"p2","machine":2,"job_order":["A","B"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

The selected orders give start times (0, 2) for A and (3, 5) for B. The resulting makespan is six.

```
{"machine_columns":[
  {"machine":1,"column":"p1"},
  {"machine":2,"column":"p2"}],
 "makespan":6}
</OR:MASTER>
```

<OR:COLUMNS>

I add the alternative order B before A on machine 2. The order on machine 1 remains available.

```
{"columns":[{"id":"p3","machine":2,"job_order":["B","A"]}]}
</OR:COLUMNS>
```

<OR:MASTER>

I select p1 and p3. Both jobs start at time zero on different machines, then start their second operations at time two. Makespan decreases to three.

```
{"machine_columns":[
  {"machine":1,"column":"p1"},
  {"machine":2,"column":"p3"}],
 "makespan":3}
</OR:MASTER>
```

</think>

<OR:FINAL>

I submit the implied earliest-start schedule. Each machine has total processing time three, which certifies the achieved makespan.

```
{"solution":[
  {"job":"A","tasks":[
    {"task":0,"start":0,"machine":1,"duration":2},
    {"task":1,"start":2,"machine":2,"duration":1}]},
  {"job":"B","tasks":[
    {"task":0,"start":0,"machine":2,"duration":2},
    {"task":1,"start":2,"machine":1,"duration":1}]}],
 "makespan":3}
</OR:FINAL>
```

C Mathematical Foundations of Process Evaluation

C.1 Pricing bounds and approximate restricted masters

All statements in this subsection concern the same fixed master formulation and its full configuration domains. Consider the dual in Eq. (A.3), with group masses $m_k \geq 0$. For any candidate multipliers with the required signs, suppose a certified lower bound $\underline{\rho}_k$ satisfies

$$\underline{\rho}_k \leq \rho_k^\star = \min_{p \in \mathcal{P}_k} \big(c_{kp} - a_{kp}^\top y - e_{kp}^\top \nu - \sigma_k\big).$$

Set $\delta_k = \max(0, -\underline{\rho}_k)$ and $\sigma'_k = \sigma_k - \delta_k$. Every repaired reduced cost satisfies

$$c_{kp} - a_{kp}^\top y - e_{kp}^\top \nu - \sigma'_k = r_{kp} + \delta_k \geq \underline{\rho}_k + \delta_k \geq 0.$$

The repaired multipliers are therefore feasible for the full dual, with objective

$$L_{\mathrm{DW}} = d(y, \nu, \sigma) - \sum_k m_k \delta_k \leq z_L^\star \leq z_I^\star. \tag{C.1}$$

This is an application of dual feasibility and the column-generation bounds discussed by Gondzio et al. (2013).

Proof of Proposition 1. The repair above proves the first inequality. If u_C and d_C are objectives of a primal–dual feasible pair for the same RMP, weak duality gives $\varepsilon_R = u_C - d_C \geq 0$. Extending the restricted primal by zero coefficients gives a full-master feasible solution, so $z_L^\star \leq u_C$. Consequently,

$$0 \leq u_C - z_L^\star \leq u_C - L_{\mathrm{DW}} = \varepsilon_R + \sum_k m_k \delta_k,$$

which proves the second inequality. The certificate in Eq. (C.1) does not itself require restricted primal or dual optimality. Restricted dual feasibility is needed to interpret ε_R as a nonnegative RMP gap.

For a feasible integer plan with objective U, any feasible RMP pair additionally gives the identity

$$U - L_{\mathrm{DW}} = (U - u_C) + (u_C - d_C) + \sum_k m_k \delta_k. \tag{C.2}$$

All three terms are nonnegative when $u_C \leq U$. The first term is not generally an integer optimization error. If the restricted LP is solved optimally, $u_C = v_L(\mathcal{C})$, it can be further decomposed as

$$U - v_L(\mathcal{C}) = \big[U - v_I(\mathcal{C})\big] + \big[v_I(\mathcal{C}) - v_L(\mathcal{C})\big],$$

where $v_I(\mathcal{C})$ is the restricted integer optimum and the plan is feasible within $\mathcal{C}$. Thus the certificate gap can include assembly error, an integrality gap, restricted-solve error, and pricing-bound slack. In particular, $\delta_k = 0$ certifies full-dual feasibility, but does not certify LP optimality unless the primal–dual gap also vanishes.

The group constraints are essential to this additive repair. For an ungrouped nonnegative master, an alternative is a certified bound M on the total coefficient mass of at least one optimal LP solution. If all full-domain reduced costs are at least $-\delta$, then $d - M\delta$ is a lower bound, by evaluating the Lagrangian at that solution. The number of generated columns is not such a bound. Formulations without appropriate group constraints or a certified mass bound require a different repair, such as the proportional construction below.

C.2 A nonnegative primal–dual gap decomposition

Write a full master compactly as

$$\min_{\lambda \geq 0} c^\top \lambda \quad \text{subject to} \quad A\lambda \geq b, \qquad H\lambda = h.$$

Let $(\widetilde{y}, \widetilde{\nu})$ be any full-dual feasible pair, with $\widetilde{y} \geq 0$ and $\widetilde{r} = c - A^\top \widetilde{y} - H^\top \widetilde{\nu} \geq 0$. For a primal-feasible plan λ with objective $U = c^\top \lambda$, its gap to the dual value $L = b^\top \widetilde{y} + h^\top \widetilde{\nu}$ is

$$\begin{aligned} U - L &= \widetilde{r}^\top \lambda + \widetilde{y}^\top (A\lambda - b) + \widetilde{\nu}^\top (H\lambda - h) \\ &= \widetilde{r}^\top \lambda + \widetilde{y}^\top (A\lambda - b) \geq 0. \end{aligned} \tag{C.3}$$

The equality constraints cancel, and both remaining terms are nonnegative. This identity requires feasibility, not optimality or exact solution of either optimization problem. It explains how reduced costs and constraint surplus contribute to a certified objective gap.

For an integer plan, $U-L$ also contains the LP integrality gap and any looseness of the dual bound. It is not the exact regret $U-z_I^\star$. A dual feasible only for an RMP supports the corresponding restricted identity, not a full-domain certificate. If objective values are integral, $\lceil L \rceil$ remains an integer lower bound, but need not be an LP lower bound or a dual objective. Equation (C.3) uses the unrounded value.

C.3 Pool quality, assembly quality, and local comparisons

For a fixed original problem and a feasible integer plan contained in the current pool,

$$U - z_I^\star = \underbrace{v_I(\mathcal{C}) - z_I^\star}_{\text{restricted-pool gap}} + \underbrace{U - v_I(\mathcal{C})}_{\text{assembly gap}}. \tag{C.4}$$

Both terms are nonnegative. This separation motivates reusable column construction and complete-plan revision. For two feasible plans over the same pool, the difference in their objective values equals the difference in their assembly gaps, because the unknown $v_I(\mathcal{C})$ cancels.

A similar cancellation applies to pricing comparisons. Fix a visible prefix, one pricing subproblem, its dual anchor, and the positive scale s_C. For a nonempty batch of legal columns B with mean reduced cost $\bar{r}_h(B)$,

$$\ell_C(B \mid h) = \frac{\bar{r}_h(B) - \rho_h^\star}{s_C} + \frac{\rho_h^\star - \underline{\rho}_h}{s_C}. \tag{C.5}$$

The first term measures the batch's mean pricing suboptimality. The second is relaxation slack, common to all candidates under this anchor. Both are nonnegative, and

$$\ell_C(B^- \mid h) - \ell_C(B^+ \mid h) = \frac{\bar{r}_h(B^-) - \bar{r}_h(B^+)}{s_C}.$$

A tighter common pricing bound changes absolute process scores but does not change this difference or preferences weighted by the raw reduced-cost gap.

For batches spanning several subproblems, the cancellation requires fixed group composition. If a batch aggregates group losses with weights $w_k(B)$, the difference between two candidates includes the additional term

$$\sum_k \big(w_k(B^-) - w_k(B^+)\big) \frac{\rho_k^\star - \underline{\rho}_k}{s_k}.$$

The framework therefore compares one subproblem at a time, or uses a prescribed common aggregation. Across different prefixes, relaxation slack can differ and is not a policy-independent constant. Pricing quality is also distinct from the restricted integer gap. A negative reduced cost need not yield an immediate LP improvement under degeneracy, and neither implies integer improvement.

C.4 Proportional certificates for positive-cost masters

For the full covering LP

$$\min_{\lambda \geq 0} c^\top \lambda \quad \text{subject to} \quad A\lambda \geq b,$$

suppose $c_j > 0$ for every column. Given $y \geq 0$, let $\kappa \geq 1$ be certified to satisfy $a_j^\top y \leq \kappa c_j$ for every legal column. Then y/κ is full-dual feasible, so

$$L = \frac{b^\top y}{\kappa} \leq z_L^\star \leq z_I^\star \leq U \tag{C.6}$$

for every feasible integer objective U. A feasible pricing candidate cannot supply the required upper bound on $\max_j a_j^\top y / c_j$.

If u and $d = b^\top y$ are objectives of a feasible restricted primal–dual pair, let $\varepsilon_R = u - d \geq 0$. For $\kappa = 1 + \varepsilon_P$ with $\varepsilon_P \geq 0$,

$$L = \frac{u - \varepsilon_R}{1 + \varepsilon_P}, \qquad 0 \leq u - z_L^\star \leq \frac{u\varepsilon_P + \varepsilon_R}{1 + \varepsilon_P}. \tag{C.7}$$

The upper bound follows by subtracting L from u. Thus approximate restricted solutions can be used without equating the candidate dual objective with the restricted optimum. An integer RMP objective is a feasible upper bound and cannot replace d in the certificate numerator. With equality or additional master rows, all associated multipliers and right-hand-side contributions must be included and all dual constraints verified.

For unit column costs, a certified $M \geq \max_j a_j^\top y$ permits $\kappa = \max(1, M)$. If y is an exact optimal RMP dual and $M \leq 1 + \epsilon$ for $\epsilon \geq 0$, then

$$\frac{v_L(\mathcal{C})}{1+\epsilon} \leq z_L^\star \leq v_L(\mathcal{C}).$$

The case $\epsilon = 0$ certifies full LP optimality. $\epsilon > 0$ gives an approximate certificate. The associated multiplicative ratio additionally requires a positive full LP optimum. The general DW repair in Proposition 1 does not require positive column costs.

D Reward Specification and Learning Procedure

D.1 Block scores and complete-response rewards

The evaluator replays each response in generation order. Valid pricing-stage `COLUMNS` and feasible `MASTER` blocks use Eq. (6). Before pricing, columns use the task's nonnegative feasibility-restoration score. A structurally valid but infeasible `MASTER` receives $\ell_M = 1 + v$, with violation $v \in [0, 1]$. Let $\mathcal{K}(\tau)$ contain blocks with available mathematical scores and $N(\tau)$ count attempted optimization blocks. Then

$$D(\tau) = \sum_{t \in \mathcal{K}(\tau)} \alpha_t \ell_t + c_{\text{step}} N(\tau) + E_{\text{aux}}(\tau), \qquad P(\tau) = \begin{cases} s_P/(s_P + D(\tau)), & \mathcal{K}(\tau) \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases} \tag{D.1}$$

Here α_t is the block-type weight, and E_{aux} accounts for invalid blocks, no-ops, missing evaluations, and response-format errors. Invalidity takes precedence over no-op handling. A column batch with no novel configuration is invalid. An unavailable evaluation incurs the fixed missing-data cost, not a certified loss or a numerical preference label. Errors preserve earlier valid scores and permit subsequent recovery. Final feasibility is checked independently from the submitted solution.

For a positive-cost task with validated feasible reference objective $U_{\text{ref}} > 0$, the terminal reward is

$$R_F(\tau) = \begin{cases} b_F + w_F \left[1 + (U(\tau)/U_{\text{ref}})^3\right]^{-1}, & \text{if the submitted solution is feasible,} \\ -w_v v(\tau), & \text{otherwise,} \end{cases} \tag{D.2}$$

where $v(\tau) \in [0, 1]$, and $R = R_F + \lambda_P P$. The reference sets the scale and need not be optimal. Zero or negative objectives require a different normalization. With nonnegative costs and $s_P > 0$, $P \in [0, 1]$.

D.2 Preference construction and weighting

For each instance and block type, RDO selects up to a fixed number of current-policy prefixes uniformly before inspecting scores. Candidates share the instance, visible prefix, operation type, and evaluator state. The fixed prefix ends immediately after the operation header. The sampled suffix includes the candidate's reasoning and ends at its closing delimiter, EOS, or generation limit. Only boundaries matching the actual sampled token prefix are used. At most one original block is reused. Alternatives use the same frozen sampling policy. Duplicate, invalid, and tied draws still consume the fixed budget. No extra draws are made to obtain usable pairs.

A valid `COLUMNS` block is preferred to an invalid one, and a structurally valid feasible `MASTER` to a candidate failing either check. Pairs of two invalid column blocks or two infeasible master proposals are omitted. Numerical comparisons prefer lower Phase-I loss, mean reduced cost, or feasible integer objective, with common stages, dual anchors, scales, and the grouping convention of Appendix C. Numerical ties and unavailable evaluations are omitted. For raw quality difference $\Delta_q > 0$ within comparison subtype q,

$$w_q = \min\left\{ w_q^{\max}, \left(\frac{\Delta_q}{s_q^{\text{gap}}} \right)^\rho \right\}, \tag{D.3}$$

with positive scale, cap, and exponent. Categorical preferences use a fixed weight. Numerical weights use raw differences, not differences after bounded score transforms. Crossing zero reduced cost adds no bonus.

Equation (4) uses the frozen SFT policy as p_{ref} (Rafailov et al., 2023). For each block type, count-based averages weight instances, then subtypes, prefixes, and pairs equally, without normalizing away quality weights. Both block types are collected before either update. Actor likelihoods are recomputed at each update while reference likelihoods remain fixed.

D.3 Complete-response RLOO and the causal history baseline

After RDO, sample $K \geq 2$ fresh independent responses per instance from the updated actor. The leave-one-out baseline is $\overline{R}_{-i} = (K-1)^{-1} \sum_{j \neq i} R_j$ (Ahmadian et al., 2024). Equation (1) gives

$$\nabla_\theta \log p_\theta(B_t \mid h_t) = \sum_{j \in B_t} \nabla_\theta \log \pi_\theta(y_j \mid I, y_{<j}). \tag{D.4}$$

We apply Eq. (5) to a partition containing every generated token exactly once, including headers, intervening prose, `FINAL`, EOS, and malformed or truncated tails. Its unbiased-gradient interpretation requires on-policy sampling, likelihoods matching the generation temperature, support, and stopping rule, and inclusion independent of response success.

For loss $D_{\mathrm{past}}(h_t)$ accumulated before the current segment, the optional history adjustment is

$$q(h_t) = \frac{D_{\mathrm{past}}(h_t)}{s_P + D_{\mathrm{past}}(h_t)}, \qquad \widehat{A}_{i,t} = R_i - \overline{R}_{-i} + \eta q(h_{i,t}). \tag{D.5}$$

A closing or recovery token precedes its score update, so that score affects only later tokens. If replay cannot continue, the last valid history value is retained for the remaining tokens. All rewards and baselines are detached during differentiation. For a history-dependent, action-independent baseline $b(h_t)$, normalization gives, under the usual differentiation–expectation interchange,

$$\mathbb{E}_{B_t \mid h_t}\left[b(h_t) \nabla_\theta \log p_\theta(B_t \mid h_t)\right] = b(h_t) \nabla_\theta \sum_{B_t} p_\theta(B_t \mid h_t) = 0. \tag{D.6}$$

Thus the history adjustment, denoted RTG in the ablations, preserves the expected complete-reward gradient. $\eta = 0$ recovers ordinary RLOO. This adjustment changes the gradient estimator while preserving the objective. Because P is nonlinear in accumulated loss, raw block losses are not additive immediate rewards.

D.4 Training cycle

Initialize by negative log-likelihood training on complete CG-CoT demonstrations and freeze the SFT reference. Each subsequent iteration:

1. Samples current-policy responses and collects same-prefix alternatives for both block types.
2. Privately evaluates candidates, forms weighted pairs, and updates `COLUMNS` preferences followed by `MASTER` preferences.
3. Samples fresh independent complete responses from the updated actor and computes final rewards, process scores, and history traces.
4. Applies one RLOO update using Eq. (5), then recollects local candidates for the next iteration.

Evaluator feedback is never appended during generation. Inference remains one autonomous response.

E Task Instantiation for Covering Cutting Stock

E.1 Task interface and explicit solutions

A task instantiation specifies legal columns, the integer master, decoding and feasibility checks for complete plans, a feasibility-restoration criterion, and certified evaluation bounds. These objects

connect the general construction in Sections 3–4.4 to a concrete optimization problem. We describe the covering cutting-stock implementation as one such instantiation. Appendix G gives the compact formulations and decompositions for all six tasks.

Let W be a positive integer stock capacity, with item widths $0 < w_i \le W$ and integer demands $d_i > 0$. A column is a nonempty integer pattern

$$a \in \mathcal{A} = \{a \in \mathbb{Z}^m_{\ge 0} : 0 < w^\top a \le W\}.$$

The master minimizes the number of stock units

$$z^\star_I = \min_{\lambda \in \mathbb{Z}^{|\mathcal{A}|}_{\ge 0}} \left\{ \sum_{a \in \mathcal{A}} \lambda_a : \sum_{a \in \mathcal{A}} a_i \lambda_a \ge d_i \quad \forall i \right\}. \tag{E.1}$$

Overproduction is allowed. Here the master has no group equalities. The proportional certificate of Appendix C applies directly.

A `COLUMNS` payload supplies at most 32 new patterns with fresh identifiers. A `MASTER` or `FINAL` payload lists one pattern dictionary per physical stock unit, repeating dictionaries when needed. The objective is the actual list length, independently of the reported total. The prompt requests 1–16 expansion/reassembly rounds within the output budget and a final copy of the last master plan. The demonstration compiler uses the chronological branch-and-price construction in Section 4.2, enforces a configurable total-token budget, and defaults to a two-second restricted integer solve per cumulative pool. These offline solves are separate from the online process evaluator.

E.2 Feasibility restoration and the private LP anchor

For a pool $\mathcal{C}$, consider

$$\Phi(\mathcal{C}) = \min_{x,s \ge 0} \left\{ \frac{\mathbf{1}^\top s}{\mathbf{1}^\top d} : A_{\mathcal{C}} x + s \ge d \right\}.$$

Because column multiplicities are unbounded and columns are nonnegative, a covered item type can receive arbitrary supply without harming other rows. Hence

$$\Phi(\mathcal{C}) = \frac{\sum_{i:\, a_i = 0 \ \forall a \in \mathcal{C}} d_i}{\sum_i d_i}. \tag{E.2}$$

When $\Phi(\mathcal{C}) > 0$, a candidate batch B receives the Phase-I score $\Phi(\mathcal{C} \cup B)$. The pre-action pool determines the stage, including when a candidate first completes coverage.

At the pricing stage, the evaluator solves the continuous covering RMP over the complete available pool recorded at the most recent structurally valid `MASTER`. This is a snapshot of all available columns, not only the selected plan's support. Plan multiplicities are not fixed in the LP. The resulting nonnegative dual y is shared by all alternatives at that prefix, with reduced cost $r(a) = 1 - y^\top a$.

E.3 Certified pricing relaxations

Let j maximize y_i/w_i, let $n = \lfloor W/w_j \rfloor$, and, when there is another item type, let $\rho_2 = \max_{i \ne j} y_i/w_i$. The evaluator computes three upper bounds on maximum pattern profit

$$\begin{aligned} M_0 &= W \frac{y_j}{w_j}, \\ M_1 &= n y_j + (W - n w_j)\rho_2, \\ M_2 &= \left\lfloor \frac{W}{\min_i w_i} \right\rfloor \max_i y_i, \qquad M = \min\{M_0, M_1, M_2\}. \end{aligned} \tag{E.3}$$

For a single item type, $M_1 = n y_j$. The density bound M_0 follows from $y^\top a \le (w^\top a) \max_i y_i/w_i$. For M_1, every legal pattern satisfies

$$y^\top a \le a_j y_j + (W - a_j w_j)\rho_2 \le n y_j + (W - n w_j)\rho_2,$$

since $a_j \le n$ and $y_j - w_j\rho_2 \ge 0$. The bound M_2 follows from the maximum number of items in one stock unit. Thus all three bounds are valid, and

$$\underline{\rho} = 1 - M \le \min_{a \in \mathcal{A}} r(a), \qquad L_{\text{scale}} = \frac{d^\top y}{\max(1, M)} \le z_L^\star \le z_I^\star. \tag{E.4}$$

The column score uses the mean reduced cost of all unique novel patterns and the certified lower bound $\underline{\rho}$. The scaled bound is recorded as an additional certificate.

E.4 The master bound and numerical certification

The actual `MASTER` score uses

$$L_M = \left\lceil \max\left\{ \frac{\sum_i w_i d_i}{W}, \sum_{i:\, 2w_i > W} d_i \right\} \right\rceil. \tag{E.5}$$

The first term follows from total item volume. The second counts items that cannot share a stock unit with another such item. Equivalently, $y_i = w_i/W$ and $y_i = \mathbf{1}\{2w_i > W\}$ are both full-dual feasible. Their maximum is a valid LP lower bound, and rounding up gives an integer lower bound. The score in Eq. (6) uses this L_M, not L_{scale}.

For the density bound alone, the scaled certificate cannot exceed the volume bound. Indeed, with $q = \max_i y_i/w_i$ and $V = \sum_i w_i d_i/W$, $d^\top y \le qWV$, so $d^\top y/\max(1, qW) \le V$. This limitation concerns M_0 alone. It does not establish the same ordering for the minimum of all three bounds.

The LP routine accepts a numerical optimum only after primal feasibility, dual feasibility, and primal–dual consistency checks. It then recovers nonnegative rational coefficients with denominator at most 10^6. Any primal covering deficit is repaired by increasing an already present covering column. The dual is divided by $\max\{1, \max_{a \in \mathcal{C}} a^\top y\}$. Exact rational checks subsequently verify a restricted primal–dual interval $[d_C, u_C]$.

E.5 Native terminal assessment

The native evaluator checks the submitted list directly. It verifies item identifiers, integer nonnegative counts, per-stock capacity, and aggregate demand coverage, and counts the stock units used. When counts are parseable, the violation score is the mean of normalized unmet demand and normalized capacity excess

$$v = \frac{1}{2}\left[\frac{\sum_i \max(0, d_i - \sum_{a \in M} a_i)}{\sum_i d_i} + \frac{e_{\text{cap}}}{\sum_i w_i d_i + e_{\text{cap}}} \right], \quad e_{\text{cap}} = \sum_{a \in M} \max(0, w^\top a - W).$$

Malformed or invalid-count submissions receive $v = 1$. Feasible submissions receive $v = 0$. An independently validated feasible training reference supplies U_{ref} in Eq. (D.2).

F Optimization Behavior Evaluation

F.1 Correctness and consistency checks

A COLUMNS capacity-validity check requires a parseable payload of nonempty patterns with valid identifiers, nonnegative integer counts, and legal resource use. This local check differs from replay validity, which also requires novel configurations and fresh identifiers. MASTER feasibility additionally requires legal declared columns, integer multiplicities, and satisfaction of the original task's linking constraints. FINAL consistency compares the actual final plan with the last attempted MASTER, not a best-so-far or last-feasible substitute. Canonicalized patterns and multiplicities determine plan equality. Equal objective values alone do not. A missing or unparseable final or last MASTER fails the consistency check. These checks distinguish local legality, global feasibility, and faithful final submission.

For method m, instance i, and block type b, let N_{mib} count attempted blocks and V_{mib} count passing blocks. An instance-averaged diagnostic is

$$s_{mib} = \begin{cases} V_{mib}/N_{mib}, & N_{mib} > 0, \\ 0, & N_{mib} = 0, \end{cases} \qquad R_{mb} = \frac{100}{n} \sum_{i=1}^{n} s_{mib}.$$

Malformed attempts remain in the denominator, and a response with no attempted block receives zero. FINAL consistency is a binary per-instance check on the same population.

F.2 Single-response trajectory illustrations

Figure 2 uses four complete, unmodified responses from post-training logs: GCP, BPP, job-shop scheduling (JSP), and TSPTW, with 10, 9, 10, and 5 rounds, respectively. At each round, all accepted declared columns form the cumulative pool $\mathcal{C}_t$. Rejected batches do not add any columns. The solid objective is $U_t = f(x_t)$ for the generated feasible plan x_t. The dashed reference is $v_I(\mathcal{C}_t) = \min_{x \in \mathcal{X}_I(\mathcal{C}_t)} f(x)$, the restricted integer optimum under the original task constraints, as defined in Equation (3). All displayed MASTER plans are valid. GCP decreases from seven to four colors and TSPTW from 579.662679 to 523.584944, matching their restricted integer optima at each round. The BPP pool supports seven bins from round 2, while the generated plan first uses seven in round 8. Its sixth COLUMNS batch is rejected for a repeated pattern, leaving the pool unchanged. Consequently, no blue candidate arrows are drawn for that batch at round 5. For JSP, the pool supports makespan 263 from round 2 and 257 from round 7, while the generated plan reaches 263 in round 7 and 259 in the final round. Restricted integer optima are computed using SCIP for GCP and BPP, complete enumeration of declared machine-order combinations for JSP, and complete restricted block-pattern enumeration for TSPTW.

Each arrow starts at the current MASTER and encodes a next-batch reduced cost using a fixed scale within its panel: 0.55 for GCP, 0.42 for BPP, 0.16 for JSP, and 0.34 for TSPTW. Under the frozen current-prefix LP dual, each accepted new column has reduced cost

$$r_{t,kp} = c_{kp} - a_{kp}^{\top} y_t - e_{kp}^{\top} \nu_t - \sigma_{t,k}, \qquad \rho_t^{\star} = \min_{k,\, p \in \mathcal{P}_k} r_{t,kp},$$

where absent dual terms are omitted and the pricing reference $\rho_t^{\star}$ minimizes over the full legal column domain. For panel scale γ, each blue candidate arrow ends at $U_t + \gamma r_{t,kp}$, and the orange pricing-reference arrow ends at $U_t + \gamma \rho_t^{\star}$. Blue denotes generated quantities, orange references, and darker shades later rounds. Downward arrows are negative, upward arrows positive, and diamonds zero. There are no arrows after the final MASTER. Candidates and pricing references at a prefix use the same frozen LP dual and legal column domain. The pricing references exhaustively minimize reduced cost over that domain using rational arithmetic. TSPTW uses the native floating-point distance and LP-dual coefficients, with the original LP solver's numerical tolerances.

F.3 Process rewards and sequence advantages

Figure 3 analyzes 60 completed TSPTW training cycles along the recorded continuation chain, with eight instances per cycle and four fresh RLOO responses per instance. The resulting 480 groups contain 1,920 responses. RDO seed responses are excluded, and resumed cycles are counted once.

Terminal and Full denote reward components evaluated on these same responses. To reproduce the logged sequence advantages, we retain this run's overall reward scale of two and process coefficient $\lambda_P = 0.1$:

$$R_i^F = 2R_F(\tau_i), \qquad R_i^P = 0.2P(\tau_i), \qquad \widetilde{R}_i = R_i^F + R_i^P. \tag{F.1}$$

This run uses the calibrated scale $s_P \simeq 2.24370723$ to compute $P(\tau_i) = s_P/(s_P + D_i)$ for eligible responses. For $K = 4$, define the leave-one-out advantages of each contribution and their sum by

$$A_i^x = R_i^x - \frac{1}{K-1}\sum_{j\neq i} R_j^x \quad (x \in \{F, P\}), \qquad A_i = \widetilde{R}_i - \frac{1}{K-1}\sum_{j\neq i}\widetilde{R}_j = A_i^F + A_i^P. \tag{F.2}$$

Equal terminal contributions imply $A_i^F = 0$, leaving $A_i = A_i^P$. This is a reward-centering identity. The RTG history adjustment $\eta q(h_{i,t})$ in Equation (5) remains separate from the plotted sequence advantages.

Panel (a) plots one observation per group, using $S_g = (K^{-1}\sum_i A_i^2)^{1/2}$ and the corresponding terminal-only value. Among 35 groups with equal terminal rewards, 27 have $S_g > 0$ with full rewards and eight remain at zero. The violin includes all 35 observations, with a median marker and interquartile interval. Panel (b) uses six consecutive ten-cycle windows, each containing 80 groups, with 4, 3, 4, 4, 5, and 7 groups distinguished only by process feedback. Of those 27 groups, 21 have four infeasible final solutions and six have four feasible final solutions. These frequencies measure additional reward discrimination across the sampled groups.

Panels (c,d) use the original four-response group for instance `tsptw_train_002159` at cycle 32. The illustrative B/C pair was selected after inspection by maximizing the difference in column validity and then minimizing relative token-count differences among pairs with matching action counts and protocol penalties, without truncation or missing evaluator scores. B and C each contain six `COLUMNS` and six `MASTER` operations. Both have a step cost of 1.44. Their combined column and weighted master quality losses are 1.98973771 and 1.99604220, respectively. C additionally incurs 18 units of invalid-column penalties, yielding total losses $D_B = 3.42973771$ and $D_C = 21.43604220$. The zero column-quality loss recorded for C reflects invalid blocks being charged through the error term. The distinction is therefore primarily due to local validity. All four terminal contributions equal -0.4. Keeping the original four-response baseline gives advantages $(0.04527579,\ 0.03871252,\ -0.04148031,\ -0.04250800)$ for A–D.

F.4 Optimization error and certificate slack

The reward-signal analysis above concerns differences between sampled responses. The OR quality terms can additionally be decomposed into optimization error and certificate slack. For a valid pricing-stage batch at a fixed dual anchor, let $\bar{r}$ be its mean reduced cost, $\rho^\star$ the exact pricing optimum on the full legal domain, and $\underline{\rho}$ the bound used by the training evaluator. Then

$$\ell_C = \underbrace{\frac{\bar{r} - \rho^\star}{s_C}}_{\text{true pricing error}} + \underbrace{\frac{\rho^\star - \underline{\rho}}{s_C}}_{\text{bound slack}}. \tag{F.3}$$

Both terms are nonnegative, but a smaller total need not imply a smaller true pricing error when anchors change. At the same prefix with a common pricing group, dual, bound, and scale,

$$\ell_C(B^- \mid h) - \ell_C(B^+ \mid h) = \frac{\bar{r}_h(B^-) - \bar{r}_h(B^+)}{s_C}. \tag{F.4}$$

The common slack cancels in RDO comparisons. Multi-group comparisons require fixed aggregation weights (Appendix C).

For a feasible generated MASTER with objective U and a certified positive global integer optimum $z_I^\star$,

$$U - L_M = \underbrace{U - z_I^\star}_{\text{true integer error}} + \underbrace{z_I^\star - L_M}_{\text{bound slack}}. \tag{F.5}$$

Dividing both terms by $z_I^\star$ gives an additive normalized certificate gap, not an additive decomposition of the bounded loss itself. At fixed L_M and $s_M > 0$, $\ell_M = (U - L_M)/(s_M + U - L_M)$ strictly decreases with decreasing U but can remain positive at integer optimality when $L_M < z_I^\star$. For a fixed instance set with fixed weights and bounds, the average slack is constant. Changing feasible subsets can change it without tightening any bound.

G Compact Formulations and Task-Specific Decompositions

This appendix specifies the compact formulations used by SCIP and the column representations used by the task-specific branch-and-price solvers. Both formulations retain the native task objective and feasibility conditions. Their LP relaxations need not coincide. The decomposition defines the optimization objects underlying CG-CoT, while the complete branch-and-price search supplies offline solution trajectories. CG-CoT reassembles its cumulative pool under the original constraints, as described in Appendix B.

G.1 Common construction

Let $\mathcal{P}_k$ denote the feasible configurations of block k. A configuration p has cost c_{kp} and linking coefficients a_{kp}. The master selects configurations through nonnegative variables λ_{kp}, with $\sum_{p\in\mathcal{P}_k}\lambda_{kp}=1$ for each block. Empty configurations represent unused resources when applicable. Shared constraints enforce coverage, connectivity, or temporal consistency across blocks. Restricting $\mathcal{P}_k$ to a finite pool gives the RMP.

For linking prices π and convexity price σ_k, pricing minimizes $c_{kp}-\pi^\top a_{kp}-\sigma_k$ over the current block domain. Additional master rows contribute their corresponding dual terms. Negative reduced-cost configurations expand the pool, after which the RMP and its dual prices are recomputed. Branch decisions restrict the original variables or the admissible configurations. If the inherited pool is infeasible, artificial variables and feasibility pricing restore the RMP before objective-based pricing resumes. Global lower bounds require valid full-domain pricing bounds, as established in Appendix C. An incomplete pricing search cannot establish the absence of improving columns.

G.2 Cutting stock

Compact formulation. Item type i has width w_i and demand d_i, and stock capacity is W. A feasible packing provides an upper bound K on the number of stock units needed. Let $y_k\in\{0,1\}$ indicate stock use and $x_{ik}\in\mathbb{Z}_{\geq 0}$ denote the number of items of type i cut from stock k. The compact model is

$$\begin{aligned}\min\quad & \sum_{k=1}^{K} y_k \\ \text{s.t.}\quad & \sum_k x_{ik}\geq d_i && \forall i,\\ & \sum_i w_i x_{ik}\leq W y_k && \forall k,\\ & x_{ik}\leq \lfloor W/w_i\rfloor y_k && \forall i,k.\end{aligned} \tag{G.1}$$

The implementation also imposes $y_k\leq\sum_i x_{ik}$ and $y_k\geq y_{k+1}$ to eliminate empty active units and break resource symmetry. Decimal widths and capacity are converted to exact integer units. Demand constraints permit overproduction.

Column generation and branching. A stock configuration is $a\in\mathbb{Z}^m_{\geq 0}$ satisfying $w^\top a\leq W$, with cost $c(a)=\mathbf{1}\{a\neq 0\}$. The stock-indexed master uses

$$\min\sum_{k,a}c(a)\lambda_{ka},\qquad \sum_{k,a}a_i\lambda_{ka}\geq d_i,\qquad \sum_a\lambda_{ka}=1. \tag{G.2}$$

Its integer form chooses one configuration per stock unit. For demand prices $\pi_i\geq 0$, pricing minimizes $c(a)-\pi^\top a-\sigma_k$ subject to capacity and node-specific bounds on item counts and stock activation. The implementation uses integer pricing with a bounded-knapsack dynamic program to verify pricing bounds. Branching splits a fractional reconstructed item count at its floor and ceiling, or fixes a stock-activation variable. Symmetry handling groups interchangeable stock domains. Removing stock indices yields the covering-pattern master in Appendix E, which is used for cumulative integer-plan construction.

G.3 BIN PACKING

Compact formulation. Each item i has weight w_i and must occupy exactly one bin of capacity W. With binary assignment x_{ik} and bin-use y_k, the model minimizes $\sum_k y_k$ subject to

$$\sum_k x_{ik} = 1, \qquad \sum_i w_i x_{ik} \le W y_k, \qquad x_{ik} \le y_k. \tag{G.3}$$

A first-fit-decreasing packing supplies the finite bin bound K. The same nonempty-bin and ordering constraints as in cutting stock remove redundant active bins and permutations.

Column generation and branching. A column is a capacity-feasible subset S of distinct items, including the empty subset for an unused bin. With $c(S) = \mathbf{1}\{S \neq \emptyset\}$, the bin-indexed master is

$$\min \sum_{k,S} c(S)\lambda_{kS}, \qquad \sum_{k,S:i\in S} \lambda_{kS} = 1, \qquad \sum_S \lambda_{kS} = 1. \tag{G.4}$$

Pricing minimizes $c(S) - \sum_{i\in S} \pi_i - \sigma_k$, a binary knapsack selection problem with the current required, forbidden, and item-pair restrictions. Customer prices are unrestricted because coverage is an equality. Ryan–Foster branching requires an item pair to occur together or separately. Assignment-based symmetry branches provide a fallback. Unlike cutting-stock patterns, bin configurations cannot repeat an item or exceed its unit demand.

G.4 CAPACITATED VEHICLE ROUTING

Compact formulation. Customers N have positive demands $q_i \le Q$, depot 0, and unrounded Euclidean distances d_{ij}. There is no fixed fleet-size constraint. Binary arc variables x_{ij} minimize $\sum_{i\neq j} d_{ij}x_{ij}$. Each customer has one incoming and one outgoing arc, and depot arrivals equal departures. A single-commodity delivery flow satisfies

$$\sum_{j\neq i} f_{ji} - \sum_{j\neq i} f_{ij} = q_i \quad (i \in N), \qquad 0 \le f_{ij} \le Q x_{ij}, \qquad f_{i0} = 0. \tag{G.5}$$

Positive customer demands exclude disconnected customer cycles, and the flow enforces vehicle capacity.

Column generation and branching. A column r is an elementary depot-to-depot route with load at most Q, incidence a_{ir}, and distance c_r. Integer plan construction uses the route-partitioning master

$$\min \sum_r c_r z_r, \qquad \sum_r a_{ir} z_r = 1 \quad (i \in N), \qquad z_r \in \{0,1\}. \tag{G.6}$$

The branch-and-price implementation instead indexes route choices by vehicle blocks and includes an empty route. A safe block bound is

$$K = \min\left\{|N|, \max\left(1, \left\lceil \frac{2\sum_i q_i}{Q} \right\rceil - 1\right)\right\}. \tag{G.7}$$

Indeed, two routes with combined load at most Q can be merged without increasing Euclidean length. After repeated merging, any two remaining loads sum to more than Q, implying $m < 2\sum_i q_i/Q$ for $m \ge 2$ routes. Thus this bound preserves at least one optimum. Vehicle symmetry also permits requiring the first $\lceil \sum_i q_i/Q \rceil$ blocks to be nonempty without fixing customer assignments.

The vehicle master has customer equalities and unit convexity constraints. Pricing minimizes $c_r - \sum_i \pi_i a_{ir} - \sigma_k$ over elementary capacity-feasible routes compatible with the branch. The implementation uses subset dynamic programming with an integer-programming fallback. Stabilized prices may guide candidate search, but acceptance uses the current RMP reduced cost. Branching uses customer pairs with an assignment-based fallback. Route distances remain unrounded for evaluating solutions, and integer-objective rounding rules from cutting stock do not apply.

G.5 TRAVELING SALESMAN WITH TIME WINDOWS

Compact formulation. Each node i has window $[a_i, b_i]$. Travel time equals Euclidean distance d_{ij}, service time is zero, and waiting is allowed. Binary arcs define one incoming and one outgoing arc at every node and minimize total distance. Departure from the depot is fixed to $t_0^{\mathrm{dep}} = \max(0, a_0)$, while return uses a distinct variable $t_0^{\mathrm{ret}} \leq b_0$. For customers, $a_i \leq t_i \leq b_i$. Every selected arc enforces

$$\widetilde{t}_j \geq t_i + d_{ij} - M_{ij}(1 - x_{ij}), \tag{G.8}$$

where $\widetilde{t}_0 = t_0^{\mathrm{ret}}$ and otherwise $\widetilde{t}_j = t_j$. Each M_{ij} is derived from the source upper and destination lower time bounds. Continuous ordering variables $1 \leq u_i \leq n$, for n customers, satisfy

$$u_j \geq u_i + 1 - n(1 - x_{ij}) \qquad (i, j \neq 0,\ i \neq j). \tag{G.9}$$

These constraints exclude subtours even when distinct locations have zero distance.

Column generation and branching. The decomposition partitions departure nodes into fixed blocks B_k of at most three nodes. A column p assigns one successor to every node in B_k, with incidence a_{ij}^{kp} and cost $c_{kp} = \sum_{i \in B_k, j} d_{ij} a_{ij}^{kp}$. Local domains exclude self-loops, repeated successors, proper local cycles, and locally time-infeasible paths. Successors need not belong to the same block. The master imposes

$$\sum_p \lambda_{kp} = 1, \qquad x_{ij} = \sum_p a_{ij}^{kp} \lambda_{kp} \quad (i \in B_k), \qquad (x, t, f) \in \mathcal{Q}. \tag{G.10}$$

Here $\mathcal{Q}$ retains global degree, timing, and single-commodity connectivity constraints, together with valid strengthening inequalities. Thus the branch-and-price shared model uses flow connectivity, whereas the compact SCIP baseline uses ordering constraints. Binary reconstructed arcs recover a complete tour. For linking rows $\sum_p a_{ij}^{kp} \lambda_{kp} - x_{ij} = 0$ with prices β_{ij}, pricing minimizes

$$c_{kp} - \sum_{i \in B_k, j} \beta_{ij} a_{ij}^{kp} - \sigma_k. \tag{G.11}$$

Enumeration has at most $O(|V|^3)$ successor assignments per block. Branches fix a fractional arc to zero or one and restrict the affected local domains. The implementation may prioritize negative-cost columns proposed by route heuristics, but full local pricing supplies the bounds. Every feasible tour projects to legal block configurations, and the shared constraints reconstruct tour feasibility, so the fixed grouping preserves the integer problem.

G.6 GRAPH COLORING

Compact formulation. For graph $G = (V, E)$, binary x_{vk} assigns vertex v to color k and y_k indicates color use. A greedy coloring bounds the available colors. The model minimizes $\sum_k y_k$ subject to

$$\sum_k x_{vk} = 1, \qquad x_{vk} \leq y_k, \qquad x_{uk} + x_{vk} \leq y_k \quad (\{u, v\} \in E). \tag{G.12}$$

Nonempty-color and color-ordering constraints remove redundant representations.

Column generation and branching. A column is an independent vertex set S. The color-indexed master has the same partitioning and convexity structure as bin packing, replacing the capacity-feasible subsets by independent sets. A nonempty set costs one and the empty set costs zero. Pricing minimizes $\mathbf{1}\{S \neq \emptyset\} - \sum_{v \in S} \pi_v - \sigma_k$, which is a maximum-weight independent-set problem for nonempty columns, subject to the node domain. Pair branching places vertices in the same or different color classes, with assignment-based symmetry branching as a fallback. Pricing completeness is required for bound-based pruning, rather than inferred from a heuristic's failure to find a new independent set.

G.7 JOB-SHOP SCHEDULING

Compact formulation. Each operation o has positive integer duration p_o and a fixed machine. Jobs prescribe operation precedence. Let $s_o \geq 0$ be its start time and C the makespan. With horizon $H = \sum_o p_o$, the model minimizes C subject to $C \geq s_o + p_o$, job-precedence inequalities, and, for operations o, v from different jobs on the same machine,

$$s_v \geq s_o + p_o - H(1 - q_{ov}), \qquad s_o \geq s_v + p_v - H q_{ov}, \qquad q_{ov} \in \{0, 1\}. \tag{G.13}$$

The implementation uses integer starts, which preserve an optimum under integral durations and the absence of release dates.

Column generation and branching. A column (a, π) is a complete job permutation on machine a. Columns on different machines remain distinct even when their permutations coincide. With $v_{ij}^{a\pi} = \mathbf{1}\{i \text{ precedes } j \text{ in } \pi\}$, the master imposes

$$\sum_{\pi} \lambda_{a\pi} = 1, \qquad q_{aij} = \sum_{\pi} v_{ij}^{a\pi} \lambda_{a\pi} \quad (i < j). \tag{G.14}$$

Job precedence, start times, and machine-disjunction constraints remain in the master. Binary q forces all positively weighted permutations on a machine to agree. Prefix and suffix processing-time inequalities further strengthen temporal coordination without requiring an idle-free schedule.

Pricing minimizes the reduced cost of a machine permutation under the linking and strengthening-row duals. Its cost decomposes into pair-order contributions, enabling subset dynamic programming. With propagated time windows, labels retain completion time and reduced cost and discard dominated states. A locally feasible permutation still requires coordination with other machines to produce a feasible complete schedule. The archived single-column solver branches on a machine pair order $q_{aij} = 0$ or 1.

G.8 RELATIONSHIP TO CG-COT AND TRAINING SUPERVISION

Columns encode cutting patterns, item subsets, routes, departure-block assignments, independent sets, or machine permutations. Their shared role is to supply reusable local decisions that an integer master combines into a complete solution. The master therefore retains task-specific global constraints, especially connectivity in TSPTW and temporal coordination in JSSP. This structure motivates the distinction between column quality and complete-plan quality in Section 4.4.

The offline branch-and-price solvers may use exact, potentially expensive pricing and integer optimization. The online training evaluator uses the relaxation bounds and mathematical checks described in Appendices C and E. The cumulative integer problems expressed by CG-CoT retain the original constraints but omit teacher-node branching restrictions. Their nested feasible sets support the monotonicity statements in Appendix B.