STRUCTURE-AWARE PRIMAL–DUAL AND LARGE-NEIGHBORHOOD SEARCH IN AN LLM-CONFIGURED OPTIMIZATION AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

We study large-scale multi-market promotion optimization on Alibaba's e-commerce platform, where high-quality feasible policies are required within a short operational window. Subsidies affect demand exponentially and coupon tiers create discontinuities, while budgets, GMV-weighted average subsidies, cross-market consistency, and merchandise-group priorities couple decisions. We develop two complementary structure-aware methods. A primal–dual (PD) method relaxes shared constraints, solves independent product subproblems, and recombines locally valid decisions through customized feasibility recovery. Adaptive large neighborhood search (LNS) with simulated-annealing acceptance instead coordinates products linked by tight resources or group priorities, enabling joint coupon-threshold crossings. An optimization agent optionally uses an LLM to select hyperparameters from predeclared finite sets, freezes the configuration, executes the portfolio, and returns the best commonly verified solution. We prove exact product-wise separation of the PD relaxation and a certifying branch-and-bound procedure for the product subproblems. On 72 held-out instances from six families with $N = 100$ to 10^5, the best verified result among PD and three predeclared LNS seeds is feasible on all 72 under a 900-second limit per run; Gurobi finds an incumbent on 33. On these common-feasible cases, the mean objective shortfall is 0.0059%. At $N \geq 10^4$, feasibility is 36/36 versus 6/36, and at $N \geq 50{,}000$ it is 18/18 versus 1/18.

1 INTRODUCTION

This problem arises from an industry collaboration with Alibaba on promotion planning for its e-commerce platform. A single campaign can involve a large product catalog across multiple markets, while the promotion policy must be generated within an operational window of roughly 15 minutes. Missing this window can delay deployment, whereas returning a merely feasible solution may lead to substantial losses in gross merchandise value (GMV). The practical goal is therefore to rapidly identify a high-quality feasible policy under a tight wall-clock budget. The resulting optimization problem is challenging for several reasons. Each subsidy affects demand, consumes budget, changes the customer-facing price, and may activate or deactivate a product-level coupon tier. At the same time, decisions are coupled through global and market-level budgets, bounds on the GMV-weighted average subsidy rate, cross-market consistency requirements, and priority relationships among merchandise groups. Together, these features give rise to a large-scale nonconvex optimization problem with smooth nonlinearities and discontinuities induced by coupon-tier thresholds, making generic optimization methods difficult to apply within the required operational time scale.

Off-the-shelf global optimization is effective at small scales, where it provides useful benchmark solutions and bounds, but its monolithic formulation becomes difficult to solve as the catalog grows to tens of thousands of products. The key exploitable structure is that each product's demand response, coupon rule, subsidy bounds, and cross-market range constraints are product-specific, whereas budgets, the overall GMV-weighted average subsidy, and merchandise-group ordering constraints couple decisions across products. Our primal–dual construction relaxes only these cross-product constraints. For each merchandise-group ordering constraint, we introduce an auxiliary subsidy-rate separator between the average subsidy rates of the higher- and lower-priority groups. With the multipliers and separators fixed, the Lagrangian decomposes into independent product-level subproblems, allowing all products to be optimized separately within each iteration. This decomposition, how-

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.

ever, does not guarantee primal feasibility: the resulting product-level decisions may collectively exceed budgets, violate the required average-subsidy range, or break merchandise-group ordering constraints. The difficulty is compounded by coupon-tier thresholds, whose crossings induce discontinuous changes in demand and subsidy cost, and by group-order constraints, whose repair may require coordinated changes across multiple products. Such coordinated adjustments can be difficult to recover through gradual multiplier updates alone within a tight computational budget. These limitations motivate our customized primal-recovery procedure and a complementary structure-aware large-neighborhood search.

We exploit these two aspects of the problem with complementary solvers. Our primal–dual (PD) method applies Lagrangian decomposition (Fisher, 1981), solves product-level subproblems in parallel, updates nonnegative multipliers via projected residual steps, and periodically recovers a feasible primal policy. In its certified mode, PD globally certifies the product-level subproblems to prescribed tolerances and treats the group-ordering substructure by finite corner enumeration, yielding a valid optimality bound under explicit conditions. At large scale, it replaces these exact routines with scalable candidate-based product optimization and feasibility repair, trading certification for computational efficiency. Our second solver is an adaptive large neighborhood search (LNS) (Shaw, 1998; Ropke & Pisinger, 2006) with simulated-annealing (SA) acceptance (Kirkpatrick et al., 1983). LNS modifies entire product-level subsidy vectors, targets neighborhoods around connected merchandise-group components and coupon boundaries, and repairs each neighborhood through a restricted multiple-choice master problem.

The two solvers have complementary search behavior: PD coordinates global resource pressure through low-dimensional dual prices and can provide a certified upper bound under explicit conditions, whereas LNS directly coordinates bounded product subsets across coupon discontinuities. We combine them in a configurable optimization agent. Before optimization, an instance card summarizes scale, active markets, enabled constraints, coupon boundaries, and group-order components; an LLM selects consequential PD and LNS hyperparameters from predefined finite sets. A deterministic controller validates and freezes the configuration, runs PD and independently seeded LNS, and returns the best evaluator-verified solution. The LLM never modifies the mathematical instance, solver code, computational budget, or returned policy. In the benchmark we run PD once and LNS with three fixed seeds; in the agent case study each solver runs once, always with a 900-second limit.

Our contributions are:

- A unified formulation for nonlinear multi-market subsidy optimization that captures real-world operational requirements arising from our collaboration with Alibaba, including resource constraints, cross-market consistency, coupon tiers, and merchandise-group ordering constraints.
- Two customized optimization methods: a PD framework with a scalable mode that exploits product-level decomposition and feasible primal recovery, together with a certified mode that solves the product subproblems globally to prescribed optimality tolerances; and a structure-aware LNS+SA method that jointly updates products in resource- and group-aware neighborhoods.
- An optimization-agent framework that uses an LLM for hyperparameter selection, freezes the configuration, runs PD and LNS independently, and returns the best verified feasible solution.
- An evaluation on 72 instances spanning six structural families and up to 100,000 products, where the portfolio finds a verified feasible solution on every instance within 900 seconds per run, whereas Gurobi finds an incumbent on only 33.

2 Problem Formulation and Structure

Products, markets, and response. Let $i \in \mathcal{I} = \{1, \dots, N\}$ denote products and let $r \in \mathcal{R}_i$ denote an active market of product i. The decision $s_{ir} \in [0, 1]$ is the direct subsidy rate. A deterministic tiered coupon rule, called CODE in our implementation, selects an amount $c_{ir}(s_{ir})$ from a finite set of tiers according to the price after direct subsidy; $c_{ir} \equiv 0$ when no coupon rule is declared. Given reference price p^0_{ir}, demand scale a_{ir}, price sensitivity b_{ir}, and exchange rate e_r, define $\widetilde{s}_{ir}(s_{ir})$ as effective subsidy rate, $q_{ir}(s_{ir})$ as demand, and $G_{ir}(s_{ir})$ as GMV expressed in a common reporting currency.

$$\widetilde{s}_{ir}(s_{ir}) = s_{ir} + c_{ir}(s_{ir})/p^0_{ir}, \quad q_{ir}(s_{ir}) = a_{ir} \exp\{-b_{ir} p^0_{ir}(1 - \widetilde{s}_{ir}(s_{ir}))\}, \tag{1}$$

$$G_{ir}(s_{ir}) = e_r p^0_{ir}(1 - \widetilde{s}_{ir}(s_{ir})) q_{ir}(s_{ir}). \tag{2}$$

Let $\mathbf{s} = (s_{ir})_{i\in\mathcal{I},\, r\in\mathcal{R}_i}$ denote the full subsidy vector. Let $\mathcal{X}_i$ contain product i's locally feasible subsidy vectors (local bounds, any effective-subsidy cap, and the range constraint), let $\mathcal{X} = \prod_i \mathcal{X}_i$, and let $\mathcal{S} \subseteq \mathcal{X}$ additionally impose all enabled aggregate constraints below. The objective is total post-discount GMV,

$$\max_{\mathbf{s}\in\mathcal{S}} \quad F(\mathbf{s}) = \sum_{i\in\mathcal{I}} \sum_{r\in\mathcal{R}_i} G_{ir}(s_{ir}). \tag{3}$$

Here, the coupon thresholds partition the subsidy interval into finitely many *fixed-coupon regimes*. Within a regime the coupon amount is constant and the response functions are smooth; crossing a threshold can change the coupon amount abruptly, inducing jumps in effective subsidy, demand, GMV, and expenditure.

Optional business constraints. For each active product–market pair (i, r), the subsidy rate satisfies local bounds $\underline{s}_{ir} \le s_{ir} \le \overline{s}_{ir}$ and, when declared, an effective-subsidy cap $\widetilde{s}_{ir}(s_{ir}) \le \overline{\widetilde{s}}_{ir}$. The subsidy expenditure is $D_{ir}(s_{ir}) = e_r p^0_{ir} \widetilde{s}_{ir}(s_{ir}) q_{ir}(s_{ir})$. Write $D_0(\mathbf{s}) = \sum_{i\in\mathcal{I}} \sum_{r\in\mathcal{R}_i} D_{ir}(s_{ir})$, $D_r(\mathbf{s}) = \sum_{i:r\in\mathcal{R}_i} D_{ir}(s_{ir})$, $H_0(\mathbf{s}) = \sum_{i\in\mathcal{I}} \sum_{r\in\mathcal{R}_i} s_{ir} G_{ir}(s_{ir})$, and $Q_0(\mathbf{s}) = \sum_{i\in\mathcal{I}} \sum_{r\in\mathcal{R}_i} G_{ir}(s_{ir})$. Let $\mathcal{R}_B$ denote the set of markets for which a market-level budget constraint is declared, and let $\mathcal{I}_\Delta$ denote the set of products for which a cross-market subsidy-range constraint is declared. An instance may include any subset of the following constraints:

$$D_0(\mathbf{s}) \le B_0, \qquad D_r(\mathbf{s}) \le B_r \qquad (r \in \mathcal{R}_B), \tag{4}$$

$$\underline{\rho} \le \frac{H_0(\mathbf{s})}{Q_0(\mathbf{s})} \le \overline{\rho}, \tag{5}$$

$$\max_{r\in\mathcal{R}_i} s_{ir} - \min_{r\in\mathcal{R}_i} s_{ir} \le \delta_i \qquad (i \in \mathcal{I}_\Delta). \tag{6}$$

These constraints impose, respectively, global and market-level subsidy budgets, a band on the GMV-weighted average subsidy rate, and cross-market consistency of a product's subsidy rates.

Some products belong to a primary merchandise group. A global group-order edge $e = (A_e, B_e, m_e) \in \mathcal{E}$ requires

$$\overline{s}_{A_e}(\mathbf{s}) - \overline{s}_{B_e}(\mathbf{s}) \ge m_e, \qquad \overline{s}_G(\mathbf{s}) = \frac{H_G(\mathbf{s})}{Q_G(\mathbf{s})}, \tag{7}$$

where $Q_G(\mathbf{s}) = \sum_{i\in G} \sum_{r\in\mathcal{R}_i} G_{ir}(s_{ir})$ and $H_G(\mathbf{s}) = \sum_{i\in G} \sum_{r\in\mathcal{R}_i} s_{ir} G_{ir}(s_{ir})$. For every group appearing in an order edge, the local feasible domain is assumed to satisfy $Q_G(\mathbf{s}) > 0$.

These optional constraint families are motivated by business requirements arising from our ongoing collaboration with Alibaba and by the structure of the promotion data currently shared with us through this collaboration. The data include a promotion pool of tens of thousands of products and product–market records across 46 country markets, including the US, Great Britain, Germany, Japan, and the United Arab Emirates, together with product categories, fulfillment types, and seasonal or festival tags. For example, a campaign may optimize subsidies for Home & Garden and Consumer Electronics products across multiple markets under a global budget and separate country-level budgets, require the same product to receive similar subsidy rates across countries, and cap the overall GMV-weighted average subsidy rate at 12%. Business-priority rules may further require the warehouse-fulfilled group to receive an average subsidy at least two percentage points above the just-in-time (JIT) group, impose an ordering among invited tiers $A > B > C$ with adjacent tiers separated by at least 1.5 percentage points. These constraints therefore reflect requirements arising in Alibaba's promotion operations.

3 An LLM-Configured Two-Solver Optimization Agent

Our agent separates instance-conditioned configuration from optimization. Figure 1 summarizes its end-to-end architecture, and Figure 2 in Appendix shows its user interface. Given a validated instance, it freezes the solver configuration, independently runs scalable PD and LNS+SA (Algorithms 1 and 3), and returns the best feasible policy verified by a solver-independent evaluator. Algorithm 2gives the certified PD variant. The following subsections describe the solvers and controller.

3.1 Primal–dual decomposition

For an edge e, introduce a threshold θ_e and the lifted constraints

$$h^A_e(\mathbf{s}, \theta_e) = H_{A_e}(\mathbf{s}) - (\theta_e + m_e/2) Q_{A_e}(\mathbf{s}) \ge 0, \tag{8}$$

$$h^B_e(\mathbf{s}, \theta_e) = (\theta_e - m_e/2) Q_{B_e}(\mathbf{s}) - H_{B_e}(\mathbf{s}) \ge 0. \tag{9}$$

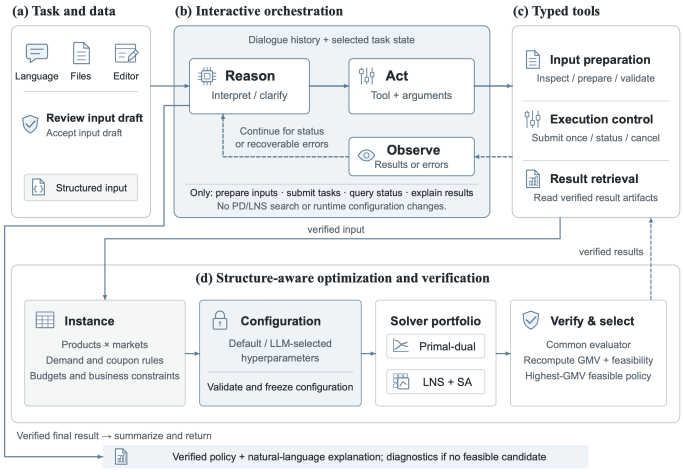

Figure 1: Overview of the optimization-agent architecture. The configuration is validated and frozen before PD and LNS+SA are executed independently; a common evaluator verifies candidate policies and returns the best feasible solution.

Let multipliers $\lambda_0, \lambda_r \geq 0$ be associated with the budget constraints 4; $\mu^+, \mu^- \geq 0$ with the upper and lower average-band constraint 5; and $\alpha_e, \beta_e \geq 0$ with the lifted constraints, respectively. For this maximization problem, the Lagrangian is

$$\mathcal{L}(\mathbf{s}, \boldsymbol{\theta}; \mathbf{y}) = F(\mathbf{s}) - \lambda_0(D_0(\mathbf{s}) - B_0) - \sum_{r \in \mathcal{R}_B} \lambda_r(D_r(\mathbf{s}) - B_r) - \mu^+(H_0(\mathbf{s}) - \overline{\rho} Q_0(\mathbf{s})) - \mu^-(\underline{\rho} Q_0(\mathbf{s}) - H_0(\mathbf{s}))$$
$$+ \sum_{e \in \mathcal{E}} \{\alpha_e h_e^A(\mathbf{s}, \theta_e) + \beta_e h_e^B(\mathbf{s}, \theta_e)\}, \tag{10}$$

where $\mathbf{y}$ collects all nonnegative multipliers. For fixed $(\mathbf{y}, \boldsymbol{\theta})$, this expression is a constant plus independent product terms. The product-level subproblems are solved in parallel, aggregate constraint residuals are computed, and the multipliers are updated using normalized projected residual steps. Because the cross-product constraints are relaxed, the resulting Lagrangian solution may violate budgets, the average-subsidy band, or group-order constraints. The workflow therefore periodically invokes the structure-aware primal recovery described next.

For each product, we split the feasible subsidy range at its coupon thresholds, optimize within each fixed-coupon segment, and compare the segment optima. This avoids treating a discontinuity as part of a smooth interval. We use two regimes for the product-level subproblems. The certified regime runs one-dimensional anchor B&B, when needed, until each product meets a prescribed product-level global-gap stopping test, yielding a valid dual upper bound under stated conditions. The scalable regime instead evaluates structured candidate solutions, uses approximate updates of the group-order thresholds, and devotes more computation to primal recovery. Its Lagrangian value is used only as a diagnostic, not as an optimality certificate.

3.2 Structure-aware primal recovery

A raw PD policy satisfies each product-local constraint, but concatenating independently optimized products can violate the shared budgets, the GMV-weighted average-subsidy band, or the group-order constraints. Primal recovery coordinates complete product assignments to remove these aggregate violations. Recovery is not invoked after every dual iteration. It is called when a periodic or timing-adaptive schedule triggers, and once more within a protected final-recovery window. On a triggered call, recovery constructs for each product i a small menu $\mathcal{V}_i = \{\mathbf{v}_i^j\}_j$, where $\mathbf{v}_i^j = (v_{ir}^j : r \in \mathcal{R}_i)$ specifies the subsidies at all active markets of that product. The menu contains the current assignment, assignments at subsidy bounds and coupon boundaries, selected assignments from earlier dual iterations, and alternatives designed to relieve the currently violated aggregate constraints. Every menu assignment is screened against the product's subsidy bounds, coupon rules, and cross-market range constraint. Let a_{ij} denote replacement of the current assignment $\mathbf{v}_i^0$ by $\mathbf{v}_i^j$, the

retained elementary actions satisfy $\mathcal{A}(i) \subseteq \{a_{ij} : \mathbf{v}_i^j \in \mathcal{V}_i \setminus \{\mathbf{v}_i^0\}\}$. Appendix B.1 describes the details of menu construction.

When group-order constraints are active, recovery first computes target GMV-weighted average subsidy rates for the merchandise groups. These targets satisfy the ordering margins while remaining close to the current group averages, and they guide the construction of upward and downward product assignments. Recovery then coordinates the resulting assignments within each group-order component. Here a component is a connected component of the undirected graph obtained by treating every declared group-order relation as an edge while ignoring its direction for connectivity. Thus relations $A > B$ and $B > C$ belong to one component, whereas an unrelated relation $D > E$ belongs to another. For a moderate-size component, a multiple-choice integer master jointly selects at most one complete candidate assignment per product using a linear prediction of the resulting resource and group-order residuals. Large components are covered by overlapping blocks centered on violated ordering relations. For example, a block for $A > B$ combines upward assignments in A with downward assignments in B; a subsequent block for $B > C$ retains selected products from B, allowing coordinated changes to propagate through the component. The target projection and component masters are detailed in Appendices B.2 and B.3.

If the component master does not reach feasibility, recovery applies a two-phase continuous restricted master. Phase I combines single-product, paired, and bundled actions to reduce the maximum normalized violation while preventing any individual constraint from becoming more violated. Once the linear prediction admits a feasible combination, Phase II selects among such combinations to minimize predicted GMV loss. The resulting fractional action weights define only a proposed direction: the original nonlinear objective and all constraint residuals are recomputed before any step is accepted. Appendix B.4 gives the formulations of the Phase I and Phase II problems.

Recovery then applies deterministic cleanup to subsidy budgets and the upper bound on GMV-weighted average subsidy, as detailed in Appendix B.5. Because this cleanup may reopen group-order violations, group projection, component cleanup, and the continuous master can be invoked again when their residuals remain. A remaining average-subsidy lower-bound violation is handled by a full-product scan on instances without group-order constraints and by a bounded cleanup on group-coupled instances. Any remaining coupled resource residuals trigger a resource-residual continuous master, followed when needed by a final cleanup of the violated group-order components. Modules whose constraints are inactive or already feasible are skipped; Recovery stops its feasibility stages once a feasible policy is obtained. Appendix B.6 gives the complete recovery orchestration.

During the protected final recovery, a feasible policy is further improved through single-product replacements, budget-transferring paired moves, and small joint replacements within group-order components. A polished candidate replaces the incumbent only when evaluation under the original nonlinear model verifies every constraint and confirms a strict GMV improvement. The best previously verified feasible policy is retained throughout, so a later unsuccessful repair or timeout cannot erase it. Appendix B.7 gives the polishing and time-allocation details.

3.3 Structure-aware adaptive LNS with feasibility-aware SA acceptance

Unlike standard LNS, our method searches over complete multi-market product assignments, constructs resource-aware, group-aware, and coupon-aware neighborhoods, and jointly repairs each neighborhood with a restricted multiple-choice master. The search first constructs up to five complementary starting policies using productwise maximum-GMV, minimum-cost, minimum-mean-subsidy, maximum-mean-subsidy, and group-order-oriented assignments. It designates a preferred starting policy according to the enabled constraint types.

At each iteration, an adaptive destroy operator selects a product neighborhood $\mathcal{N} \subseteq \mathcal{I}$. Its sampling probability depends on the performance obtained by that operator in earlier iterations. Generic operators consider random or stale products, whereas structure-aware operators target products contributing to tight resource constraints, products near coupon thresholds, or products linked by the same group-order component. For each destroyed product i, LNS constructs a menu $\mathcal{V}_i$ of locally valid assignments, including the current assignment, subsidy-bound assignments, coupon-regime endpoints, and locally refined alternatives. A product can therefore move directly between separated coupon regimes without passing through a sequence of invalid local interpolations.

For a general destroyed neighborhood, LNS adaptively selects among greedy, randomized-greedy, regret-based, and restricted multiple-choice repair operators according to their observed performance per unit computation time. A complete group-order neighborhood bypasses this sampling and uses either the component-aware group master repair or its refined variant, depending on the

search stage. For an ordering relation $A > B$ with margin m, it fixes a separator ϑ and imposes

$$H_A - (\vartheta + m)Q_A \geq 0, \qquad H_B - \vartheta Q_B \leq 0, \tag{11}$$

which imply $\overline{s}_A \geq \overline{s}_B + m$. This master coordinates multi-product changes across budgets, coupon regimes, and both sides of an ordering relation, distinguishing our method from greedy repair.

Every repaired policy is re-evaluated under the original nonlinear model. A verified feasible improvement may update the incumbent if it has a higher objective, while SA separately decides whether the proposal becomes the current search state. Before the first feasible policy, the energy prioritizes normalized constraint violations and gives GMV increasing weight near feasibility. After feasibility is attained, LNS gradually reduces the constraint violation allowed in exploratory proposals. After each iteration, the selected destroy and repair operators are rewarded according to improvement per unit running time, and their future sampling probabilities are updated. The temperature follows an elapsed-time cooling schedule and is temporarily reheated after search stagnation. Appendix C gives the complete LNS algorithm and formulations.

3.4 Agent Configuration and Solver Selection

The agent defines a predeclared finite set $\mathcal{H}$ of admissible hyperparameter configurations for the two solvers. An algorithm card describes the hyperparameter options, and an instance card summarizes the problem scale and constraint structure. The controller either uses a default configuration $\mathbf{h}_0$ or asks an LLM to select $\mathbf{h} \in \mathcal{H}$ before solver execution begins. The selected configuration is validated against a closed schema and then frozen. Once optimization begins, the configuration, instance, solver logic, seeds, and computational budgets remain fixed. Thus the LLM acts as an instance-conditioned configurator. The configuration used in our case study is reported in Appendix D.

Let $\mathbf{s}_{\mathrm{PD}}(t)$ and $\mathbf{s}_{\mathrm{LNS},k}(t)$ be the best verified incumbents by time t, where the predeclared set $\mathcal{K}$ contains one or more LNS seeds. After freezing the configuration, the agent retains

$$\mathbf{s}_{\mathrm{port}}(t) \in \arg \max_{\mathbf{s} \in \{\mathbf{s}_{\mathrm{PD}}(t)\} \cup \{\mathbf{s}_{\mathrm{LNS},k}(t) : k \in \mathcal{K}\}} F(\mathbf{s}). \tag{12}$$

The controller runs the solvers independently. Each candidate policy is re-evaluated by the common evaluator and replaces the current portfolio solution only if it is verified feasible and has a higher recomputed objective. The solvers exchange neither internal states nor solutions. When separate computational resources are available, they may run in parallel; otherwise, they may run sequentially, with their trajectories compared using the elapsed optimization time of each solver.

4 Theoretical Properties

This section establishes the structural properties underlying the PD method: exact product-wise decomposition after dualizing the aggregate constraints, globally certifiable solution of range-constrained product subproblems, component-wise treatment of the auxiliary group-order thresholds, and the resulting primal–dual objective bound. Proofs and implementation-level edge cases are deferred to Appendix A.

Theorem 1 (Product-wise Lagrangian decomposition)**.** *Fix $\mathbf{y} \geq 0$ and $\boldsymbol{\theta}$. After dualizing all enabled aggregate constraints, the remaining domain is the product-local set $\mathcal{X} = \prod_i \mathcal{X}_i$, and there are a constant $K(\mathbf{y})$ and functions ℓ_i such that*

$$\sup_{\mathbf{s} \in \mathcal{X}} \mathcal{L}(\mathbf{s}, \boldsymbol{\theta}; \mathbf{y}) = K(\mathbf{y}) + \sum_{i \in \mathcal{I}} \sup_{\mathbf{s}_i \in \mathcal{X}_i} \ell_i(\mathbf{s}_i; \boldsymbol{\theta}, \mathbf{y}). \tag{13}$$

The decomposition reduces the dual evaluation to independent product subproblems. We next show how each such subproblem can be globally certified by one-dimensional B&B. For each product–market pair (i, r), the coupon thresholds partition the locally feasible subsidy domain into finitely many fixed-coupon regimes J_{irk}, on each of which the coupon amount is constant; each threshold is assigned to exactly one adjacent regime according to the deterministic coupon rule. Because q_{ir} is exponential, G_{ir} and D_{ir} are affine-times-exponential, while $s_{ir}G_{ir}$ is quadratic-times-exponential; the corresponding market term in ℓ_i has the form

$$T_{irk}(s_{ir}) = K_{irk} e^{\xi_{ir} s_{ir}} (A_{irk} + B_{irk} s_{ir} + C_{irk} s_{ir}^2), \tag{14}$$

where $K_{irk} > 0$, ξ_{ir}, and $A_{irk}, B_{irk}, C_{irk}$ are constants determined by the instance data, regime k, and the fixed dual quantities. Its derivative is

$$\frac{dT_{irk}}{ds_{ir}} = K_{irk} e^{\xi_{ir} s_{ir}} \left[\xi_{ir} C_{irk} s_{ir}^2 + (\xi_{ir} B_{irk} + 2C_{irk}) s_{ir} + (\xi_{ir} A_{irk} + B_{irk}) \right]. \tag{15}$$

Here the interior stationary points are exactly the feasible roots of the quadratic expression in brackets. On each feasible portion of J_{irk}, the regime supremum is determined by feasible interior stationary points, included interval endpoints, and one-sided limits at excluded coupon-boundary endpoints. Enumerating these candidates over all regimes gives the exact market-level supremum used within the product subproblem.

Theorem 2 (Certified anchor B&B for a product subproblem)**.** *Fix $(\mathbf{y}, \boldsymbol{\theta})$ and product i with local bounds, a finite coupon partition, and range limit δ_i. Introduce an anchor z and define*

$$Z_i = \left[\max_r(\underline{s}_{ir} - \delta_i), \min_r \overline{s}_{ir}\right], \qquad I_{ir}(z) = [\max\{\underline{s}_{ir}, z\}, \min\{\overline{s}_{ir}, z + \delta_i\}], \tag{16}$$

$$V_i^\star = \sup_{z \in Z_i} \Phi_i(z), \qquad \Phi_i(z) = \sum_r \max_k \sup_{s_{ir} \in I_{ir}(z) \cap J_{irk}} T_{irk}(s_{ir}). \tag{17}$$

Then (i) $V_i^\star$ is the supremum value of the range-constrained product subproblem. (ii) For fixed z, r, k, the regime supremum is obtained by comparing feasible interior stationary points from equation 15, endpoints that belong to the regime, and one-sided limits at coupon boundaries excluded from the regime. (iii) A one-dimensional branch-and-bound (B&B) over z provides a globally valid certificate for the outer problem. For an anchor node $J = [a, b] \subseteq Z_i$, define $\widehat{I}_{ir}(J) = [\max\{\underline{s}_{ir}, a\}, \min\{\overline{s}_{ir}, b + \delta_i\}]$, $U_i(J) = \sum_r \max_k \sup_{s_{ir} \in \widehat{I}_{ir}(J) \cap J_{irk}} T_{irk}(s_{ir})$. Since $I_{ir}(z) \subseteq \widehat{I}_{ir}(J)$ for every $z \in J$, $U_i(J) \geq \sup_{z \in J} \Phi_i(z)$ is its upper bound. Let L_i be the best feasible product value found and $\mathcal{Q}$ be the set of active anchor nodes. Then $L_i \leq V_i^\star \leq \max_{J \in \mathcal{Q}} U_i(J)$. Hence, when $\max_{J \in \mathcal{Q}} U_i(J) \leq L_i + \varepsilon$, the algorithm certifies $0 \leq V_i^\star - L_i \leq \varepsilon$.

Theorem 2 certifies each product subproblem for fixed group-order thresholds. It remains to optimize over the threshold vector $\boldsymbol{\theta}$. Choose $\Theta = \prod_e [\theta_e^L, \theta_e^U]$ such that every primal-feasible policy admits separators in Θ, and define $\Psi(\boldsymbol{\theta}) = \sup_{\mathbf{s} \in \mathcal{X}} \mathcal{L}(\mathbf{s}, \boldsymbol{\theta}; \mathbf{y})$. Construct the threshold-incidence graph between group-order edges and products, connecting edge e to product i whenever i contributes to A_e or B_e.

Theorem 3 (Threshold corners and component factorization)**.** *For fixed $\mathbf{y}$, $\Psi(\boldsymbol{\theta})$ is convex in $\boldsymbol{\theta}$, and $\max_{\boldsymbol{\theta} \in \Theta} \Psi(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta} \in \text{corners}(\Theta)} \Psi(\boldsymbol{\theta})$. If the incidence graph has connected components $c \in \mathcal{C}$, let $\mathcal{E}_c$ denote the group-order edges in component c and define $D_c = |\mathcal{E}_c|$, $\boldsymbol{\theta}_c = (\theta_e : e \in \mathcal{E}_c)$, and $\Theta_c = \prod_{e \in \mathcal{E}_c} [\theta_e^L, \theta_e^U]$, then $\max_{\boldsymbol{\theta} \in \Theta} \Psi(\boldsymbol{\theta}) = C_0 + \sum_{c \in \mathcal{C}} \max_{\boldsymbol{\theta}_c \in \text{corners}(\Theta_c)} \Psi_c(\boldsymbol{\theta}_c)$, where C_0 is the contribution independent of group thresholds and Ψ_c is the contribution of component c. Thus only $\sum_c 2^{D_c}$ component corners need be evaluated, rather than 2^D global corners, where $D = \sum_c D_c$.*

5 Experiments

5.1 Experimental protocol

We use the same problem instances and a solver-independent evaluator for PD, LNS+SA, and a monolithic Gurobi 13 baseline. Our synthetic instances are designed to reflect the multi-market structure, demand characteristics, and optional business constraints observed in our ongoing collaboration with Alibaba. Runs are performed on an Apple M1 Max laptop with 64 GB RAM. Each isolated solver process is limited to eight CPU cores and 900 seconds of optimization time; instance construction and final verification are measured separately. The frozen benchmark suite contains 72 feasible instances. For each $N \in \{100, 1000, 10000\}$, it includes three instances from each of six families: `smooth_resources`, `sparse_markets`, `code_thresholds`, `group_components`, `code_group_coupled`, and `tight_mixed`. For each $N \in \{50000, 100000\}$, it includes three instances from three representative families: `smooth_resources`, `code_group_coupled`, and `tight_mixed`. The first four families isolate major structural features, whereas the last two combine resource pressure with coupon discontinuities and group-order constraints. Appendix Table 3 summarizes their construction. PD and Gurobi run once per instance, while three predeclared LNS seeds quantify stochastic variation. For each instance, the portfolio returns the feasible solution with the highest common-evaluator objective among one PD run and the three LNS+SA runs. We report verified feasibility and objective quality, Gurobi bounds and gaps, time to the first feasible solution, and variation across LNS+SA seeds. Separately, a predeclared single-instance case study evaluates the LLM-based configuration interface.

5.2 Main held-out comparison

PD performs strongly within the 900-second budget, finding a verified feasible solution on 71/72 instances; best-of-three LNS covers all 72. On the 71 commonly feasible cases, PD achieves the

Table 1: Held-out results at 900 seconds. "LNS runs" counts feasible runs over three predeclared seeds. Mean $\Delta_{\mathrm{PD-LNS}}$ compares PD with the best feasible objective value across the three LNS seeds for each instance. Winner counts compare PD, best-of-three LNS, and the Gurobi incumbent.

N	Instances	PD feasible	LNS runs	Gurobi feasible	Mean $\Delta_{\mathrm{PD-LNS}}$	Winners PD/LNS/Gurobi
100	18	18	54/54	18	-0.033%	0/2/16
1000	18	18	54/54	9	$+0.015\%$	6/6/6
10,000	18	18	54/54	5	$+0.230\%$	11/2/5
50,000	9	8	27/27	1	$+1.163\%$	7/1/1
100,000	9	9	27/27	0	$+1.417\%$	9/0/0
Total	72	71	216/216	33	–	33/11/28

Table 2: Mechanism ablation at 900 seconds. "Mean Full improvement" is $100(F_{\mathrm{Full}} - F_{\mathrm{variant}})/F_{\mathrm{variant}}$ on common-feasible cases; LNS feasibility counts runs over three seeds.

(a) PD

Variant	Feasible	Mean Full improvement
Full PD	6/6	–
PD: generic greedy recovery	3/6	$+0.7162\%$
PD: no earlier-iterate assignments	6/6	$+0.0044\%$
PD: no feasible polishing	6/6	$+0.0398\%$
PD: no group-component multiple-choice master	3/3	$+0.1170\%$

(b) LNS+SA

Variant	Feasible	Mean Full improvement
Full LNS+SA	18/18	–
LNS: random/vanilla neighborhoods	17/18	$+1.4435\%$
LNS: no SA acceptance	18/18	$+0.0061\%$
LNS: no group-component search	9/9	$+0.4600\%$

higher objective on 45 instances and LNS on 26. Their relative performance also varies with scale: PD trails LNS by a mean 0.033% (median 0.015%) at $N = 100$, is effectively tied at $N = 1000$ (mean $+0.015\%$; median -0.003%), and leads by mean 0.230%, 1.163%, and 1.417% (median 0.206%, 0.996%, and 1.178%) at $N = 10{,}000$, $N = 50{,}000$, and $N = 100{,}000$, respectively. LNS reaches feasibility earlier at $N = 10{,}000$ and $N = 50000$ (median 55.9 and 290.5 seconds, versus 78.2 and 466.8 for PD), while PD is both faster and higher-quality at $N = 100{,}000$. Thus, LNS provides a useful complement to an increasingly strong PD method at large scale.

The combined PD–LNS portfolio is feasible on all 72 instances, whereas Gurobi finds a verified incumbent on only 33: all 18 at $N = 100$, 9 at $N = 1000$, 5 at $N = 10{,}000$, 1 at $N = 50{,}000$, and none at $N = 100{,}000$. Across all 72 instances, the portfolio provides the best observed incumbent on 44 instances, compared with 28 for Gurobi. Conditional on the 33 cases where Gurobi is feasible, the portfolio beats its incumbent on five and otherwise remains very close, with mean and median objective shortfalls of only 0.0059% and 0.0054%. The median relative gap between Gurobi's upper bound and the portfolio objective increases from 0.57% at $N = 100$ to 1.96% at $N = 10{,}000$ and approximately 27% at $N \geq 50{,}000$. The main advantage of the portfolio is therefore reliable high-quality feasibility within the fixed time budget. Appendix D reports additional uncertainty and seed-level results.

5.3 Mechanism ablation

Table 2 compares the complete PD and LNS+SA methods against controlled variants that remove selected algorithmic mechanisms, using six frozen $N = 10{,}000$ ablation instances, one per family. LNS+SA uses three fixed seeds; its objective differences are computed using the best verified run for each instance. The PD group-component ablation removes its component-level multiple-choice recovery master, while the corresponding LNS ablation removes component-based initialization and group-component neighborhoods. These two ablations are evaluated only on the three families containing group-order constraints.

The results indicate that structure-aware coordination primarily improves feasibility robustness and also provides modest objective gains. Generic PD recovery fails on all three group-order instances, while removing only the group-component multiple-choice master reduces the PD objective by 0.117% on average. Full structure-aware LNS improves over vanilla LNS by 1.4435%. On the group-order families, removing group-component search reduces its objective by 0.4600% on average, compared with 0.0061% when SA acceptance is removed. Appendix D gives per-family results.

5.4 Certified PD bounds and a discretized Gurobi benchmark

To measure the optimality gap of the feasible solution returned by certified PD, we compare its objective with the rigorous upper bound computed by the same method. We generate six fresh small instances with $N = 12$ and $R = 2$, one per family, and run certified PD for 3600 seconds. On

every instance it returns a verified feasible objective F_{PD} and a rigorous upper bound U_{PD}, thereby certifying $F_{\mathrm{PD}} \leq F^{\star} \leq U_{\mathrm{PD}}$. The certified gap $100(U_{\mathrm{PD}} - F_{\mathrm{PD}})/F_{\mathrm{PD}}$ is 0.578% on average (median 0.367%). It is below 0.8% on five of the six instances and reaches 1.909% on the remaining instance. Appendix Table 7 gives the per-family results. We also test whether discretizing subsidy decisions makes the problem substantially easier for Gurobi. With finite candidate sets built from a 0.01 base grid and structural boundary points, 900-second Gurobi MILP runs find an incumbent on only 9 of the 18 held-out instances with $N = 10{,}000$. Appendix Table 8 provides detailed results.

5.5 LLM-CONFIGURED AGENT CASE STUDY

We demonstrate the complete agent workflow on a fresh 50,000-product, eight-market `code_group_coupled` instance. Given a concise description of the two solvers, their configurable hyperparameters, and a numerical summary of the instance, one `gpt-6-astra` call selected five hyperparameters from predefined finite options: the multiplier-update step size and the time reserved for final feasibility recovery in PD, together with the number of candidate product decisions, the maximum size of a group-based neighborhood, and the time limit for each restricted optimization problem in LNS. The selected configuration was automatically validated and frozen before optimization. The agent then ran PD and LNS independently for 900 seconds on eight CPU cores, verified both returned policies with the common evaluator, and returned the higher-objective feasible policy, which was produced by PD. This case study demonstrates a pipeline from instance-conditioned LLM configuration to verified optimization output. Appendix Table 9 reports the selected configuration, solver outcomes, and comparison with the frozen default configuration.

6 RELATED WORK

Our PD extends Lagrangian decomposition (Fisher, 1981) by reformulating group-order constraints to preserve product-wise separability, globally certifying the resulting single-product subproblems, processing independent groups of ordering constraints separately, and performing problem-specific feasibility recovery. Unlike generic LNS (Shaw, 1998; Ropke & Pisinger, 2006), ours jointly changes the subsidy decisions of a product across its active markets, constructs neighborhoods from shared resource and group-order constraints, and uses simulated-annealing acceptance (Kirkpatrick et al., 1983). Algorithm portfolios exploit complementary search (Gomes & Selman, 2001); our agent adds a constrained one-shot LLM configuration step while keeping configuration validation, optimization, solution verification, and final selection explicit.

Related promotion-optimization work has primarily studied multi-period retail promotion planning and personalized promotion design. Retail models optimize promotion prices for one or multiple items under business rules using LP/IP approximations or geometric-program reformulations (Cohen et al., 2017; 2021; Khandan & Hoseinpour, 2025), while recent industrial systems optimize large-scale promotion calendars using MILP (Llenas et al., 2026). Personalized retail promotion has also been studied through customer-level choice modeling and individualized offer design (Jagabathula et al., 2022). Our setting instead considers single-campaign, multi-country product–market subsidy allocation with continuous subsidy decisions, exponential demand, coupon-tier discontinuities, and multiple nonlinear coupling constraints.

7 CONCLUSION AND FUTURE WORK

We present a structure-aware PD–LNS portfolio that produces high-quality feasible policies for large-scale promotion optimization under short computational budgets, together with an optimization-agent framework for instance-conditioned configuration and solution verification. Although motivated by concrete operational requirements arising from our collaboration with Alibaba, the proposed methods exploit common optimization structures: separable nonlinear decisions coupled by shared resources, discontinuous threshold rules, and priorities across decision groups. As part of our ongoing collaboration, the framework will be further evaluated through internal experiments and pilot applications at Alibaba. More broadly, the primal–dual decomposition, structure-aware feasibility recovery, and customized neighborhood search form a modular framework that can extend beyond promotion optimization. Future work will evaluate and adapt this framework to other large-scale resource-allocation, pricing, and planning problems with similar nonlinear and combinatorial structure.

AI USE STATEMENT

Generative AI tools were used to assist with methodological and experimental design, theoretical analysis and proof development, code implementation, experimental orchestration, and drafting and editing the manuscript. All AI-assisted mathematical claims, proofs, implementations, experiments, and reported results were independently reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

The framework optimizes promotional subsidies across products and markets, and its recommendations depend on the supplied data, business constraints, and merchandise-group definitions. Any production deployment should therefore review these inputs and the resulting allocation outcomes for their operational implications. The experiments in this paper use synthetic instances and do not involve customer-level personal data.

REPRODUCIBILITY STATEMENT

The appendix provides complete proofs, detailed algorithmic descriptions, experimental protocols, and benchmark specifications. The supplementary material includes the instance-generation code and manifests, solver configurations and seeds, returned solutions and independent evaluation reports, together with the agent interface, instance summaries and solver descriptions supplied to the LLM, allowed hyperparameter options, original LLM responses, and validated configurations. These materials record the information needed to reproduce the reported optimization runs and independently verify their objectives and constraint feasibility.

REFERENCES

Maxime C. Cohen, Ngai-Hang Zachary Leung, Kiran Panchamgam, Georgia Perakis, and Anthony Smith. The impact of linear optimization on promotion planning. *Operations Research*, 65(2): 446–468, 2017. doi: 10.1287/opre.2016.1573.

Maxime C. Cohen, Jeremy J. Kalas, and Georgia Perakis. Promotion optimization for multiple items in supermarkets. *Management Science*, 67(4):2340–2364, 2021. doi: 10.1287/mnsc.2020.3641.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1–2):43–62, 2001.

Srikanth Jagabathula, Dmitry Mitrofanov, and Gustavo Vulcano. Personalized retail promotions through a directed acyclic graph–based representation of customer preferences. *Operations Research*, 70(2):641–665, 2022. doi: 10.1287/opre.2021.2108.

Melika Khandan and Pooya Hoseinpour. Incorporating promotional effects in sales planning of the retail industry using geometric programming. *INFORMS Journal on Computing*, 37(5):1284–1305, 2025. doi: 10.1287/ijoc.2023.0275.

Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

Aleix Llenas, Eduardo Salazar-Treviño, Francisco Leskovar, Francesc Pons-Llopis, Federico Todeschini, Deepika Gowda, David Bofill, Laxmi Anish, and Michael Cleavinger. Pepsico deploys ai-driven pricing and promotion optimization at scale. *INFORMS Journal on Applied Analytics*, 2026. doi: 10.1287/inte.2025.0302. Articles in Advance.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98*, pp. 417–431. Springer, 1998.

APPENDIX

A PROOFS

A.1 THRESHOLD LIFTING

Proposition 4 (Threshold lifting)**.** *For a group-order edge e with $Q_{A_e}, Q_{B_e} > 0$, the condition in equation 7 holds if and only if there exists θ_e satisfying inequalities 8 and 9.*

Proof. Dividing inequalities 8 and 9 by their positive GMV denominators gives $\overline{s}_{A_e} \geq \theta_e + m_e/2$ and $\overline{s}_{B_e} \leq \theta_e - m_e/2$. Subtracting the second inequality from the first gives equation 7. Conversely, the condition in equation 7 is equivalent to $\overline{s}_{B_e} + m_e/2 \leq \overline{s}_{A_e} - m_e/2$; hence any θ_e in the resulting nonempty interval satisfies both lifted inequalities. □

A.2 PROOF OF THEOREM 1

Proof. Expand every aggregate in equation 10. Budget and average-band terms are sums over active product–market cells, and each lifted group term is a sum over products in its higher or lower group. Hence all terms depending on $\mathbf{s}_i$ can be collected into ℓ_i, while multiplier–right-hand-side terms give $K(\mathbf{y})$. After dualizing the aggregate constraints, the remaining domain is exactly $\mathcal{X} = \prod_i \mathcal{X}_i$. Taking the supremum of this separable sum over the Cartesian product yields equation 13; no attainment assumption is needed. □

A.3 COUPON-REGIME FORM OF A MARKET TERM

Recall that J_{irk} is the locally feasible subsidy interval associated with coupon regime k. Fix a coupon regime k with coupon amount c_{irk}. Writing $\gamma_{irk} = c_{irk}/p^0_{ir}$ and $\xi_{ir} = b_{ir}p^0_{ir}$ gives

$$q_{ir}(s) = a_{ir}e^{-\xi_{ir}(1-\gamma_{irk})}e^{\xi_{ir}s}, \quad G_{ir}(s) = \kappa_{irk}e^{\xi_{ir}s}(1 - \gamma_{irk} - s),$$

for a positive constant κ_{irk}. Likewise $D_{ir}(s) = \kappa_{irk}e^{\xi_{ir}s}(s + \gamma_{irk})$, up to the same fixed price/exchange-rate scaling already absorbed in κ_{irk}. The only additional cellwise term created by the average constraints is $sG_{ir}(s)$, which is quadratic-times-exponential. Therefore every market contribution to ℓ_i is a linear combination of G_{ir}, D_{ir}, and sG_{ir} and has the form in equation 14.

A.4 PROOF OF THEOREM 2

Proof. If $\mathbf{s}_i$ obeys equation 6, set $z = \min_r s_{ir}$. Then $z \leq s_{ir} \leq z + \delta_i$ for every market and the local bounds place z in Z_i. Conversely, any $z \in Z_i$ and $s_{ir} \in I_{ir}(z)$ satisfy the range constraint. This proves the anchor equivalence.

For fixed z, the feasible set is a Cartesian product across markets. On the interior of coupon regime k, differentiating equation 14 gives the positive exponential factor times the quadratic expression in equation 15. Hence the supremum on that regime is determined by feasible stationary roots, included endpoints, and one-sided endpoint limits when a coupon boundary is excluded from the regime. Because the deterministic coupon rule assigns each threshold to a specific tier, the threshold itself is evaluated using that assigned tier; when an adjacent regime excludes the threshold, its one-sided endpoint limit is retained only as a supremum candidate. Taking the largest value over the finitely many regimes therefore gives the exact market supremum for fixed z.

For $z \in J = [a, b]$, direct comparison of endpoints gives $I_{ir}(z) \subseteq \widehat{I}_{ir}(J)$, where

$$\widehat{I}_{ir}(J) = [\max\{\underline{s}_{ir}, a\}, \min\{\overline{s}_{ir}, b + \delta_i\}], \qquad U_i(J) = \sum_r \max_k \sup_{s_{ir} \in \widehat{I}_{ir}(J) \cap J_{irk}} T_{irk}(s_{ir}).$$

Therefore $U_i(J) \geq \sup_{z \in J} \Phi_i(z)$. Thus every feasible product assignment gives a lower bound L_i, while $\max_{J \in \mathcal{Q}} U_i(J)$ is an upper bound on $V_i^\star$. Consequently, whenever the stated stopping test is met, $0 \leq V_i^\star - L_i \leq \varepsilon$. This certification claim requires only validity of the node bounds, not a separate assertion that branch-and-bound must reach the stopping test in finite time. □

A.5 PROOF OF THEOREM 3

Proof. For fixed $\mathbf{s}$, the Lagrangian in equation 10 is affine in $\boldsymbol{\theta}$. Hence $\Psi(\boldsymbol{\theta})$, as a pointwise supremum of affine functions, is convex. A convex function on a compact box attains its supremum at a corner: any point in the box is a convex combination of corners, and convexity bounds its

value by the corresponding convex combination of corner values. Each product term depends only on thresholds of group-order edges incident to that product. Therefore terms belonging to distinct connected components of the threshold-incidence graph share no threshold coordinate. Collecting them gives $C_0 + \sum_c \Psi_c(\boldsymbol{\theta}_c)$ over a Cartesian product of component boxes, so the component maximizations separate. A component with D_c threshold coordinates has 2^{D_c} corners, yielding $\sum_c 2^{D_c}$ component-corner evaluations rather than $2^{\sum_c D_c}$ global corners. □

A.6 Certified primal–dual upper bound

Proposition 5 (Certified PD upper bound)**.** *Let $F^\star := \sup_{\mathbf{s}\in\mathcal{S}} F(\mathbf{s})$. Suppose Θ contains at least one valid separator vector for every primal-feasible $\mathbf{s} \in \mathcal{S}$. For any nonnegative multiplier vector $\mathbf{y}$, define*

$$g(\mathbf{y}) = \max_{\boldsymbol{\theta}\in\Theta} \sup_{\mathbf{s}\in\mathcal{X}} \mathcal{L}(\mathbf{s}, \boldsymbol{\theta}; \mathbf{y}).$$

For fixed $(\boldsymbol{\theta}, \mathbf{y})$, let

$$V_i^\star(\boldsymbol{\theta}, \mathbf{y}) = \sup_{\mathbf{s}_i\in\mathcal{X}_i} \ell_i(\mathbf{s}_i; \boldsymbol{\theta}, \mathbf{y})$$

be product i's exact subproblem value, and let $\overline{V}_i(\boldsymbol{\theta}, \mathbf{y})$ be a certified upper bound satisfying

$$\overline{V}_i(\boldsymbol{\theta}, \mathbf{y}) \geq V_i^\star(\boldsymbol{\theta}, \mathbf{y}).$$

Define

$$\overline{g}(\mathbf{y}) := \max_{\boldsymbol{\theta}\in\text{corners}(\Theta)} \left\{ K(\mathbf{y}) + \sum_{i\in\mathcal{I}} \overline{V}_i(\boldsymbol{\theta}, \mathbf{y}) \right\},$$

with the equivalent component-wise evaluation of Theorem 3 used in computation. Then

$$F^\star \leq g(\mathbf{y}) \leq \overline{g}(\mathbf{y}).$$

Consequently,

$$U_{\text{PD}} = \min_t \overline{g}(\mathbf{y}^t)$$

over certified PD iterations is also a valid upper bound on $F^\star$.

Proof. Let $\mathbf{s} \in \mathcal{S}$ be any primal-feasible policy. By assumption, there exists a valid separator vector $\boldsymbol{\theta}(\mathbf{s}) \in \Theta$ for this policy. Because $\mathbf{s}$ satisfies all original aggregate constraints, every relaxed constraint contributes a nonnegative term to the Lagrangian under the sign convention in equation 10. Hence

$$\mathcal{L}\big(\mathbf{s}, \boldsymbol{\theta}(\mathbf{s}); \mathbf{y}\big) \geq F(\mathbf{s}).$$

Since $\mathbf{s} \in \mathcal{X}$ and $\boldsymbol{\theta}(\mathbf{s}) \in \Theta$,

$$F(\mathbf{s}) \leq \max_{\boldsymbol{\theta}\in\Theta} \sup_{\mathbf{s}'\in\mathcal{X}} \mathcal{L}(\mathbf{s}', \boldsymbol{\theta}; \mathbf{y}) = g(\mathbf{y}).$$

Taking the supremum over all primal-feasible $\mathbf{s} \in \mathcal{S}$ gives

$$F^\star \leq g(\mathbf{y}).$$

For fixed $(\boldsymbol{\theta}, \mathbf{y})$, Theorem 1 gives

$$\sup_{\mathbf{s}\in\mathcal{X}} \mathcal{L}(\mathbf{s}, \boldsymbol{\theta}; \mathbf{y}) = K(\mathbf{y}) + \sum_{i\in\mathcal{I}} V_i^\star(\boldsymbol{\theta}, \mathbf{y}).$$

By definition of the certified product upper bounds,

$$\overline{V}_i(\boldsymbol{\theta}, \mathbf{y}) \geq V_i^\star(\boldsymbol{\theta}, \mathbf{y})$$

for every product i and every threshold corner that must be evaluated. Therefore,

$$K(\mathbf{y}) + \sum_{i\in\mathcal{I}} V_i^\star(\boldsymbol{\theta}, \mathbf{y}) \leq K(\mathbf{y}) + \sum_{i\in\mathcal{I}} \overline{V}_i(\boldsymbol{\theta}, \mathbf{y})$$

at every such corner.

By Theorem 3, maximizing the exact Lagrangian value over Θ requires only corner evaluations, which can be factorized by threshold-incidence components. At every such corner, replacing $V_i^\star$ by $\overline{V}_i$ can only increase the evaluated value. Therefore the explicitly defined quantity $\overline{g}(\mathbf{y})$ satisfies

$$g(\mathbf{y}) \le \overline{g}(\mathbf{y}).$$

Combining this with the first part of the proof yields

$$F^\star \le g(\mathbf{y}) \le \overline{g}(\mathbf{y}).$$

Finally, every certified PD iteration t produces an upper bound $\overline{g}(\mathbf{y}^t) \ge F^\star$. Taking the minimum over such bounds therefore preserves validity:

$$U_{\mathrm{PD}} = \min_t \overline{g}(\mathbf{y}^t) \ge F^\star.$$

□

Remark. Since $0 \le \overline{s}_G \le 1$, Proposition 4 implies that the universal box $\Theta = \prod_e [m_e/2,\, 1 - m_e/2]$ contains a valid separator vector for every primal-feasible policy and hence satisfies the assumption of Proposition 5.

B Structure-Aware Primal Recovery Details

The two PD regimes share the same multiplier updates and retain their best evaluator-verified feasible policies, but differ in how they solve the product subproblems and the group-separator master. Algorithm 1 is the large-scale solver used in the main experiments; Algorithm 2 is the certifying solver used in the small-scale experiments.

Algorithm 1: Scalable PD

Input: instance data; initial nonnegative multipliers $\mathbf{y}^0$ and group separators $\boldsymbol{\theta}^0$; step scale α_0; time budget T; a periodic or timing-adaptive recovery schedule; and a protected final-recovery rule.

Initialize: set the best feasible policy verified under the original nonlinear constraints to empty, and initialize a bounded archive of complete product assignments generated across dual iterations.

Repeat for $t = 0, 1, \ldots$ **until the protected final recovery has completed or the time budget is exhausted:**

1. With $(\mathbf{y}^t, \boldsymbol{\theta}^t)$ fixed, solve all product subproblems in parallel by evaluating structurally motivated candidates, including subsidy bounds, coupon boundaries, and stationary points. Concatenate the resulting product assignments into a raw policy $\mathbf{s}^t$.
2. Recompute the original objective and constraint residuals of $\mathbf{s}^t$. For each product, retain a bounded set of representative assignments generated in the current and earlier dual iterations.
3. Determine whether primal recovery is due. Recovery is invoked only when the periodic or timing-adaptive recovery schedule triggers, or when the run enters its protected final-recovery window. If recovery is not due, skip to Step 5.
4. On a triggered recovery call, form each product's recovery menu. Apply the following residual-driven recovery procedure:
   (a) If group-order constraints are active, first project the current group averages toward values satisfying the ordering margins, and then solve the group-component multiple-choice masters. If this produces a feasible policy, skip the remaining feasibility-recovery stages. The projection and component-level masters are detailed in Appendices B.2 and B.3.
   (b) If infeasibility remains, apply the pre-cleanup two-phase continuous recovery master to coordinate product-level changes across the currently violated resource and group-order constraints, as described in Appendix B.4.
   (c) Apply deterministic cleanup for market budgets, the global budget, and the upper bound on the GMV-weighted average subsidy, using range-aware fallback moves when necessary. The candidate assignments and exact acceptance rules used by these cleanup procedures are described in Appendices B.1 and B.5
   (d) If group-order constraints remain violated, repeat group-average projection and targeted component cleanup. Invoke a post-cleanup continuous recovery master and bounded rescue rounds only when the corresponding residuals remain. This residual-dependent orchestration is described in Appendix B.6, using the modules in Appendices B.2, B.3, and B.4.
   (e) Repair any remaining lower-bound violation on the GMV-weighted average subsidy. When no group-order constraint is active, scan all products for valid upward assignments; on group-coupled instances, use a bounded average-floor cleanup to avoid reopening repaired group-order violations. The full-product scan and the bounded group-aware cleanup are described in Appendix B.1, and their position in the complete recovery sequence is given in Appendix B.6.
   (f) If resource residuals remain, invoke the resource-residual continuous master. Then apply a final narrow cleanup to any remaining violated group-order component. The continuous master is described in Appendix B.4, while the final component cleanup and its position in the recovery sequence are described in Appendix B.3 and Appendix B.6, respectively.

   Recovery modules whose associated constraints are inactive, whose residuals are already feasible, or whose execution would exceed the remaining recovery time are skipped. If feasibility is reached during an intermediate stage, the remaining feasibility-recovery stages may also be skipped.

   During the protected final recovery, treat the best feasible policy found so far as the incumbent and apply feasible polishing described in Appendix B.7. Accept a polished policy only if evaluation under the original nonlinear model confirms feasibility and objective improvement.
5. Update each nonnegative multiplier using the residual of its corresponding dualized constraint:

$$\mathbf{y}^{t+1} = \left[\mathbf{y}^t + \alpha_t \mathbf{r}^t\right]_+, \qquad \alpha_t = \frac{\alpha_0}{\sqrt{t+1}},$$

   where $\mathbf{r}^t$ contains the residuals of the raw policy $\mathbf{s}^t$. For each group-order relation, update its auxiliary separator by a damped step toward the midpoint of the two current group-average subsidy rates.
6. Update the measured product-subproblem and recovery times, adjust the timing-adaptive recovery interval when necessary, and preserve sufficient time for the protected final recovery.

Return: the highest-objective feasible policy verified under the original nonlinear model. The scalable product-subproblem routine is not globally certifying, so its Lagrangian value is reported only as a diagnostic.

Algorithm 2: Certified PD
Input: instance data; initial multipliers $\mathbf{y}^0$; certified product tolerance $\varepsilon_{\mathrm{prod}}$; dual stopping tolerance $\varepsilon_{\mathrm{dual}}$; time budget T.
Initialize: set $U_{\mathrm{PD}} = +\infty$ and the best verified feasible policy to empty.
Repeat for $t = 0, 1, \ldots$ **until certification, the iteration limit, or** T**:**

1. Enumerate the required corners of each connected component of the threshold-incidence graph from Theorem 3. If no group-order constraint is active, evaluate one separator-free dual problem.
2. At every required corner and fixed $\mathbf{y}^t$, solve every product subproblem with the anchor branch-and-bound procedure of Theorem 2. Continue until each returned product upper bound is within $\varepsilon_{\mathrm{prod}}$ of its product incumbent.
3. Sum the constant Lagrangian terms and certified product upper bounds. Maximize over the required component corners as in Theorem 3, obtaining a certified iteration bound U^t, and set $U_{\mathrm{PD}} \leftarrow \min\{U_{\mathrm{PD}}, U^t\}$.
4. Assemble the product incumbents at the maximizing corner, apply the same nonlinear-evaluator-checked primal recovery as in Algorithm 1, and update the best feasible value L_{PD} and policy when improved.
5. Use the constraint residuals of the product incumbents at the selected corner to perform the projected multiplier update $\mathbf{y}^{t+1} = [\mathbf{y}^t + \alpha_t \mathbf{r}^t]_+$.
6. If both bounds exist and $U_{\mathrm{PD}} - L_{\mathrm{PD}} \le \varepsilon_{\mathrm{dual}}$, terminate with a certified gap.

Return: the best verified feasible policy, its value L_{PD}, and U_{PD}. The upper bound is released only when all required product solves and separator-corner evaluations have completed with certification.

Multiplier-update residuals. The multiplier update uses signed residuals matched to the Lagrangian signs, which are distinct from the original group-ratio residual used by primal recovery. Before scale normalization, the components are

$$\begin{aligned}
\widetilde{r}_{\lambda_0} &= D_0 - B_0, & \widetilde{r}_{\lambda_r} &= D_r - B_r, \\
\widetilde{r}_{\mu^+} &= H_0 - \overline{\rho} Q_0, & \widetilde{r}_{\mu^-} &= \underline{\rho} Q_0 - H_0, \\
\widetilde{r}_{\alpha_e} &= \left(\theta_e + \frac{m_e}{2}\right) Q_{A_e} - H_{A_e} = -h_e^A, & \widetilde{r}_{\beta_e} &= H_{B_e} - \left(\theta_e - \frac{m_e}{2}\right) Q_{B_e} = -h_e^B.
\end{aligned} \tag{18}$$

Thus a positive component indicates violation of the corresponding dualized constraint. Budget components are normalized by their budget scales and the average-band and lifted group components by the corresponding GMV scales before forming $\mathbf{r}^t$ in Algorithm 1. In particular, $r_{\alpha_e} = \theta_e + m_e/2 - \overline{s}_{A_e}$ and $r_{\beta_e} = \overline{s}_{B_e} - \theta_e + m_e/2$ after division by the positive group GMVs.

Cross-market range constraints in scalable product subproblems. For a product i subject to $\max_{r \in \mathcal{R}_i} s_{ir} - \min_{r \in \mathcal{R}_i} s_{ir} \le \delta_i$, the scalable routine uses an anchor z_i and restricts every market to

$$s_{ir} \in \left[\max\{\underline{s}_{ir}, z_i\}, \min\{\overline{s}_{ir}, z_i + \delta_i\}\right].$$

Hence every assignment generated for a fixed anchor satisfies the range constraint by construction. The routine first solves the market-level problems separately for each market of product i and accepts the resulting assignments if their subsidy range is already at most δ_i. Otherwise it evaluates a finite anchor set containing feasible anchor-interval endpoints and midpoints, coupon-boundary locations and their δ_i-shifted counterparts, structurally relevant stationary-point locations, the anchor retained from the preceding dual iteration, and midpoints between consecutive structural anchors. For each tested anchor, the market problems separate again over the restricted intervals, which are split at coupon thresholds and optimized using endpoints and feasible stationary points. The highest-value tested assignment is returned. Thus the scalable routine preserves product-local and range feasibility; its approximation lies only in searching a finite anchor set rather than globally optimizing over the continuous anchor interval, so it does not produce a certified product upper bound.

Damped separator update in scalable PD. For a group-order relation e requiring $\overline{s}_{A_e} \ge \overline{s}_{B_e} + m_e$, the lifted formulation uses an auxiliary separator θ_e to impose $\overline{s}_{A_e} \ge \theta_e + m_e/2$ and $\overline{s}_{B_e} \le \theta_e - m_e/2$. Let $\overline{s}^t_{A_e}$ and $\overline{s}^t_{B_e}$ be the two GMV-weighted average subsidy rates under the raw policy produced at scalable PD iteration t. Their midpoint defines the target

$$\theta^t_{e,\mathrm{target}} = \frac{\overline{s}^t_{A_e} + \overline{s}^t_{B_e}}{2}. \tag{19}$$

Rather than replacing the separator immediately, scalable PD takes the damped and projected update

$$\theta_e^{t+1} = \Pi_{[\theta_e^L, \theta_e^U]} \left((1 - \gamma_\theta)\theta_e^t + \gamma_\theta \theta_{e,\text{target}}^t \right), \qquad 0 < \gamma_\theta \leq 1, \tag{20}$$

where $\Pi_{[\theta_e^L, \theta_e^U]}$ projects onto the prescribed admissible separator interval. Damping limits oscillation when consecutive raw policies place the two group averages on different sides of the current separator. This midpoint update is a scalable heuristic and does not itself provide a dual certificate; certified PD instead maximizes the dual value over the required separator corners in Algorithm 2.

B.1 Residuals and locally feasible candidate assignments

Recovery associates a signed residual with every enabled aggregate constraint; a positive residual indicates a violation. Before normalization, the resource residuals are

$$D_0(\mathbf{s}) - B_0, \qquad D_r(\mathbf{s}) - B_r, \qquad H_0(\mathbf{s}) - \overline{\rho} Q_0(\mathbf{s}), \qquad \underline{\rho} Q_0(\mathbf{s}) - H_0(\mathbf{s}), \tag{21}$$

and the residual of group-order edge e is

$$\overline{s}_{B_e}(\mathbf{s}) + m_e - \overline{s}_{A_e}(\mathbf{s}). \tag{22}$$

Budget residuals are divided by the corresponding budget, average-band residuals by total GMV, and group residuals are already rates. This produces the normalized residual $r_k(\mathbf{s})$ used below.

Before a complete product assignment enters a recovery master, it is checked against the product's subsidy bounds, coupon rules, and cross-market range constraint. A *downward assignment* lowers one product's subsidies to release global or market budget or to reduce an average-subsidy upper-bound violation. An *upward assignment* raises them to address an average-subsidy lower-bound violation. When a product has neither coupon thresholds nor additional effective-subsidy limits, a common-shift assignment subtracts the same amount d from every active-market subsidy, $s_{ir} \leftarrow s_{ir} - d$, and therefore preserves all within-product subsidy differences. In the general case, an endpoint assignment chooses, for every active market, a subsidy at or toward the endpoint of one feasible fixed-coupon regime. Different markets may move by different amounts, but the complete assignment is retained only after all product-local constraints have been checked.

Recovery can also pair an increase in a higher-priority group with a budget-releasing decrease elsewhere. In addition, it stores complete locally feasible assignments produced at selected earlier multiplier iterations. These earlier-iterate assignments are retained in a bounded archive, including representatives with the best objective, total normalized violation, group-order violation, and resource violation. This preserves useful alternatives when several consecutive multiplier iterations produce nearly identical assignments.

Average-floor cleanup. For a remaining average-floor violation on an instance without group-order constraints, a streaming pass scans all products. For each product i, recovery constructs an upward candidate assignment

$$\mathbf{a}_i^{\text{up}} = \left(s_{ir} + d_i\right)_{r \in \mathcal{R}_i},$$

where $\mathcal{R}_i$ is the set of markets in which product i is active and $d_i \geq 0$ is the largest common increase that keeps every updated subsidy rate within its current fixed-coupon regime and satisfies all product-local bounds and the cross-market range constraint. Let $\mathbf{s}^{(i)}$ denote the policy obtained by replacing product i's current assignment with $\mathbf{a}_i^{\text{up}}$, and define the average-floor violation by

$$v_{\text{floor}}(\mathbf{s}) = \left[\underline{\rho} Q_0(\mathbf{s}) - H_0(\mathbf{s})\right]_+.$$

For every candidate satisfying

$$\Delta_i^{\text{floor}} = v_{\text{floor}}(\mathbf{s}) - v_{\text{floor}}\left(\mathbf{s}^{(i)}\right) > 0,$$

recovery computes its GMV loss

$$\Delta_i^F = F(\mathbf{s}) - F\left(\mathbf{s}^{(i)}\right)$$

and ranks the candidates in increasing order of

$$\kappa_i = \frac{\Delta_i^F}{\Delta_i^{\text{floor}}}.$$

Thus, assignments that remove more average-floor violation per unit of GMV sacrificed are considered first; a negative value of κ_i indicates that the assignment both reduces the violation and improves GMV. Recovery applies the ranked assignments until the average floor becomes feasible. If the final complete assignment would make a resource constraint binding, or if only part of it is needed to eliminate the remaining violation, bisection selects a smaller common shift $d_i' \in [0, d_i]$. Every accepted assignment is evaluated using the original nonlinear demand, GMV, and constraint expressions.

When group-order constraints are active, an unrestricted full-product scan could reopen many ordering relations by changing the group averages independently. Recovery therefore uses a bounded candidate pool constructed from products in, or coupled to, the currently affected group components. Upward assignments are prioritized when they reduce the average-floor residual without increasing any group-order residual beyond tolerance. Candidate assignments are applied with backtracking and exact nonlinear evaluation. If no individual assignment is acceptable, the remaining average-floor residual is passed to the continuous recovery master, which can combine an upward assignment with budget-releasing or group-balancing assignments.

B.2 Target group-average projection

For the group-order constraints considered in this work, each product is associated with at most one designated primary merchandise group. Let $\mathcal{G}_{\text{all}}$ contain all designated primary groups together with an unconstrained neutral group for products to which no such business group applies, and let $\mathcal{G} \subseteq \mathcal{G}_{\text{all}}$ contain the groups appearing in at least one declared group-order relation. At the beginning of each group-order projection round, recovery evaluates the current policy $\mathbf{s}^{\text{cur}}$ and, for every $g \in \mathcal{G}_{\text{all}}$, computes

$$Q_g^{\text{cur}} = \sum_{i \in g} \sum_{r \in \mathcal{R}_i} G_{ir}(s_{ir}^{\text{cur}}), \qquad H_g^{\text{cur}} = \sum_{i \in g} \sum_{r \in \mathcal{R}_i} s_{ir}^{\text{cur}} G_{ir}(s_{ir}^{\text{cur}}), \qquad \overline{s}_g^{\text{cur}} = \frac{H_g^{\text{cur}}}{Q_g^{\text{cur}}}.$$

Only groups in $\mathcal{G}$ are order-constrained and receive target averages in the projection. Products outside these groups remain subject to all product-local and resource constraints and continue to contribute to campaign-wide aggregates. Define

$$\mathcal{I}_g = \{i \in \mathcal{I} : g(i) = g\}, \qquad g \in \mathcal{G}_{\text{all}}.$$

The sets $\{\mathcal{I}_g : g \in \mathcal{G}_{\text{all}}\}$ form a partition of the promotion pool. Therefore, under the current policy $\mathbf{s}^{\text{cur}}$, the campaign-wide quantities decompose as

$$Q_0^{\text{cur}} = \sum_{g \in \mathcal{G}_{\text{all}}} Q_g^{\text{cur}}, \qquad H_0^{\text{cur}} = \sum_{g \in \mathcal{G}_{\text{all}}} H_g^{\text{cur}}, \qquad \overline{s}_0^{\text{cur}} = \frac{H_0^{\text{cur}}}{Q_0^{\text{cur}}}.$$

Define

$$H_{\text{out}}^{\text{cur}} = H_0^{\text{cur}} - \sum_{g \in \mathcal{G}} H_g^{\text{cur}}.$$

Here $H_{\text{out}}^{\text{cur}}$ denotes the current subsidy-weighted GMV contributed by products that do not belong to any group included in the target-average projection. This term is zero when the projected groups cover the complete promotion pool. All of these quantities are fixed constants within the following projection problem.

Recovery chooses the reference campaign-wide average

$$\rho^{\text{ref}} = \begin{cases} (\underline{\rho} + \overline{\rho})/2, & \text{if both average bounds are active,} \\ \min\{\overline{s}_0^{\text{cur}}, \overline{\rho}\}, & \text{if only the upper bound is active,} \\ \max\{\overline{s}_0^{\text{cur}}, \underline{\rho}\}, & \text{if only the lower bound is active,} \\ \overline{s}_0^{\text{cur}}, & \text{otherwise.} \end{cases}$$

It then computes target group-average subsidy rates $\mathbf{t} = (t_g : g \in \mathcal{G})$ that satisfy the ordering margins while remaining close to the current group averages. To express the absolute deviations linearly, introduce $u_g^+, u_g^- \geq 0$ such that

$$t_g = \overline{s}_g^{\text{cur}} + u_g^+ - u_g^-.$$

The projection problem is

$$\begin{aligned}
\min_{\mathbf{t},\mathbf{u}^+,\mathbf{u}^-} \quad & \sum_{g\in\mathcal{G}} \frac{Q_g^{\mathrm{cur}}}{Q_0^{\mathrm{cur}}}(u_g^+ + u_g^-) && && (23)\\
\text{s.t.} \quad & t_{A_e} - t_{B_e} \geq m_e, && e=(A_e,B_e,m_e)\in\mathcal{E},\\
& t_g = \bar{s}_g^{\mathrm{cur}} + u_g^+ - u_g^-, && g\in\mathcal{G},\\
& \sum_{g\in\mathcal{G}} Q_g^{\mathrm{cur}} t_g = \rho^{\mathrm{ref}} Q_0^{\mathrm{cur}} - H_{\mathrm{out}}^{\mathrm{cur}}, &&\\
& \underline{t}_g \leq t_g \leq \bar{t}_g, && g\in\mathcal{G},\\
& u_g^+ \geq 0, \qquad u_g^- \geq 0, && g\in\mathcal{G}.
\end{aligned}$$

At an optimum, $u_g^+ + u_g^- = |t_g - \bar{s}_g^{\mathrm{cur}}|$. The objective therefore finds order-feasible targets with the smallest GMV-weighted absolute deviation from the current group averages. The equality involving ρ^{ref} preserves the selected campaign-wide average under the current GMV weights, while the contributions of products that do not belong to any group are held fixed.

The bounds $\underline{t}_g$ and $\bar{t}_g$ are obtained from the smallest local subsidy lower bound and the largest local subsidy upper bound, respectively, among products in group g. The resulting target shift

$$\Delta_g^{\mathrm{target}} = t_g - \bar{s}_g^{\mathrm{cur}}$$

guides the construction of upward assignments when $\Delta_g^{\mathrm{target}} > 0$ and downward assignments when $\Delta_g^{\mathrm{target}} < 0$.

Because changing subsidies also changes demand and GMV, the projection does not treat $\mathbf{t}$ as an exact prediction of the realized group averages. Recovery tests step fractions $1, 1/2, 1/4, \ldots$ toward the targets and recomputes the original nonlinear quantities after every trial. A trial is accepted only if this exact evaluation reduces both the maximum group-order violation and the overall normalized feasibility violation.

B.3 Component multiple-choice master and large blocks

For a group-order component c that contains at least one violated ordering relation, let $\mathcal{I}_c$ be its products, let $\mathcal{C}_c$ contain its group-order constraints together with all enabled resource constraints, and let $\mathcal{A}(i)$ be the candidate replacements retained for product i. Selecting no replacement leaves that product unchanged. Because group-ratio residual changes from simultaneous replacements are not additive, the quantities below form a linear prediction model rather than an exact reformulation. The restricted component master is

$$\begin{aligned}
\min_{z,\eta} \quad & \eta + \epsilon_{\mathrm{mip}} \sum_{i\in\mathcal{I}_c} \sum_{a\in\mathcal{A}(i)} \ell_{ia} z_{ia} && && (24)\\
\text{s.t.} \quad & r_k(\mathbf{s}) + \sum_{i\in\mathcal{I}_c} \sum_{a\in\mathcal{A}(i)} \Delta_{k,ia} z_{ia} \leq \eta, && k\in\mathcal{C}_c,\\
& r_k(\mathbf{s}) + \sum_{i\in\mathcal{I}_c} \sum_{a\in\mathcal{A}(i)} \Delta_{k,ia} z_{ia} \leq \max\{r_k(\mathbf{s}), \tau_k\}, && k\in\mathcal{C}_c,\\
& \sum_{a\in\mathcal{A}(i)} z_{ia} \leq 1, && i\in\mathcal{I}_c,\\
& z_{ia}\in\{0,1\}, \qquad \eta \geq 0. &&
\end{aligned}$$

Here ℓ_{ia} and $\Delta_{k,ia}$ are respectively the predicted GMV loss and residual change when replacement $a \in \mathcal{A}(i)$ is selected for product i. The first constraint, together with minimizing η, reduces the largest predicted normalized violation over all constraints included in the component master. The second is a constraint-wise no-worsening safeguard: it keeps a currently feasible constraint within tolerance τ_k and prevents a currently violated constraint from becoming more violated. The small coefficient ϵ_{mip} in the objective uses predicted GMV loss only to break ties after reducing the largest violation. The binary variable z_{ia} selects a complete candidate assignment rather than interpolating between two assignments. This matters when the assignments lie in different coupon regimes: an interpolated subsidy can activate a different coupon tier, so its GMV and resource usage need not

interpolate between the two stored values. The predicted group-order residuals coordinate these choices across products, while restricting the MIP to one connected component limits the combinatorial search. Every selected combination is subsequently checked by the original nonlinear evaluator.

The implementation limits a component master to 192 products when coupon tiers are present and 256 products otherwise. A larger component is covered by overlapping blocks centered on its violated ordering relations. Let M be the maximum number of products allowed in one component master. For each violated relation $A > B$, an edge-centered block contains at most $\lfloor M/2 \rfloor$ products from A, ranked by their GMV-weighted capacity to increase subsidy, and at most $\lfloor M/2 \rfloor$ products from B, ranked by their GMV-weighted capacity to decrease subsidy. Thus each block contains at most M products; it may contain fewer when either side has fewer eligible products. Consecutive blocks retain up to eight high-impact products from the preceding block and fill the remaining positions with new products. This overlap lets an adjustment propagate through relations such as $A > B$ and $B > C$. Blocks from different edges and disconnected components are interleaved so that one large component does not consume the entire recovery budget. Every proposed block solution is evaluated against all original constraints and rejected if it worsens another enabled violation beyond tolerance.

B.4 Two-phase continuous master

Each retained action $a \in \mathcal{A}$ defines a direction $\mathbf{d}^a$ from the current policy $\mathbf{s}^0$ toward one or more precomputed locally feasible target assignments. An action may modify one product, a bundle of products, or a coordinated upward–downward product pair. The LP variable $x_a \in [0, 1]$ is the proposed step along that direction, so the policy predicted by the master is

$$\widehat{\mathbf{s}}(\mathbf{x}) = \mathbf{s}^0 + \sum_{a \in \mathcal{A}} x_a \mathbf{d}^a.$$

Let Δ_{ka} be the exactly evaluated change in normalized residual k produced by applying the full action a to $\mathbf{s}^0$, and let ℓ_a be its GMV loss (negative when the full action improves GMV). Let $\mathcal{C}_{\text{agg}}$ index the enabled aggregate constraints and let $\mathcal{A}(i)$ contain the actions that modify product i. Phase I solves

$$\begin{aligned}
\min_{x,\eta} \quad & \eta + \epsilon_{\text{obj}} \sum_{a \in \mathcal{A}} \ell_a x_a && && (25)\\
\text{s.t.} \quad & r_k(\mathbf{s}^0) + \sum_{a \in \mathcal{A}} \Delta_{ka} x_a \le \eta, && k \in \mathcal{C}_{\text{agg}},\\
& r_k(\mathbf{s}^0) + \sum_{a \in \mathcal{A}} \Delta_{ka} x_a \le \max\{r_k(\mathbf{s}^0), \tau_k\}, && k \in \mathcal{C}_{\text{agg}},\\
& \sum_{a \in \mathcal{A}(i)} x_a \le 1, && i \in \mathcal{I},\\
& 0 \le x_a \le 1, \qquad \eta \ge 0.
\end{aligned}$$

The constraint $\sum_{a \in \mathcal{A}(i)} x_a \le 1$ limits the total interpolation weight assigned to actions involving product i; equivalently, the current product assignment retains the remaining weight. A paired or bundled action uses the same x_a for all constituent changes, preserving their intended coordination. The master's linear prediction of the normalized residual of constraint k under action weights $\mathbf{x}$ is

$$\widehat{r}_k(\mathbf{x}) = r_k(\mathbf{s}^0) + \sum_{a \in \mathcal{A}} \Delta_{ka} x_a.$$

Because demand, coupon tiers, and group-average constraints are nonlinear, this secant model need not equal the true residual of a fractional or combined move. The variable η bounds the largest predicted residual, τ_k is the feasibility tolerance, and ϵ_{obj} breaks ties in favor of lower GMV loss. The no-worsening constraints allow a currently feasible constraint to move only up to its tolerance and prohibit a violated constraint from becoming more violated. If Phase I predicts that all residuals

can reach tolerance, Phase II solves

$$\begin{aligned} \min_{x} \quad & \sum_{a\in\mathcal{A}} \ell_a x_a && && (26)\\ \text{s.t.} \quad & r_k(\mathbf{s}^0) + \sum_{a\in\mathcal{A}} \Delta_{ka} x_a \le \tau_k, && k \in \mathcal{C}_{\text{agg}}, \\ & \sum_{a\in\mathcal{A}(i)} x_a \le 1, && i \in \mathcal{I}, \\ & 0 \le x_a \le 1. \end{aligned}$$

Phase II therefore minimizes predicted GMV loss subject to predicted feasibility. For instances without group-order constraints, the action pool still includes complete locally feasible product assignments that move a product's active-market subsidies toward lower or upper endpoints. This allows the continuous master to coordinate resource constraints directly. For example, it can combine a downward assignment that releases budget with an upward assignment that repairs an average-subsidy lower-bound violation. Neither assignment may be feasible or useful alone, whereas their joint effect can satisfy both constraints. In contrast, separate greedy passes may repeatedly reopen the constraint repaired by the preceding pass. To remain scalable, the master uses a bounded action pool and a sparse constraint matrix.

The LP solution is used only to propose a direction. Before updating the recovery state, the implementation forms

$$\mathbf{s}^{\text{trial}} = \mathbf{s}^0 + \gamma \sum_{a\in\mathcal{A}} x_a \mathbf{d}^a, \qquad \gamma \in (0, 1],$$

and recomputes the original nonlinear objective and every constraint residual. Backtracking tests step sizes $\gamma = 1, 1/2, 1/4, \ldots$ and accepts the largest trial that preserves all product-local and range constraints. The accepted trial must also satisfy the aggregate no-worsening limits and strictly reduce the maximum normalized violation. Otherwise the LP proposal is rejected.

B.5 Deterministic resource cleanup

After the joint recovery master, PD applies residual-specific deterministic cleanup passes. For a violated market budget, recovery first constructs legal single-market downward assignments and ranks them by GMV loss per unit of budget released. It applies the assignments in this order until the residual falls within tolerance. It then considers whole-product common-shift assignments, which lower all active-market subsidy rates of one product by a common amount and therefore preserve their pairwise differences. For the global budget, the same two assignment classes are pooled across markets and ranked by GMV loss per unit of total budget released.

To repair the upper bound on the GMV-weighted average subsidy, downward assignments are instead ranked by GMV loss per unit decrease in

$$H_0(\mathbf{s}) - \overline{\rho} Q_0(\mathbf{s}).$$

For each residual, if the final full assignment would reduce the constrained quantity more than necessary, bisection shortens that assignment to the smallest tested change that closes the residual.

Every candidate assignment must satisfy the product-local subsidy bounds, respect the coupon thresholds, and preserve the cross-market range constraint. For a product satisfying

$$\max_{r\in\mathcal{R}_i} s_{ir} - \min_{r\in\mathcal{R}_i} s_{ir} \le \delta_i,$$

the smallest value to which the subsidy at market r can currently be reduced while the other market subsidies remain fixed is

$$L_{ir}^{\text{range}} = \max\left\{ \underline{s}_{ir}, \max_{r'\in\mathcal{R}_i\setminus\{r\}} s_{ir'} - \delta_i \right\},$$

where $\underline{s}_{ir}$ is the local subsidy lower bound. If the ordinary ranked cleanup assignments cannot eliminate the remaining resource violation, the range-aware fallback lowers selected market subsidies toward these currently admissible values. It repeats the scan because lowering one market can reduce the range-induced lower bound at another market and thereby permit an additional reduction in a later pass. After each accepted update, recovery recomputes the resulting demand, GMV, and

resource residuals using the original nonlinear functions. Because downward resource cleanup may reopen a group-order constraint or the lower bound on the GMV-weighted average subsidy, the subsequent recovery stages re-evaluate these residuals and invoke the corresponding group-aware or average-floor cleanup only when needed.

B.6 Recovery schedule, exact acceptance, and preservation of the best feasible policy

When group-order constraints are present, recovery first computes target group averages that satisfy the ordering margins and generates product assignments that move the current averages toward those targets. It then applies the component multiple-choice masters, the continuous master, and deterministic cleanup of market budgets, the global budget, and the average-subsidy upper bound. A second component pass repairs any ordering damage caused by resource cleanup. If violations remain and recent progress is weak, recovery enlarges the candidate-assignment pools and repeats these steps. When all resource constraints are feasible and only a few small ordering violations remain in one component, a final cascade lowers selected products on the lower-priority sides of the violated relations. Backtracking retains a cascade move only if exact evaluation improves feasibility. For instances without group-order constraints, the full-scan average-floor procedure is followed, when needed, by the resource-only continuous master.

The default candidate pool contains at most 800 replacements and may expand to 1600; each component MIP has its own time limit. Because the predicted residual changes can be inaccurate for nonlinear fractional moves, a master solution is never accepted from its linear approximation alone. The workflow tests step lengths $1, 1/2, 1/4, \ldots$ with the original nonlinear evaluator and accepts only a step that satisfies all product-local constraints, does not worsen any aggregate violation beyond tolerance, and strictly improves the current feasibility measure. After every accepted stage, the implementation safely records the best evaluator-verified feasible solution found so far, so a later timeout or unsuccessful repair cannot erase it.

B.7 Feasible polishing and time allocation

Monotone feasible polishing. Ordinary periodic recovery omits feasible polishing. During final recovery, PD first applies the feasibility-recovery modules. Once they produce a feasible policy, PD records it as the best feasible incumbent found so far and uses the remaining recovery time for feasible polishing. A polished candidate replaces the stored incumbent only if evaluation under the original nonlinear model confirms that it satisfies all constraints and has a strictly higher objective value. The timing rule that starts final recovery is described below.

The first polishing module constructs short single-market moves. For an active market r of product i, an upward move increases s_{ir} toward its largest locally admissible value and is retained only if it improves GMV. A downward move decreases s_{ir} toward its smallest locally admissible value and is retained when it releases budget or creates slack in another active resource constraint. Both values account for the product-local bounds, coupon regimes, and cross-market range constraint. An upward move may be accepted alone when the incumbent has sufficient resource slack. Otherwise, polishing pairs it with a downward donor move whose released resource permits the upward move. Candidate pairs are ranked by their predicted net GMV gain. If the full upward move would violate a resource constraint, bisection finds the largest feasible fraction of that move. A single move or paired move is committed only when evaluation under the original nonlinear model confirms feasibility and a strict improvement in F.

For instances without group-order constraints, polishing also uses a binary restricted master over complete product assignments. For each product i, let $\mathcal{A}_i^{\mathrm{pol}}$ contain selected complete assignments from earlier dual iterations and locally valid assignments toward subsidy or fixed-coupon-regime endpoints. The bounded candidate pool retains both objective-improving assignments and donor assignments that sacrifice some GMV but create useful resource slack. For $a \in \mathcal{A}_i^{\mathrm{pol}}$, let ΔF_{ia} be its change in objective relative to the incumbent and let $\Delta r_{k,ia}$ be its exact change in the additive

residual of resource constraint k. The polishing master is

$$\begin{aligned} \max_{y} \quad & \sum_{i} \sum_{a \in \mathcal{A}_i^{\mathrm{pol}}} \Delta F_{ia} y_{ia} && (27) \\ \text{s.t.} \quad & r_k(\mathbf{s}) + \sum_{i} \sum_{a \in \mathcal{A}_i^{\mathrm{pol}}} \Delta r_{k,ia} y_{ia} \le \tau_k, && k \in \mathcal{C}_{\mathrm{res}}, \\ & \sum_{a \in \mathcal{A}_i^{\mathrm{pol}}} y_{ia} \le 1, && i \in \mathcal{I}, \\ & y_{ia} \in \{0,1\}. \end{aligned}$$

Here $\mathcal{C}_{\mathrm{res}}$ contains the active budget and average-subsidy constraints, and τ_k is the feasibility tolerance for constraint k. The master can therefore combine a profitable assignment that consumes resource slack with one or more donor assignments that create the required slack. The selected combination is accepted only if recomputation under the original nonlinear model confirms feasibility and a strict objective improvement.

For a feasible group-order component, polishing instead uses a component-level multiple-choice master. Consider an ordering relation $A_e > B_e$ with margin m_e. Because the incumbent is feasible, the interval

$$\left[\overline{s}_{B_e}(\mathbf{s}) + \frac{m_e}{2},\ \overline{s}_{A_e}(\mathbf{s}) - \frac{m_e}{2}\right]$$

is nonempty. Polishing fixes a separator θ_e at the midpoint of this interval and requires the selected assignments to satisfy

$$H_{A_e}(\mathbf{s}') - \left(\theta_e + \frac{m_e}{2}\right) Q_{A_e}(\mathbf{s}') \ge 0, \tag{28}$$

$$H_{B_e}(\mathbf{s}') - \left(\theta_e - \frac{m_e}{2}\right) Q_{B_e}(\mathbf{s}') \le 0. \tag{29}$$

These two additive inequalities imply $\overline{s}_{A_e}(\mathbf{s}') \ge \overline{s}_{B_e}(\mathbf{s}') + m_e$. The component master selects at most one complete endpoint or earlier-iterate assignment for each participating product, maximizes the predicted GMV gain, and enforces these fixed-separator inequalities together with all active resource constraints. Although the fixed-separator conditions are sufficient for preserving the original ordering relation, every selected combination is again accepted only after exact evaluation of the original group-average ratios and all other constraints.

Time allocation between dual search and recovery. Recovery is not invoked after every PD iteration. It is normally called after every q completed dual iterations. A recovery call may be brought forward in two cases. First, if the best primal lower bound has not improved by a prescribed amount over several iterations, recovery tests whether the current dual assignments can be recombined into a better feasible policy. Second, using the recently observed iteration times, the workflow checks whether waiting another q iterations would consume the time protected for final recovery. If so, it performs an intermediate recovery call before entering that protected time window. These periodic calls focus on finding or improving feasibility and omit the more expensive feasible-polishing modules.

Before observing any recovery call, the workflow sets an initial reserve R_0 using a bounded estimate of the recovery work implied by the number of active product–market pairs and the enabled recovery modules. This initial reserve is used only as a safeguard; it is capped at a fixed fraction of the total time budget and is replaced by a measurement-based estimate once periodic recovery has been observed. After at least one recovery call has completed, the protected final-recovery allowance is updated as

$$R_{\mathrm{final}} = \max\left\{R_0,\ \beta\, c_{\mathrm{coord}} \widehat{t}_{\mathrm{rec}} + R_{\mathrm{pol}}\right\},$$

where $\widehat{t}_{\mathrm{rec}}$ is a conservative estimate obtained from completed recovery calls on the current instance, $\beta > 1$ is a safety factor, $c_{\mathrm{coord}} > 1$ allows an additional coordination sweep when a group-coupled policy remains infeasible, and R_{pol} reserves bounded time for final feasible polishing. Thus the reserve is based primarily on recovery time measured on the current machine and instance, rather than being selected directly from the instance size or family.

When recovery is much more expensive than one dual iteration, its periodic cadence is also adjusted using the observed iteration and recovery durations. For a target recovery-time fraction ϕ, the

timing calculation uses $q_{\text{time}} = \left\lceil \frac{\widehat{t}_{\text{rec}}(1-\phi)}{\phi\,\widehat{t}_{\text{iter}}} \right\rceil$ as a guide for the number of dual iterations between recovery calls. For a group-coupled instance that remains infeasible after at least two completed recovery calls, the workflow may increase the number of dual iterations between periodic recovery calls. Let q_0 be the default number of iterations between calls, and let $q_{\max} = \left\lfloor \frac{G_{\max}}{\widehat{t}_{\text{iter}}} \right\rfloor$ be the largest interval consistent with the maximum allowed dual-search gap $G_{\max}$. The timing-based interval q_{time} is adopted only when it is at least three times q_0; otherwise, the workflow retains q_0 to avoid reacting to ordinary timing variation. When adaptation is triggered, the interval is set to $q = \max\{q_0,\, \min\{q_{\text{time}}, q_{\max}\}\}$. Once a feasible recovery state has been found, the workflow returns to the default cadence q_0 so that subsequent feasible solutions can be updated more frequently.

Under a wall-clock limit, the nominal iteration count is therefore not a hard early stopping rule. PD continues launching complete dual iterations only while the estimated cost of another iteration fits before the protected final recovery window. Once the remaining time reaches R_{final}, the workflow stops launching new dual iterations and invokes final recovery, starting from the best feasible incumbent when one exists or otherwise from the best previously completed recovery state. Feasible polishing is enabled during this final reserved call. After every completed recovery or polishing stage, the full policy is re-evaluated under the original nonlinear model, and the stored incumbent is updated only when the new policy is feasible and has a strictly higher objective. Consequently, a timeout or an unsuccessful later stage cannot erase the best verified feasible policy found earlier.

C Structure-Aware LNS+SA Details

Algorithm 3: Structure-aware adaptive LNS+SA

Input: instance data; random seed; time budget $T_{\max}$; destroy and repair operator sets; per-product candidate-menu limit.

Initialize: Construct up to five structure-specific starting policies, select the starting policy according to the active constraints and feasibility, and separately retain the best verified feasible policy. Initialize the operator and violation weights and temperature, and set the number of consecutive non-improving iterations to zero, as detailed in Appendices C.1 and C.6.

Repeat until $T_{\max}$ or the iteration limit:

1. Sample an applicable destroy operator from its adaptive probability distribution, as described in Appendix C.2.
2. Construct its product neighborhood, enlarging the neighborhood after search stalls and retaining complete group-order blocks when applicable; see Appendix C.2.
3. Generate a menu of locally valid complete multi-market assignments for each destroyed product, as detailed in Appendix C.1.
4. Select a repair operator using its adaptive probability distribution. For a complete group-order neighborhood that fits within the restricted-master size limit, bypass adaptive sampling. Select `group_master` operator during the regular search and switch to `group_master_refined` operator during the final portion of the time budget or after the configured stall threshold. See Appendix C.3 for details.
5. Apply the selected repair operator to construct a candidate replacement for the destroyed products. If it returns no candidate replacement, apply `greedy` operator as a backup.
6. Recompute the candidate objective and all constraint residuals from the selected assignments, and perform a full exact check before retaining a new best feasible policy; see Appendix C.5.
7. When a feasible incumbent exists, first apply the feasibility-corridor test. Accept an admissible proposal if $\Delta E \le 0$; if $\Delta E > 0$ and $f < 1 - \phi$, accept it with the SA probability defined in Appendix C.6. Otherwise, reject it.
8. Update the selected destroy and repair operator scores using the improvement obtained per unit running time. After acceptance, also update the constraint-violation weights. See details in Appendices C.6 and C.7.
9. If the accepted candidate is exactly feasible and improves the stored feasible objective, replace the best feasible incumbent. This incumbent is kept separately from the possibly infeasible current search state; see Appendix C.5.
10. Cool the temperature. Reheat after a shorter stall, and restart from a deferred initial policy or the best stored state after a longer stall; see Appendix C.6.

Return: the best evaluator-verified feasible incumbent, if one was found; otherwise the least-violating state for diagnostics.

C.1 INITIAL POLICIES AND CANDIDATE-ASSIGNMENT MENUS

The five base initializations independently choose, for each product, the locally feasible assignment with maximum GMV, minimum budget cost, minimum mean subsidy, maximum mean subsidy, or a group-order-oriented rate. The preferred initialization depends only on declared constraints: group order takes priority, followed by an average floor, a budget, an average cap, and finally the unconstrained objective. When the order graph contains several product-disjoint components, the initializer constructs a small menu of component-level policies, each assigning all products in one component while satisfying its internal group-order constraints. A multiple-choice restricted master selects exactly one policy per component to maximize total GMV subject to the shared budget and campaign-average constraints.The resulting policies initialize the global LNS; they do not decompose or restrict its subsequent search.

Each product menu contains its assignment in the current LNS state together with a small set of alternative complete multi-market assignments. For each active market, candidate subsidy values are taken from the product-local bounds, CODE-regime boundaries, and values giving low budget cost or high GMV. The generator also considers small common and single-market shifts around the current assignment. Candidate values from different markets are first assembled into a product-level target vector. This vector is then adjusted jointly to satisfy the subsidy bounds, CODE-regime intervals, and the cross-market range constraint; it is retained only if the resulting complete assignment satisfies every product-local constraint.

For a selected neighborhood, a multiple-choice restricted master chooses one candidate assignment for each neighborhood product while all other products remain fixed. It first seeks to reduce the relevant aggregate-constraint violations and, when feasibility is attainable, maximizes GMV. For group-component neighborhoods, the assignments selected by this coarse master may be refined by adding smaller common and single-market shifts around both the current assignment and the master-selected assignment. Both assignments remain in the refined menu, so the previous master solution remains available when the master is solved again.

To place objective changes and normalized constraint violations on comparable numerical scales, let $\{\mathbf{s}^{(m)}\}_m$ denote the constructed starting policies and define the reference objective scale

$$S_F = \max\left\{1, \max_m F(\mathbf{s}^{(m)})\right\}.$$

Subsequent neighborhood masters and the SA acceptance rule measure objective changes relative to S_F. If the search starts directly from a validated warm-start policy, we instead use $S_F = \max\{1, F(\mathbf{s}^{\mathrm{warm}})\}$.

C.2 DESTROY OPERATORS

The registered destroy operators are

$$\mathcal{O}_D = \{\texttt{random}, \texttt{pressure}, \texttt{related}, \texttt{stale}, \texttt{group}, \texttt{code}\}.$$

`random` samples products without structural information; `pressure` prioritizes products contributing most to violated budgets, average-subsidy bounds, or group-order constraints; `related` expands from a seed product toward products in the same merchandise group or with similar subsidy and objective scales; `stale` favors products not modified recently; `group` selects a pressured ordering relation and expands to its connected group component; and `code` targets products near coupon thresholds. The `group` operator is disabled without group-order constraints and `code` is disabled without multiple coupon tiers; the vanilla ablation uses only `random`.

The neighborhood initially contains a bounded number of products proportional to the square root of the instance size. Its size is increased geometrically after repeated iterations without an improved feasible or least-violating state, subject to the configured neighborhood cap. During the final portion of the time budget, the algorithm requires each general neighborhood to contain at least twice the initial neighborhood size, subject to the configured caps. This allows the repair master to coordinate larger multi-product changes when smaller neighborhoods can no longer improve the incumbent. A group destroy includes products from all merchandise groups in the selected connected order component when that component fits within the group-neighborhood cap. For a larger component, it constructs a balanced bounded block that retains products from each participating group, so the subsequent repair can coordinate both sides of the linked ordering relations.

C.3 REPAIR OPERATORS

The registered repair operators are

$$\mathcal{O}_R = \{\texttt{greedy}, \texttt{randomized_greedy}, \texttt{regret2}, \texttt{restricted_milp}, \texttt{group_master}, \texttt{group_master_refined}, \texttt{continuous_polish}\}.$$

The sequential repair operators `greedy`, `randomized_greedy`, and `regret2` compare candidate assignments using the same energy employed by the SA acceptance rule. The master-based repair operators instead use the restricted optimization formulations described below; their resulting proposals are subsequently evaluated by the common SA acceptance rule. Let $v_k(\mathbf{s})$ be the normalized violation of constraint k and define

$$V(\mathbf{s}) = \max_k v_k(\mathbf{s}) + 0.1 \sum_k w_k v_k(\mathbf{s}), \qquad \widehat{F}(\mathbf{s}) = \frac{F(\mathbf{s})}{S_F},$$

where w_k is the adaptive search weight assigned to the normalized violation of constraint k. Before the first feasible policy is found, the energy has the form

$$E_{\text{pre}}(\mathbf{s}) = V(\mathbf{s}) - \gamma(V(\mathbf{s}))\widehat{F}(\mathbf{s}),$$

where $\gamma(V)$ is small for a highly infeasible policy and increases as V approaches zero. Thus violation reduction has priority far from feasibility, while GMV becomes more influential near feasibility. After a feasible incumbent exists, admissible proposals are scored by

$$E_{\text{post}}(\mathbf{s}) = -\widehat{F}(\mathbf{s}) + 10V(\mathbf{s}),$$

together with the shrinking feasibility corridor defined in Appendix C.6.

The `greedy` repair visits the destroyed products in randomized order. For each product, it evaluates every candidate assignment conditional on the assignments already selected for earlier products and retains the candidate that gives the lowest resulting energy. If another selected repair operator returns no candidate replacement, the workflow invokes `greedy` as an inexpensive backup. The `randomized_greedy` variant instead samples from the three lowest-energy candidates with finite energy. The `regret2` operator also constructs the replacement sequentially, but it chooses which product to process next. For every unassigned product, it computes the energy difference between that product's best and second-best candidate assignments, then fixes the product with the largest difference to its best candidate. This prioritizes products for which losing the best available assignment would be most costly.

The `restricted_milp` operator jointly selects one candidate assignment for every destroyed product while all products outside the neighborhood remain fixed. When the current policy is infeasible, its Phase I master minimizes the maximum normalized constraint slack together with a weighted sum of the remaining slacks. If the resulting policy is verified feasible, Phase II removes the slacks and maximizes GMV over the same candidate menus. When the current policy is already feasible, the retained incumbent assignments provide a feasible master solution, so the operator directly maximizes GMV. Appendix C.4 gives the complete multiple-choice formulation.

When the destroy operator returns a complete connected block of group-order relations, all products from the involved groups are repaired jointly by `group_master`; a sequential repair could improve one ordering relation while worsening another. The `group_master_refined` variant first solves this master using the coarse candidate menus. It then identifies products whose selected assignments produce the most consequential changes, adds finer common and single-market shifts around both their current and master-selected assignments, and solves the master again. The optional `continuous_polish` operator is different from these discrete-menu repairs: it continuously adjusts a small number of subsidy variables near an already feasible policy, without crossing their current CODE-regime boundaries, and retains only an exactly verified feasible improvement.

In the current implementation, `regret2` is available only before the first feasible incumbent, outside final intensification, and for neighborhoods containing at most 16 products. The `restricted_milp` and group masters are limited to neighborhoods of at most 512 products. Final intensification disables `randomized_greedy` and `regret2`; a complete group block then uses `group_master_refined`, which is also activated after the configured sequence of non-improving iterations. The `continuous_polish` operator is available only when it is explicitly enabled, a feasible incumbent exists, and the search is in final intensification; it was disabled in the reported main experiments.

C.4 NEIGHBORHOOD MASTER AND CANDIDATE EXPANSION

For a destroyed neighborhood, products outside it contribute fixed constants to every aggregate constraint. The restricted MIP chooses exactly one retained assignment for each destroyed product. Its resource constraints use the exact precomputed contribution of each assignment. Group-average ratios use the fixed-separator sufficient conditions in equation 11; several separators anchored at the current midpoint and feasible interval boundaries are attempted when a single start fails, and subsequent rounds update the separator toward the aggregates selected by the preceding master.

Let $\mathcal{N}$ be the destroyed product set, let $\mathcal{V}_i$ be the candidate menu for product $i \in \mathcal{N}$, and let $y_{ia} = 1$ select assignment $a \in \mathcal{V}_i$. Let F_{ia} denote the GMV contribution of candidate assignment $a \in \mathcal{V}_i$. Each active resource or fixed-separator constraint is written in the common upper-bound form

$$C_k^{\text{out}} + \sum_{i \in \mathcal{N}} \sum_{a \in \mathcal{V}_i} c_{k,ia} y_{ia} \le b_k,$$

where C_k^{out} is the fixed contribution of products outside the neighborhood, $c_{k,ia}$ is the exact contribution of candidate assignment a, and lower-bound constraints have been multiplied by -1. With scale $\sigma_k > 0$, adaptive violation weight w_k, normalized slack ξ_k, and maximum slack ζ, Phase I solves

$$\begin{aligned}
\min_{y,\xi,\zeta} \quad & \zeta + 0.1 \sum_k w_k \xi_k - 10^{-3} \sum_{i \in \mathcal{N}} \sum_{a \in \mathcal{V}_i} \frac{F_{ia}}{S_F} y_{ia} && && (30)\\
\text{s.t.} \quad & \sum_{a \in \mathcal{V}_i} y_{ia} = 1, && i \in \mathcal{N}, \\
& C_k^{\text{out}} + \sum_{i \in \mathcal{N}} \sum_{a \in \mathcal{V}_i} c_{k,ia} y_{ia} \le b_k + \sigma_k \xi_k, && k \in \mathcal{C}_{\mathcal{N}}, \\
& 0 \le \xi_k \le \zeta, && k \in \mathcal{C}_{\mathcal{N}}, \\
& y_{ia} \in \{0,1\}, \qquad \zeta \ge 0.
\end{aligned}$$

The small objective term involving F_{ia} breaks ties in favor of higher-GMV assignments without overriding violation reduction. If the selected complete policy is feasible under the original nonlinear constraints, Phase II removes the slacks and solves

$$\begin{aligned}
\max_{y} \quad & \sum_{i \in \mathcal{N}} \sum_{a \in \mathcal{V}_i} F_{ia} y_{ia} && && (31)\\
\text{s.t.} \quad & \sum_{a \in \mathcal{V}_i} y_{ia} = 1, && i \in \mathcal{N}, \\
& C_k^{\text{out}} + \sum_{i \in \mathcal{N}} \sum_{a \in \mathcal{V}_i} c_{k,ia} y_{ia} \le b_k, && k \in \mathcal{C}_{\mathcal{N}}, \\
& y_{ia} \in \{0,1\}.
\end{aligned}$$

If the current state is already feasible, its assignment is present in every menu, so the no-slack master is feasible by construction and LNS proceeds directly to Phase II. A master solution is still a proposal: the original fractional group-order constraints and every other nonlinear expression are recomputed before acceptance.

After the restricted integer master has produced an exactly feasible replacement, the candidate menus may still restrict the attainable objective. To identify useful additional assignments, LNS solves the LP relaxation of the corresponding no-slack GMV-maximization master. Let $\pi_k \ge 0$ be the shadow price of shared master constraint k, and let $c_{k,ia}$ be the coefficient of candidate assignment a for product i in that constraint. The assignment is scored by

$$\text{score}_{ia} = \frac{F_{ia}}{S_F} - \sum_k \pi_k c_{k,ia}. \tag{32}$$

A high score indicates large GMV contribution after accounting for the assignment's use of scarce resources or tight group-order capacity. The pricing scores are used in two stages. First, for each neighborhood product i, the procedure computes

$$g_i = \max_{a \in \mathcal{V}_i} \text{score}_{ia} - \text{score}_{i,a_i^{\text{cur}}},$$

where a_i^{cur} is the product's assignment in the current LNS state. Only products with the largest values of g_i are selected for menu expansion. For each selected product, additional locally feasible assignments are generated around the assignment chosen by the current restricted master and evaluated using the same pricing score. A new assignment is added only if its score exceeds the best score already present in that product's menu. The procedure retains only a bounded number of the highest-scoring new assignments and then resolves the restricted integer master.

C.5 Exact evaluation and incumbent preservation

For efficiency, LNS evaluates most proposed replacements by subtracting the contributions of the replaced product assignments and adding those of the new assignments. These incremental updates provide the GMV, budget usage, subsidy-weighted GMV, and group-order aggregates used during the search. Because repeated floating-point updates can accumulate small numerical differences, the algorithm periodically recomputes these quantities from the complete subsidy matrix. A policy is recorded as feasible or returned by the solver only after its objective and every original constraint residual have been recomputed from that complete matrix.

LNS stores the current search state separately from the best fully verified feasible policy. Simulated annealing may accept a lower-objective or mildly infeasible state for continued exploration, but this state cannot replace the stored feasible policy. Replacement occurs only when a complete re-evaluation confirms feasibility and a strictly higher objective. At termination, the selected output is evaluated once more from its complete subsidy matrix. If that final check rejects the tentative candidate, LNS returns the most recent fully verified feasible policy instead.

C.6 Energy, acceptance, temperature, and adaptation

Let $v_k(\mathbf{s}) \geq 0$ be the normalized violation of enabled constraint k, let $v_{\max}(\mathbf{s}) = \max_k v_k(\mathbf{s})$, and let w_k be its adaptive weight. The violation term used by SA is

$$V(\mathbf{s}) = v_{\max}(\mathbf{s}) + 0.1 \sum_k w_k v_k(\mathbf{s}). \tag{33}$$

Whenever a proposed policy is accepted by the SA rule and becomes the new current search state, the weights are updated with learning rate $\eta_v = 0.10$:

$$\widetilde{w}_k = (1-\eta_v) w_k + \eta_v \left(1 + \frac{v_k(\mathbf{s})}{v_{\max}(\mathbf{s})}\right) \quad \text{if } v_{\max}(\mathbf{s}) > 0, \tag{34}$$

$$w_k^+ = \widetilde{w}_k \Big/ \left(|\mathcal{C}|^{-1} \sum_{j \in \mathcal{C}} \widetilde{w}_j\right).$$

The same update is applied once to the selected initial state before the first LNS iteration. If $v_{\max}(\mathbf{s}) = 0$, all normalized violations are zero, so the ratio is defined as zero and every target weight equals one. Dividing the updated weights by their mean preserves their relative emphasis while ensuring

$$\frac{1}{|\mathcal{C}|} \sum_{k \in \mathcal{C}} w_k^+ = 1,$$

which prevents the numerical scale of the violation energy from drifting over the search.

Define the normalized objective by

$$\widehat{F}(\mathbf{s}) = F(\mathbf{s}) / S_F.$$

Constraint-violation weights start from $w_k = 1$. Before any feasible incumbent has been found, the energy is

$$\gamma(V) = 10^{-3} + (1 - 10^{-3}) e^{-V}, \qquad E_{\mathrm{pre}}(\mathbf{s}) = V(\mathbf{s}) - \gamma(V(\mathbf{s}))\widehat{F}(\mathbf{s}). \tag{35}$$

Thus feasibility dominates when violations are large, while objective quality receives increasing weight as the search approaches feasibility. Once a feasible incumbent has been found, LNS may temporarily continue exploring mildly infeasible policies, but the permitted violation decreases with elapsed time. Let $f \in [0,1]$ denote the elapsed fraction of the time budget and let $\phi = 0.15$ denote its final portion. Define

$$r(f) = \max\left\{0, \frac{1 - \phi - f}{1 - \phi}\right\}, \qquad c(f) = 0.05\, r(f)^2. \tag{36}$$

A feasible proposal is always eligible for the subsequent SA acceptance test. An infeasible proposal $\mathbf{s}'$ is eligible only if

$$v_{\max}(\mathbf{s}') \le c(f).$$

Because $c(f)$ decreases continuously from 0.05 to zero as f approaches $1-\phi$, the search permits progressively less infeasibility as the remaining time decreases.

Every eligible proposal is scored using the post-feasibility energy

$$E_{\text{post}}(\mathbf{s}) = -\widehat{F}(\mathbf{s}) + 10V(\mathbf{s}). \tag{37}$$

During the final ϕ fraction of the time budget, $c(f) = 0$, so an infeasible proposal is no longer eligible. Probabilistic acceptance of a higher-energy proposal is also disabled during this period. Consequently, once the current search state is feasible, the remaining search accepts only feasible proposals that do not worsen the energy; because $V(\mathbf{s}) = 0$ for these policies, this means that their GMV cannot decrease.

If no feasible incumbent has yet been found, the corridor and E_{post} are not activated. The search continues to use E_{pre} so that it can reduce constraint violations and seek its first feasible policy. Throughout both stages, the best fully verified feasible policy is stored separately from the current search state and cannot be overwritten by an infeasible or lower-objective exploratory state.

For the current search state $\mathbf{s}$ and an admissible proposal $\mathbf{s}'$, let

$$\Delta E = E(\mathbf{s}') - E(\mathbf{s}),$$

where $E = E_{\text{pre}}$ before the first feasible incumbent is found and $E = E_{\text{post}}$ afterward. With current SA temperature $\mathcal{T}(f)$, the proposal is accepted with probability

$$\Pr(\text{accept } \mathbf{s}') = \begin{cases} 1, & \Delta E \le 0, \\ \exp\{-\Delta E/\mathcal{T}(f)\}, & \Delta E > 0 \text{ and } f < 1-\phi, \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

Thus an energy-improving proposal is always accepted, while a higher-energy proposal may be accepted probabilistically only before the final ϕ fraction of the time budget.

The initial temperature is set to the default value $\mathcal{T}_0 = 0.05$. During the early search, whenever a proposal has higher but finite energy than the current state, the algorithm records the positive difference ΔE. Let $\mathcal{D}_+$ denote the recorded differences, and stop adding observations after 64 such values have been collected. Whenever $\mathcal{D}_+$ is nonempty, the initial temperature is recalibrated using

$$\mathcal{T}_0 \leftarrow \max\left\{10^{-8}, \frac{\text{median}(\mathcal{D}_+)}{-\log p_0}\right\}, \qquad p_0 = 0.80. \tag{39}$$

Consequently, an energy increase equal to the observed median would initially be accepted with probability p_0. Until a positive finite energy increase has been observed, the default value $\mathcal{T}_0 = 0.05$ is retained.

With minimum temperature ratio $\rho_{\min} = 10^{-3}$, the elapsed-time cooling schedule is

$$\mathcal{T}(f) = \max\left\{\mathcal{T}_0\rho_{\min}, \mathcal{T}_0\rho_{\min}^{f}\right\}. \tag{40}$$

The temperature therefore decreases from $\mathcal{T}_0$ at the beginning of the run to no less than $\mathcal{T}_0\rho_{\min}$ at its end. The search is considered stalled when it produces neither a new best feasible policy nor an improved least-violating policy. Whenever the stall counter reaches a positive multiple of 40, the next 40 iterations use

$$\mathcal{T} \leftarrow \max\{\mathcal{T}(f), 0.30\mathcal{T}_0\},$$

which temporarily restores greater acceptance of higher-energy proposals. If 160 consecutive non-improving iterations occur before the final ϕ fraction of the time budget, LNS restarts its current search trajectory. It first uses an unused starting policy, requiring that policy to be feasible if a feasible incumbent has already been found. If no eligible unused starting policy remains, the current state is reset to the best verified feasible policy, or to the stored least-violating policy when no feasible policy has yet been found. The separately stored best feasible incumbent is never discarded by a restart.

When no feasible policy has yet been found, LNS identifies the stored least-violating policy by lexicographically minimizing

$$(v_{\max}(\mathbf{s}),\, V_{\text{sum}}(\mathbf{s}),\, -F(\mathbf{s}))\,, \qquad V_{\text{sum}}(\mathbf{s}) = \sum_k v_k(\mathbf{s}).$$

Thus it first minimizes the largest normalized constraint violation, then the sum of all normalized violations, and finally maximizes GMV to break any remaining tie. This comparison is applied to all infeasible policies; no separate threshold is used to classify a policy as near feasible.

C.7 Adaptive operator selection and reward updates

Destroy and repair operators have separate adaptive selectors. At the beginning of an iteration, each selector samples from the operators of its class that are applicable to the current search state and neighborhood. Let $\mathcal{O}_{\text{app}}$ denote this applicable set. Except when a complete group-order neighborhood forces the use of a group master, an operator $a \in \mathcal{O}_{\text{app}}$ is sampled with probability

$$p_a = \frac{\max\{0.05, \omega_a\}}{\sum_{b \in \mathcal{O}_{\text{app}}} \max\{0.05, \omega_b\}}, \tag{41}$$

where ω_a is the current performance weight of operator a. All enabled destroy and repair operators are initialized with $\omega_a = 1$, except the optional `continuous_polish` operator, which starts from 0.2 and is disabled in the main experiments. The lower bound 0.05 ensures that an applicable operator retains a positive sampling probability even after receiving several low rewards.

After the repair proposal has been evaluated and the SA acceptance decision has been made, the selected destroy and repair operators receive a quality gain determined by the iteration outcome. The first verified feasible policy receives

$$q = 0.01.$$

If an existing best feasible objective improves from $F_{\text{best,old}}$ to $F_{\text{best,new}}$, then

$$q = \frac{F_{\text{best,new}} - F_{\text{best,old}}}{S_F}. \tag{42}$$

Thus a feasible-objective improvement is measured relative to the reference objective scale S_F.

Before a feasible policy has been found, an iteration may instead improve the stored least-violating policy. Let $v_{\max}^{\text{old}}$ and $V_{\text{sum}}^{\text{old}}$ denote the maximum and summed normalized violations of the previously stored policy, and let $v_{\max}^{\text{new}}$ and $V_{\text{sum}}^{\text{new}}$ denote those of the new policy. Define

$$\Delta V = \max\{0, v_{\max}^{\text{old}} - v_{\max}^{\text{new}}\} + 0.1 \max\{0, V_{\text{sum}}^{\text{old}} - V_{\text{sum}}^{\text{new}}\}. \tag{43}$$

The corresponding quality gain is

$$q = \max\left\{10^{-4}\Delta V,\, \frac{\max\{0, F_{\text{new}} - F_{\text{old}}\}}{S_F}\mathbf{1}\{\Delta V \le 10^{-12}\}\right\}, \tag{44}$$

where F_{old} and F_{new} are the objectives of the previous and new least-violating policies, and $\mathbf{1}\{\cdot\}$ is the indicator function. Hence violation reduction determines the reward whenever either violation measure improves. Objective improvement contributes only when the violation measures are effectively unchanged.

An accepted proposal that creates neither a new best feasible policy nor an improved least-violating policy receives the small quality gain $q = 10^{-5}$. A rejected proposal receives $q = 0$. The same iteration-level quality gain is assigned to the selected destroy and repair operators, but their rewards differ because their running times are measured separately. If t_a is the time spent by operator a, its reward is

$$R_a = \min\left\{20,\, \frac{1000q}{\max\{t_a, 0.05\}}\right\}. \tag{45}$$

The denominator rewards improvement per unit computation time. The 0.05-second floor prevents extremely short measured times from producing unstably large rewards, while the cap of 20 limits the influence of a single iteration.

The performance weight is updated using a constant-step exponential moving average:

$$\omega_a^{+} = \max\left\{0.05,\, (1-\eta_o)\omega_a + \eta_o(0.05 + R_a)\right\}, \qquad \eta_o = 0.20. \tag{46}$$

A repeatedly effective and computationally efficient operator therefore receives a larger sampling probability in subsequent iterations, whereas an operator receiving little reward gradually moves toward the weight floor. This adaptation controls how often each applicable operator is selected. It does not determine whether the resulting proposal enters the current search trajectory; that decision is made separately by the SA acceptance rule in equation 38.

D Additional Experimental Details

The benchmark instances and experimental manifests are generated and frozen before evaluation, and reported solver configurations and seeds are not changed in response to benchmark outcomes. The solver-independent evaluator recomputes the objective and all original constraint residuals from each returned policy.

Common feasibility tolerances. PD, LNS+SA, and Gurobi are re-evaluated by the same solver-independent evaluator and the same numerical acceptance rules. For subsidy-rate bounds, effective-subsidy caps, GMV-weighted average-subsidy bounds, cross-market range constraints, and merchandise-group ordering constraints, the absolute rate tolerance is

$$\tau_{\mathrm{rate}} = 10^{-5}.$$

For example, an ordering relation $\overline{s}_A \geq \overline{s}_B + m$ is accepted only if $\max\{0, \overline{s}_B + m - \overline{s}_A\} \leq \tau_{\mathrm{rate}}$. For a global or market-specific budget limit B, the monetary tolerance is

$$\tau_B = \max\{10^{-4}, 10^{-8}|B|\}, \qquad D(\mathbf{s}) \leq B + \tau_B.$$

The evaluator recomputes all objectives and residuals directly from the returned subsidy matrix rather than using solver-reported feasibility flags. For large aggregate sums, the implementation advances the floating-point comparison boundary by 16 units in the last place to avoid changing a boundary decision through representation error; this guard does not enlarge the declared tolerance.

Table 3: Held-out instance families. A checkmark means that the corresponding constraint is present. Group-order constraints are global in every family. Active density is the proportion of the NR possible product–market pairs that are active and therefore have a subsidy decision variable; "effective-subsidy cap" bounds the sum of the direct subsidy rate and the coupon discount expressed as a fraction of reference price; and D_c counts ordering relations in one group-order component.

Family	Active density	Global budget	Market budgets	Average band	Range	Coupon tiers per market	Effective-subsidy cap	Group order
`smooth_resources`	100%	✓	all	✓	✓	–	–	–
`sparse_markets`	8–35%	✓	half	✓	✓	–	–	–
`code_thresholds`	85%	✓	all	✓	✓	2	✓	–
`group_components`	80%	✓	all	✓	✓	–	–	$D_c \leq 3$
`code_group_coupled`	75%	✓	all	✓	✓	2	✓	$D_c \leq 4$
`tight_mixed`	60%	✓	all	✓	✓	3	✓	$D_c \leq 4$

Table 4: Complementarity of portfolios containing one PD run and the indicated LNS seed. "LNS-only" counts instances for which LNS is feasible and PD is not. Wins compare PD and LNS on their common-feasible instances. "Gain vs. PD" is the mean relative improvement of the better portfolio objective over PD; "Gain vs. LNS" is defined analogously.

LNS seed	LNS feas.	PD+LNS feas.	LNS-only	PD/LNS wins	Gain vs. PD	Gain vs. LNS
104729	72/72	72/72	1	49/22	0.010%	0.389%
130363	72/72	72/72	1	48/23	0.012%	0.392%
155921	72/72	72/72	1	48/23	0.011%	0.396%

Table 5: Per-family improvement of complete PD over each ablation. "Infeasible" means that the ablation's saved candidate was rejected by the common evaluator; the group-component multiple-choice-master ablation was run only on group-order families.

Family	Generic greedy recovery	No earlier-iterate assignments	No feasible polishing	No group-component multiple-choice master
smooth_resources	$+0.7715\%$	-0.0005%	0.0000%	–
sparse_markets	$+0.5063\%$	-0.0007%	-0.0013%	–
code_thresholds	$+0.8708\%$	$+0.0003\%$	-0.0011%	–
group_components	Infeasible	$+0.0440\%$	$+0.0970\%$	$+0.0389\%$
code_group_coupled	Infeasible	$+0.0145\%$	$+0.1098\%$	$+0.0256\%$
tight_mixed	Infeasible	-0.0313%	$+0.0346\%$	$+0.2864\%$

Table 6: Per-family improvement of complete LNS+SA over each ablation, comparing the best verified result among three fixed seeds. The group-component-search ablation is evaluated only on group-order families.

Family	Random/vanilla neighborhoods	No SA acceptance	No group-component search
smooth_resources	$+1.2711\%$	-0.0020%	–
sparse_markets	$+2.0109\%$	$+0.0029\%$	–
code_thresholds	$+1.0872\%$	$+0.0308\%$	–
group_components	$+1.6171\%$	-0.0021%	$+0.3188\%$
code_group_coupled	$+1.3728\%$	-0.0030%	$+0.3708\%$
tight_mixed	$+1.3022\%$	$+0.0100\%$	$+0.6904\%$

Table 7: Exactness/certification micro-benchmark ($N = 12, R = 2$, 3600 seconds). The Gurobi gap is its final bound gap; PD shortfall is measured against the Gurobi incumbent, and the PD certified gap is $100(U_{\mathrm{PD}} - F_{\mathrm{PD}})/F_{\mathrm{PD}}$.

Family	Gurobi status	Gurobi gap	PD shortfall vs. Gurobi incumbent	PD certified gap
smooth_resources	time limit	$1.59\times10^{-6}\%$	0.0155%	0.0536%
sparse_markets	optimal	$2.10\times10^{-9}\%$	$< 10^{-6}\%$	0.0174%
code_thresholds	optimal	0	0.0052%	0.0069%
group_components	time limit	0.00344%	0.1269%	0.7981%
code_group_coupled	time limit	0.00653%	0.1892%	0.6800%
tight_mixed	time limit	$4.14\times10^{-6}\%$	0.3190%	1.9090%

Discretized MILP baseline. To test whether converting the nonlinear promotion problem to a finite-choice MILP substantially improves Gurobi's performance, each active product–market pair (i, r) is assigned a finite candidate set $\mathcal{S}_{ir} = \{s_{irk} : k \in \mathcal{K}_{ir}\}$. The base candidates are multiples of 0.01 within the declared subsidy interval, augmented with locally feasible domain endpoints and the two numerically distinguishable sides of each coupon threshold; candidates violating a product–market-local constraint are discarded. For every retained candidate, the original exponential demand and coupon rule are evaluated in advance, giving constants G_{irk}, $C_{irk} = D_{ir}(s_{irk})$, and $H_{irk} = s_{irk}G_{irk}$. One-hot binary variables satisfy

$$z_{irk} \in \{0, 1\}, \qquad \sum_{k \in \mathcal{K}_{ir}} z_{irk} = 1,$$

so selected subsidies, GMV, budget cost, and subsidy-weighted GMV are linear functions of $\mathbf{z}$. Budgets and GMV-weighted average-subsidy bounds are therefore linear, and a continuous anchor a_i with $a_i \le s_{ir} \le a_i + \delta_i$ represents each cross-market range constraint exactly over the candidate sets.

For a group-order edge e, introduce a bounded separator $\theta_e \in [\theta_e^L, \theta_e^U]$ and impose

$$H_{A_e} \ge (\theta_e + m_e/2)Q_{A_e}, \qquad H_{B_e} \le (\theta_e - m_e/2)Q_{B_e}.$$

The only remaining products are $\theta_e z_{irk}$. Defining $u_{eirk} = \theta_e z_{irk}$, their exact binary–continuous linearization is

$$\begin{aligned} &u_{eirk} \le \theta_e^U z_{irk}, && u_{eirk} \ge \theta_e^L z_{irk}, \\ &u_{eirk} \le \theta_e - \theta_e^L(1 - z_{irk}), && u_{eirk} \ge \theta_e - \theta_e^U(1 - z_{irk}). \end{aligned} \tag{47}$$

Thus the model supplied to Gurobi is a genuine MILP and is exact over the constructed finite candidate sets, although it is not an exact reformulation of the original continuous-subsidy problem.

We run this baseline for 900 seconds on the 18 held-out instances with $N = 10{,}000$. Gurobi finds an incumbent on 9 of the 18 instances, compared with 18 of 18 feasible solutions returned by the continuous portfolio. Notably, it finds no incumbent on any of the nine instances containing group-order constraints and reaches OPTIMAL status only on the three `sparse_markets` instances. These results show that strong structural coupling remains computationally challenging even after finite discretization yields an MILP. Table 8 reports the family-level results.

Table 8: Gurobi on the 0.01-grid MILP benchmark at $N = 10{,}000$ (900 seconds). "Optimal" is Gurobi's status at the configured 10^{-4} relative MIP gap.

Family	Instances	Incumbents	Optimal status
`smooth_resources`	3	3	0
`sparse_markets`	3	3	3
`code_thresholds`	3	3	0
`group_components`	3	0	0
`code_group_coupled`	3	0	0
`tight_mixed`	3	0	0
Total	18	9	3

Agent interface. Figure 2 shows the implemented user interface for the optimization agent. The left panel supports input construction, saved inputs, and run-history management; the central panel supports natural-language interaction, file upload, and execution-status queries; and the right panel summarizes the validated input, frozen configuration, and optimization results. A user may construct or upload an instance, inspect the validated input, and then either use the default solver hyperparameters or ask the LLM to select hyperparameters for PD and LNS+SA from predefined options. After both optimization methods have been executed, the interface reports the selected solver, recomputed GMV and subsidy cost, feasibility verification, run diagnostics, and downloadable solutions.

LLM-selected and default configurations. The case study uses the single configuration returned by the original LLM call; no solver result was available when that choice was made. For PD, the LLM chose a multiplier-step scale of 1.25 and a final-recovery time-reserve scale of 1.25, compared with 1.0 and 1.0 in the default configuration. For LNS, it chose 24 candidate assignments per product, at most 512 products in a group neighborhood, and two seconds for each restricted MIP, compared with the defaults of 12, 256, and one second. We then ran both the LLM-selected and default configurations on the same 50,000-product instance, using identical solver code, one 900-second run per solver, eight CPU cores per run, and LNS seed 104729. All four returned policies are feasible under the common evaluator, and PD supplies the higher-objective policy under both configurations. The LLM-selected PD reaches its first feasible policy 46.9 seconds earlier than default PD, while its final objective is 0.0139% lower; the LLM-selected LNS objective is 0.1691% lower than default LNS. Thus this case study demonstrates the complete configuration and solver-selection workflow, rather than establishing that one LLM-selected configuration outperforms the default.

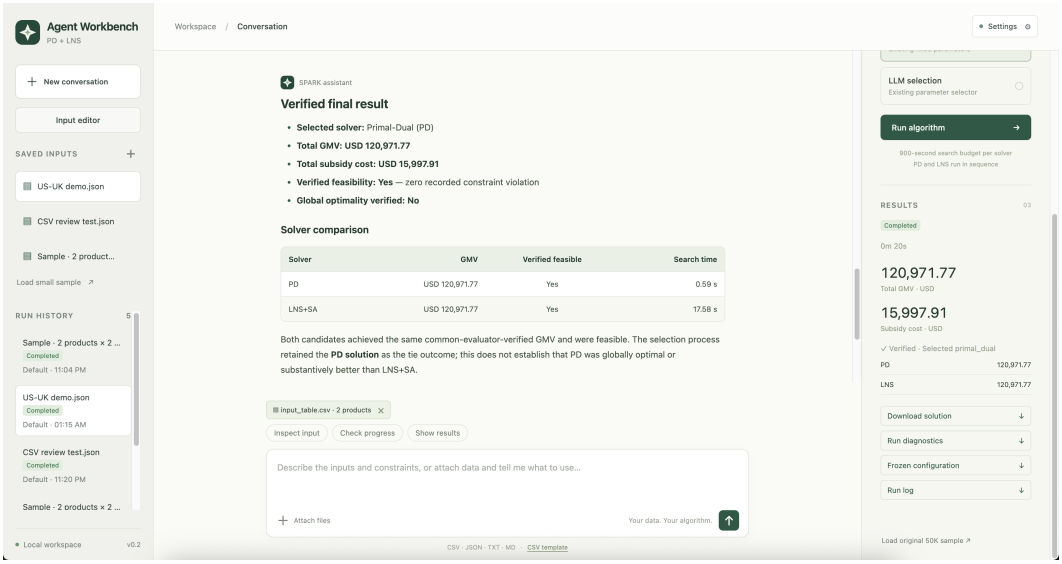

Figure 2: Implemented interface for the two-solver optimization agent.

Table 9: LLM-selected and default configurations on the same 50,000-product instance. Parentheses give time to the first feasible policy; the final column is the higher verified objective returned by PD or LNS+SA.

Configuration	PD objective (first feasible, s)	LNS objective (first feasible, s)	Selected objective
LLM-selected	464,240,666.61 (328.1)	459,390,714.20 (413.6)	464,240,666.61
Frozen default	464,305,232.68 (375.0)	460,169,034.50 (285.1)	464,305,232.68

Table 10: Family-level PD–LNS comparison. For each instance, $F_{\mathrm{LNS,best}}$ is the best verified objective among the three predeclared seeds and $\Delta_{\mathrm{PD-LNS}} = 100(F_{\mathrm{PD}} - F_{\mathrm{LNS,best}})/F_{\mathrm{LNS,best}}$. A positive value favors PD. The mean and median are taken across the common-feasible instances in each family; intervals are percentile-bootstrap 95% CIs for the mean.

Family	Inst.	Common	PD/LNS wins	Mean $\Delta_{\mathrm{PD-LNS}}$ [95% CI]	Median $\Delta_{\mathrm{PD-LNS}}$
`smooth_resources`	15	15	15/0	$+0.803\ [+0.411, +1.208]\%$	$+0.418\%$
`sparse_markets`	9	9	1/8	$-0.013\ [-0.018, -0.007]\%$	-0.017%
`code_thresholds`	9	9	8/1	$+0.209\ [+0.072, +0.366]\%$	$+0.112\%$
`group_components`	9	9	3/6	$-0.007\ [-0.038, +0.040]\%$	-0.033%
`code_group_coupled`	15	15	9/6	$+0.444\ [+0.195, +0.700]\%$	$+0.162\%$
`tight_mixed`	15	14	9/5	$+0.389\ [+0.131, +0.654]\%$	$+0.225\%$

Table 11: Scale-level PD–LNS comparison. Mean and median $\Delta_{\mathrm{PD-LNS}}$ aggregate the instance-level relative differences defined in Table 10 over common-feasible instances at each scale; intervals are percentile-bootstrap 95% CIs for the mean.

N	Common	PD/LNS wins	Mean $\Delta_{\mathrm{PD-LNS}}$ [95% CI]	Median $\Delta_{\mathrm{PD-LNS}}$
10^2	18	6/12	$-0.033\ [-0.056, -0.013]\%$	-0.015%
10^3	18	9/9	$+0.015\ [-0.007, +0.038]\%$	-0.003%
10^4	18	13/5	$+0.230\ [+0.139, +0.323]\%$	$+0.206\%$
5×10^4	8	8/0	$+1.163\ [+0.941, +1.415]\%$	$+0.996\%$
10^5	9	9/0	$+1.417\ [+1.174, +1.675]\%$	$+1.178\%$

For both tables, the three LNS seeds are first reduced to the best verified objective per instance; seeds are not treated as independent observations. We then resample instances within each family or scale 20,000 times with a fixed analysis seed. The intervals quantify variation across generated instances, not across repeated LNS trajectories. For the portfolio–Gurobi comparison, we analogously resample only the 33 instances on which Gurobi produced a verified incumbent. The mean shortfall is 0.0059% (95% CI $[0.0015, 0.0102]\%$) and the median is 0.0054% ($[0.0035, 0.0074]\%$); the 39 censored cases are not assigned objectives.

PD wins all 15 `smooth_resources` comparisons, with its median advantage reaching 1.61% at $N = 50{,}000$ and 1.93% at $N = 100{,}000$. `sparse_markets` and `group_components` favor best-of-three LNS, while the two coupled families change from LNS-competitive at small scale to PD-dominant at $N \geq 10{,}000$. This reversal is the main empirical evidence that the two search mechanisms are complementary rather than redundant.

Table 12: Median time in seconds to the first feasible incumbent. For each instance, the LNS time is the earliest among the three predeclared seeds. The PD median at $N = 50000$ excludes the one run that returned no feasible policy.

N	10^2	10^3	10^4	5×10^4	10^5
PD	0.90	9.52	78.18	466.83	399.11
LNS, earliest of three	0.88	11.16	55.86	290.49	573.53