ORDERCOVER: STRUCTURE-AWARE BUDGETED SEARCH OVER PROMPT-COMPONENT ORDERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Prompt optimization typically rewrites instructions or selects demonstrations, although reordering existing components can change model behavior while keeping their text fixed. We formulate fixed-content prompt-order optimization and introduce ORDERCOVER, a budgeted method that selects candidate permutations using structural feature coverage and Kendall-distance separation. Candidate construction uses only permutation structure, without labels, logits, gradients, or model scores. We evaluate ORDERCOVER through complete landscapes across 10 tasks and 7 models, held-out application prompts, and composition with GEPA and MIPROv2. In the controlled suite, ORDERCOVER improves the original Anchor by 3.35 percentage points, recovers 82.99% of the available Anchor-to-Oracle headroom, and exceeds budget-matched Random by 0.46 points. On application prompts, it improves Anchor in 12 of 15 cells, although the aggregate confidence interval includes zero. After content optimization, order search adds 1.48 points on average across 24 cells and improves 19 of them. Order headroom persists for strong starting orders and after content optimization, although the advantage over budget-matched Random is modest. ORDERCOVER offers a transparent, content-preserving post-processing step for prompts whose components can be moved as whole units.

1 INTRODUCTION

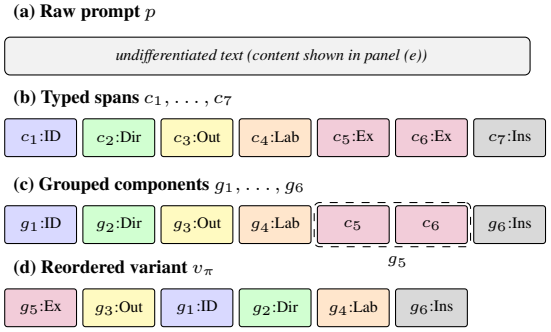
Prompts combine instructions, constraints, label definitions, demonstrations, tool descriptions, retrieved context, and instance inputs into an ordered context. Reordering these components can change model behaviour even when their text is unchanged (Lu et al., 2022; Sclar et al., 2024; Mao et al., 2024; Guan et al., 2025; Liu et al., 2024). Yet prompt optimization typically rewrites instructions or selects demonstrations, while component order is inherited from an author, template, or optimizer. We treat order as a separate optimization variable.

We study fixed-content prompt-order optimization: given fixed prompt components and a limited evaluation budget, which component orders should be tested? Exhaustive search grows factorially, while uniform random search ignores structure shared across permutations. Order search is meaningful only when components can be moved without breaking the prompt, since schemas, scoped markup, and cross-references can make some text unsafe to move. We therefore enforce validity when constructing components rather than as constraints on orders: each component preserves its text exactly, the components must reconstruct the original prompt, and search permutes them as whole units. Prompts that would require constraints between components fall outside our scope.

We introduce ORDERCOVER, a structure-aware budgeted selector. It represents each order through precedence, directed-adjacency, and coarse-position features, then greedily selects candidates that expand feature coverage while remaining separated under Kendall distance. Candidate construction uses only permutation structure, without labels, logits, gradients, or model scores.

Across complete landscapes for 10 tasks and 7 models, ORDERCOVER improves the original Anchor by 3.35 percentage points, captures 82.99% of the available Anchor-to-Oracle headroom, and exceeds budget-matched Random by 0.46 points. On held-out application prompts, it improves Anchor in 12 of 15 cells; after GEPA or MIPROv2, order search adds 1.48 points on average across 24 cells. The opportunity is not confined to weak starting orders, but gains remain heterogeneous and

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.



(e) Worked example

Span	Type	Text snippet	Group
c_1	ID	"You are given a customer review."	g_1
c_2	Dir	"Decide whether the review is positive or negative."	g_2
c_3	Out	"Answer with only positive or negative."	g_3
c_4	Lab	"Use positive for favorable sentiment…"	g_4
c_5	Ex	"Review: 'I loved it.'"	g_5
c_6	Ex	"Answer: positive."	
c_7	Ins	"Review: {{input}}"	g_6

Figure 1: **From a raw prompt p to a permuted variant. (a)-(b)** LLMDECOMPOSE partitions p into typed spans $c_1, \ldots, c_7$ (box colour encodes type). **(c)** LLMGROUP bundles adjacent same-type spans c_5, c_6 into one movable unit g_5 (dashed); other spans map one-to-one. Neither stage rewrites text. **(d)** A prompt variant permutes grouped components; text inside each g_j is unchanged. **(e)** Tabular form of the same example: seven spans $\rightarrow$ six components, anchor $\pi^0 = (g_1, \ldots, g_6)$, $|\Pi_M| = 6! = 720$ variants. Type abbreviations are defined in 3.1.

the structure-aware advantage over Random is smaller than the broader benefit of treating order as searchable.

Contributions. We contribute: (i) a fixed-content formulation in which validated, exact-text components are permuted as whole units, with prompts requiring cross-component constraints explicitly out of scope; (ii) ORDERCOVER, a structure-aware selector for budgeted permutation search; (iii) exhaustive and held-out evaluation spanning controlled, application, and post-optimization settings; and (iv) analyses of anchor quality, structuralization sensitivity, landscape movement, transfer, and failure cases.

2 RELATED WORK

2.1 PROMPT ORDER AND POSITION SENSITIVITY

Language models are sensitive not only to prompt content but also to where that content appears. Prior work has documented substantial variation under prompt formatting changes, instruction ordering, and context-position effects (Sclar et al., 2024; Mao et al., 2024; Guan et al., 2025; Liu et al., 2024), while Lu et al. (2022) search over few-shot example orders using model-derived criteria. Concurrent work uses landscape diagnostics for permutation search (Mambwe, 2026), while other approaches marginalize permutations or train for consistency (Huang et al., 2026; Zheng et al., 2026). We study a different setting: heterogeneous, semantically movable prompt components whose text remains fixed. Our question is how to construct a small evaluation portfolio using only permutation structure, before any labels, logits, or model scores are consulted.

2.2 PROMPT OPTIMIZATION

Prompt optimization methods search for better instructions, demonstrations, or program prompts. Gradient-inspired and LLM-driven methods such as APO and OPRO rewrite prompt text directly (Pryzant et al., 2023; Yang et al., 2024); DSPy-style optimizers such as BootstrapFewShot and MIPRO search over demonstrations and instruction candidates for modular language-model programs (Khattab et al., 2024; Opsahl-Ong et al., 2024); and newer optimizers such as GEPA optimize prompts through reflective or feedback-driven proposal loops (Agrawal et al., 2026). GrIPS includes phrase-level swaps, but edits and scores instructions rather than selecting unchanged component orders (Prasad et al., 2023). These approaches are content-changing: their output can differ from the input prompt in wording, examples, formatting, and structure. We study a complementary question. Once such an optimizer has produced a prompt, does the fixed text still have useful order headroom? If yes, order search can be applied after content optimization without asking the optimizer to produce new text.

2.3 Structured Prompt Representations

Several lines of work treat prompts as structured objects rather than flat strings. Prompt-template analyses decompose prompts into functional parts such as task descriptions, constraints, examples, and instance inputs (Mao et al., 2025); Adaptive Prompting composes discrete prompting techniques and predicts an input-specific composition (Spliethöver et al., 2025); modular LM programming frameworks render prompts from program components (Khattab et al., 2024); and PromptAnatomy studies the effects of modifying or perturbing individual prompt components (Zheng et al., 2025). We use structure for a narrower purpose: to define valid movable components for fixed-content order search. Our representation therefore requires exact-text preservation, and we apply order search only to prompts whose components can be moved as whole units without breaking the prompt.

3 Fixed-Content Prompt-Order Optimization

This section defines the fixed-content search space used throughout the paper. Given a raw prompt, we produce a validated ordered list of grouped components: contiguous text blocks whose text and internal order remain fixed. Order search operates over permutations of the resulting grouped components while preserving their text and internal order. Figure 1 illustrates the structuralization pipeline and records the same worked example in tabular form (panel (e)).

This fixed-content representation is the controlled intervention used throughout the paper: downstream variants differ only in component order, while component text and within-component order are preserved.

3.1 From Raw Prompt to Grouped Components

Let p denote a raw prompt template. We convert p into candidate movement units through two stages.

Typed-span decomposition: An LLM-assisted decomposer partitions p into an ordered sequence of contiguous typed spans $C(p) = (c_1, \ldots, c_K)$, where $c_i = (s_i, \tau_i)$ consists of a verbatim text span s_i and a type $\tau_i \in \mathcal{T}$. The nine types, adapted from prior prompt-template analysis (Mao et al., 2025), are `Role`, `InputDescription`, `Directive`, `OutputFormatting`, `LabelSemantics`, `Constraints`, `AuxiliaryHints`, `Examples`, and `InstanceInput`. Appendix A.1 provides their operational definitions and boundary rules.

Grouping into candidate movement units: Typed spans can be too fine-grained to reorder independently; for example, an example input and its answer should remain together. An LLM-assisted grouper therefore maps $C(p)$ to

$$G(p) = \textsc{BuildGroups}(C(p)) = (g_1, \ldots, g_M), \qquad M \leq K,$$

where each g_j is a contiguous bundle of one or more adjacent spans that form a cohesive unit. Deterministic checks validate the proposed types and group boundaries and require that concatenating the spans and groups in their original order reconstructs p exactly. Thus, structuralization changes only the granularity at which order may vary; it neither edits prompt text nor changes within-group order.

Implementation details, exact LLM prompts, validation rules, and candidate-pool pass rates are provided in Appendices A.2–A.4. Each resulting group is treated as a movable unit: its text and internal span order remain fixed, while the groups may be permuted. Henceforth, we refer to the grouped components g_j simply as *components*.

3.2 Fixed-Content Order Space and Evaluation Objective

Given $G(p) = (g_1, \ldots, g_M)$, let $\mathfrak{S}_M$ denote the full set of $M!$ component permutations. We treat every grouped component as movable as a whole and use the complete permutation space

$$\Pi(p) = \mathfrak{S}_M. \tag{1}$$

Grouping preserves dependencies within each component because its text and internal span order are never changed. The current formulation does not impose additional movement or precedence

constraints after grouping; prompts for which particular groups must remain fixed or cross-group dependencies must be preserved fall outside its scope. Validity is thus enforced when components are constructed (Section 3.1), so every $\pi \in \Pi(p)$ is a valid order, including instance-first orders.

For any $\pi \in \Pi(p)$, let $v_\pi = \textsc{Linearize}(G(p), \pi)$ denote the prompt obtained by concatenating the component texts according to π, and define $\mathcal{V}(p) = \{v_\pi : \pi \in \Pi(p)\}$. Every variant contains the same component texts and preserves within-component order; only their presentation order changes.

For a model m and prompt v evaluated on dataset $\mathcal{D}_t$ using metric ℓ, we define

$$S(v, m, t) = \frac{1}{|\mathcal{D}_t|} \sum_{(x,y)\in\mathcal{D}_t} \ell\big(m(v, x), y\big), \qquad v^* = \arg\max_{v\in\mathcal{V}(p)} S(v, m, t). \tag{2}$$

The **Anchor** prompt $v_{\text{anchor}} = v_{\pi^0}$ uses the original component order π^0.

Exhaustive evaluation becomes impractical as the order space grows. Given a budget B, let $\mathcal{E}_B \subseteq \mathcal{V}(p)$ denote the set of variants selected by a search policy, with $|\mathcal{E}_B| \le B$. After evaluating only these variants, the policy returns

$$\widehat{v}_B = \arg\max_{v\in\mathcal{E}_B} S(v, m, t), \qquad \mathcal{E}_B \subseteq \mathcal{V}(p), \quad |\mathcal{E}_B| \le B. \tag{3}$$

The search problem is therefore to construct $\mathcal{E}_B$ so that $S(\widehat{v}_B, m, t)$ is high without evaluating variants outside this set. Section 4 instantiates this budgeted search with a structure-only policy that prioritizes structural coverage and diversity among the selected orders.

4 OrderCover: Budgeted Search over Prompt Orders

Section 3 defines a fixed-content order space in which every prompt variant contains the same components and differs only in their order. The remaining question is algorithmic: under a budget of only B prompt evaluations, which orders should we test? We address this with OrderCover, a structure-only search method that scores each candidate order by (i) how many new structural patterns it adds beyond the orders already selected and (ii) how far it is from those already selected orders.

The goal is therefore not to generate new prompt text, but to choose a small set of informative orders from the fixed permutation space before any model evaluation.

4.1 Structural Representation of Prompt Orders

OrderCover constructs its candidate set without observing task labels, model outputs, or candidate scores. It therefore represents each order using only relations among component identities. For a permutation $\pi \in \Pi(p)$ over M components, let $\pi^{-1}(i)$ denote the position of component i, and let $[M] = \{1, \dots, M\}$.

We encode three complementary aspects of an order: *precedence*, which captures global pairwise order; *directed adjacency*, which captures local transitions between neighbouring components; and *coarse position*, which records the region of the component sequence occupied by each component. Formally,

$$\begin{aligned} F_{\text{prec}}(\pi) &= \{(i \prec j) \mid i, j \in [M],\ i \neq j,\ \pi^{-1}(i) < \pi^{-1}(j)\}, \\ F_{\text{adj}}(\pi) &= \{(i \triangleright j) \mid i, j \in [M],\ \pi^{-1}(j) = \pi^{-1}(i) + 1\}, \\ F_{\text{pos}}(\pi) &= \{(i, b_\pi(i)) \mid i \in [M]\}, \end{aligned} \tag{4}$$

where $b_\pi(i) \in \{1, \dots, A\}$ assigns component i to one of A equal-width position buckets. We use $A = 3$, corresponding to coarse early, middle, and late regions; Appendix B.1 gives the exact definition.

The structural representation of π is

$$F(\pi) = F_{\text{prec}}(\pi) \cup F_{\text{adj}}(\pi) \cup F_{\text{pos}}(\pi).$$

For a selected set of orders E, let

$$C(E) = \bigcup_{\rho\in E} F(\rho)$$

denote the features already covered. The coverage gain of a candidate π is therefore $|F(\pi) \setminus C(E)|$.

Coverage alone does not control how dispersed the selected orders are within the permutation space. We measure separation using normalized Kendall distance,

$$\bar{d}_K(\pi, \rho) = \frac{d_K(\pi, \rho)}{\binom{M}{2}}, \tag{5}$$

where $d_K(\pi, \rho)$ counts the pairwise order relations on which π and ρ disagree, equivalently the minimum number of adjacent swaps required to transform one order into the other (Kendall, 1938). The next subsection combines feature coverage and nearest-neighbour Kendall separation to construct the evaluation set.

These descriptors are not assumed to predict model performance or to exhaust all order effects. They provide a model-independent criterion for distributing a limited evaluation budget across orders with different global, local, and positional structure.

4.2 How OrderCover Chooses the Next Order

The search begins from the anchor order together with a small deterministic seed set of local and non-local variants. After that, OrderCover builds the evaluation set greedily. At any step, let E denote the set of orders already selected. For any unevaluated candidate $\pi \notin E$, we score it by

$$\text{Score}(\pi \mid E) = |F(\pi) \setminus C(E)| + \lambda |F(\pi)| \min_{\rho \in E} \bar{d}_K(\pi, \rho), \tag{6}$$

subject to approximately balanced allocation across the four anchor-distance buckets defined in Appendix B.1. For a fixed prompt, $|F(\pi)|$ is constant across candidate orders and places the coverage and separation terms on comparable scales. We select $\lambda = 0.6$ on the five-model subset of the controlled suite (50 of the 70 task–model pairs) and keep it fixed without retuning for the two remaining models and for the realistic and post-optimization suites. Appendix B.4 reports the corresponding sensitivity analysis.

The two terms balance breadth (new structural patterns relative to $C(E)$) and diversity (Kendall distance from the nearest already-selected order); Appendix B works through one greedy step on a four-component example.

Importantly, this procedure is entirely structure-only: before model evaluation, it uses no labels, logits, gradients, or prompt scores. Its decisions depend only on the combinatorial structure of the permutation space. Appendix B gives the formal notation, bucket thresholds, initialization details, and implementation notes.

5 Experimental Design

We evaluate OrderCover in three regimes: complete controlled landscapes, prompts from real applications, and prompts produced by content optimizers. Within each experimental cell, all methods use the same componentization, order space $\Pi(p)$, decoding configuration, and data split. **Anchor** denotes the original order; **Random** samples valid orders uniformly without replacement; and **Oracle** denotes the best order in a complete landscape. OrderCover includes Anchor and uses budget $B = \min(M^2, M!)$; whenever Random is reported, it also includes Anchor and receives the same budget.

5.1 Controlled Enumerable Suite

The controlled suite contains eight Super-NaturalInstructions tasks (Wang et al., 2022), plus ARC-Challenge (Clark et al., 2018) and MMLU-Pro (Wang et al., 2024). We evaluate seven checkpoints: Llama-3.1-8B and Llama-3.3-70B (Grattafiori et al., 2024), Ministral-3-8B (Liu et al., 2026), Qwen3.5-9B and Qwen3.5-35B-A3B (abbreviated Qwen3.5-35B) (Qwen Team, 2026a), Gemma-4-31B (Gemma Team et al., 2026), and gpt-oss-120b (Agarwal et al., 2025), yielding 70 task–model pairs. Each prompt has at most six validated components, allowing us to evaluate every valid order on a fixed evaluation set. Oracle therefore provides an exact descriptive ceiling, not a deployable baseline. Appendix C.1 gives the task map, component counts, and metrics.

Method	Acc.	ΔAnchor	HC	Oracle gap	Oracle matches
Anchor	64.03%	0.00	0.00%	3.97	4/70
Random	66.92%	+2.89	71.01%	1.08	7/70
OrderCover	67.38%	+3.35	82.99%	0.62	29/70
Oracle	68.00%	+3.97	100.00%	0.00	70/70

Table 1: Controlled-suite results at the matched evaluation budget $B = M^2$. Accuracy, ΔAnchor, and Oracle gap are reported in percentage points. HC denotes Anchor-to-Oracle headroom captured, averaged over the 66 pairs with non-zero headroom. Random is the mean best-found score across 10,000 independently resampled candidate sets; each set contains B orders sampled without replacement and always includes Anchor. All quantities are computed from unrounded scores.

5.2 Realistic Prompt Suite

The realistic suite uses five prompts from public application repositories documented by Mao et al. (2025): Steamship, LlamaIndex, DataCopilot, CASALIOY, and Indexify. They cover classification, text-to-SQL, contextual QA, and retrieval-augmented QA. We evaluate all five prompts with Gemma-4-31B, Ministral-3-8B, and GPT-5.6 Terra[1], yielding 15 template–model cells. Each method selects an order on development data, and the frozen order is evaluated on the disjoint test split. We report Anchor, Random, and OrderCover, but not Oracle. Appendix C.4 gives prompt provenance, datasets, metrics, and audit records.

5.3 Content-Optimized Prompt Suite

The post-optimization suite crosses four SNI tasks, the same three models, and GEPA (Agrawal et al., 2026) or MIPROv2 (Opsahl-Ong et al., 2024), yielding 24 task–model–optimizer cells. For each cell, we freeze the optimized instruction, reorder only its instruction blocks, select on the development split used by the optimizer, and evaluate the selected order once on the untouched test split. The central comparison is Optimized versus Optimized+OrderCover; without a matched Random arm, this suite evaluates residual order headroom rather than selector superiority. Appendix C.3 gives the tasks, splits, optimizer settings, and saved prompt artifacts.

Evaluation and uncertainty. Target-model evaluation uses deterministic decoding (temperature 0). We report task metrics, paired method differences, win/tie/loss counts, and 95% bootstrap intervals. The controlled intervals resample tasks and models; the realistic-suite intervals use hierarchical paired resampling, while the post-optimization intervals resample cell-level paired differences. Full details on splits, seeds, resampling, parsing, and cost reporting appear in Appendix C.2.

6 Results

6.1 Controlled Evaluation: Headroom and Budgeted Recovery

The complete permutation landscapes first quantify how much performance is available from component order alone. Across the 70 task-model pairs, Anchor averages 64.03% accuracy, whereas exhaustive Oracle selection averages 68.00%, corresponding to 3.97 percentage points of order headroom (95% two-way bootstrap interval: $[2.35, 5.99]$). Oracle strictly improves on Anchor in 66 pairs and ties it in four. Because all variants share the same component text, this gap is attributable to order; since selection and scoring share one evaluation set here, it is an in-sample ceiling, and Sections 6.2–6.3 report held-out results.

At the matched budget, OrderCover reaches 67.38% accuracy, improving on Anchor by 3.35 points (95% interval: $[1.94, 5.11]$) and recovering 82.99% of the available headroom. It strictly improves on Anchor in 65 pairs and ties it in five; degradation is impossible in this controlled setting because Anchor is included in the evaluated candidate set. The selected score is 0.62 points below Oracle on average and 0.10 points below at the median. OrderCover matches Oracle in 29 pairs and lies within one point of it in 56.

[1] https://developers.openai.com/api/docs/models/gpt-5.6-terra

(a) Realistic application prompts: 5 applications × 3 models					
Reference	**Cells**	**Mean Δ**	**Median Δ**	**95% CI**	**W/T/L**
Anchor	15	+3.32	+2.23	$[-0.43, 8.69]$	12/2/1
Random	15	+0.84	+0.62	$[-0.02, 1.13]$	14/0/1

(b) Content-optimized prompts: 4 tasks × 3 models × 2 optimizers						
Optimizer	**Cells**	**Optimized**	**Opt.+OC**	**Mean Δ**	**95% CI**	**W/T/L**
GEPA	12	71.58	72.92	+1.33	$[+0.58, +2.21]$	9/3/0
MIPROv2	12	69.29	70.92	+1.63	$[+0.88, +2.38]$	10/1/1
All cells	24	70.44	71.92	+1.48	$[+0.94, +2.06]$	19/4/1

Table 2: Aggregate results from two held-out experiments; the panels are not pooled. In panel (a), Δ is ORDERCOVER minus the indicated reference, with hierarchical-bootstrap intervals. In panel (b), Δ is Optimized+ORDERCOVER minus Optimized, with nonparametric cell-bootstrap intervals. Scores and differences are percentages and percentage points; W/T/L counts evaluation cells.

Budget-matched Random reaches 66.92% accuracy and recovers 71.01% of the available headroom. ORDERCOVER adds 0.46 points over Random (95% interval: $[0.21, 0.83]$), with 53 wins, 6 ties, and 11 losses. Thus, making component order searchable accounts for most of the improvement over Anchor, while structure-aware selection provides a smaller but reliable advantage at the same budget. Further results and selector ablations appear in Appendices D and B.5.

6.2 ORDERCOVER ON PROMPTS FROM REAL APPLICATIONS

Table 2(a) summarizes the separate realistic-prompt evaluation over five public application prompts and three models. ORDERCOVER improves over the repository-provided Anchor in 12 of 15 cells, ties in two, and is lower in one. The mean gain is 3.32 percentage points (95% hierarchical bootstrap interval: $[-0.43, 8.69]$), with a positive median gain of 2.23 points despite the heterogeneous effect sizes. Relative to budget-matched Random search, ORDERCOVER has a positive point estimate in 14 of 15 cells, with a mean gain of 0.84 points ($[-0.02, 1.13]$). These results indicate recurring order headroom in application prompts, while the additional advantage over Random is generally modest. Complete scores, prompt provenance, and output-validity checks appear in Appendix E.

6.3 ORDER SEARCH COMPLEMENTS CONTENT OPTIMIZATION

We separately test whether content-optimized prompts retain order-level headroom. This experiment uses four SNI tasks, three models, and two optimizers, yielding 24 task–model–optimizer cells independent of the realistic-prompt suite. For each cell, we freeze the prompt produced by GEPA or MIPROv2 and apply ORDERCOVER only to its component order.

As shown in Table 2(b), optimization raises mean held-out accuracy from 67.63% for the corresponding Anchors to 70.44%, after which ORDERCOVER increases it further to 71.92%. The additional mean gain is 1.48 percentage points ($[0.94, 2.06]$), with 19 wins, four ties, and one loss. Gains occur with both GEPA ($+1.33$ points) and MIPROv2 ($+1.63$ points); the single loss is 0.5 points, ruling out a universal-improvement claim.

These results show that frozen content-optimized instructions can retain order-level headroom, but without a matched Random arm they do not establish selector superiority in this regime; that comparison is evaluated separately in Section 6.1. Complete per-cell results, optimizer configurations, component counts, budgets, and selected orders appear in Appendix F.

7 ANALYSIS AND DISCUSSION

7.1 IS ORDER SEARCH LIMITED TO WEAK STARTING PROMPTS?

Prior work has documented substantial variation across prompt orders and formats (Lu et al., 2022; Sclar et al., 2024) and explored intrinsic proxies such as prompt perplexity for identifying effective

prompts (Gonen et al., 2023). We ask a different practical question: does a high starting accuracy indicate that little useful order headroom remains?

Across the 70 controlled task–model pairs, Anchor accuracy is only weakly correlated with Oracle–Anchor headroom, OrderCover–Anchor gain, and OrderCover–Random gain; all corresponding Pearson coefficients have absolute value at most 0.12, and all Spearman coefficients at most 0.20. Absolute accuracy is therefore a poor diagnostic of remaining order headroom, partly because it conflates task difficulty with the quality of the Anchor's position within its own landscape.

We address this confound using each Anchor's tie-aware percentile in the complete permutation landscape. Under fixed weak, medium, and strong tiers, 42 of 70 Anchors are strong, with a mean percentile of 83.65. OrderCover nevertheless moves this group to the 96.86th percentile and remains 2.14 percentage points above Anchor and 0.34 points above budget-matched Random. Moreover, the largest mean gain over Anchor occurs in the medium rather than the weak tier. The controlled improvements are thus not driven only by poorly ordered starting prompts.

The separate post-optimization experiment provides convergent evidence that prompts produced by GEPA or MIPROv2 can retain further order headroom (Table 2(b)). Nevertheless, neither high Anchor accuracy, a strong landscape rank, nor prior content optimization guarantees a useful gain. The practical prerequisites remain components that can be moved as whole units without breaking the prompt, and a development evaluation budget. Full correlation and tier breakdowns appear in Appendix D.

7.2 Sensitivity to Decomposition and Component Taxonomy

Because structuralization defines the order space, we test whether the studied $M \leq 6$ regime depends on the taxonomy or decomposition model. Under the external taxonomy of Mao et al. (2025), 92.9% of 2,149 parsed application templates have at most six non-residual components, or 84.4% when the residual `others` category is conservatively counted as a component. A weighted estimate from 200 templates processed with our operational taxonomy gives 97.1%. Both taxonomies therefore place most application prompts within the regime studied here.

Holding the prompt text, taxonomy, instructions, and validators fixed, we rerun structuralization on 209 prompts with gpt-oss-120b and Qwen3.6-35B-A3B (Qwen Team, 2026b). Agreement with the baseline on whether $M \leq 6$ is 93.8% and 96.7%, respectively, and both alternatives preserve this regime for all ten controlled prompts. Exact counts are less stable, mainly because of split and merge decisions. The evidence thus supports stability of the component-count regime, not invariant segmentations, selected orders, or gains. Appendix G.1 reports the full distributions and disagreement audit.

7.3 Movement through the Permutation Landscape

The complete landscapes reveal whether OrderCover merely finds a small local improvement or reaches a high-performing region of the order space. Across the 70 task–model pairs, the mean tie-aware percentile increases from 67.02 for Anchor to 96.45 for OrderCover. The selected order lies in the top decile in 65 pairs and the top 5% in 52. Mean percentile movement is 29.43 points (95% two-way bootstrap interval: $[20.33, 40.25]$), showing that the fixed evaluation budget usually reaches a near-top region of the complete landscape (Figure 2).

The minimum normalized Kendall distance to an accuracy-tied Oracle decreases from 0.446 for Anchor to 0.210 for OrderCover, a mean reduction of 0.236 ($[0.145, 0.329]$). OrderCover is closer to an Oracle in 55 pairs, equally distant in nine, and farther in six. The selected percentile also remains high when the evaluated fraction is small: in the seven $M = 6$ landscapes, OrderCover evaluates only $36/720 = 5\%$ of all orders but reaches the 96.84th percentile on average.

These diagnostics are descriptive rather than causal. All six pairs that move farther from every accuracy-tied Oracle still improve over Anchor, showing that accuracy is not monotonic in Kendall distance. Same-task cross-model transfer produces an above-midpoint rank signal but trails the target Anchor and target-specific OrderCover on average, so target-model selection remains necessary. Full definitions, grouped results, correlations, and failure cases appear in Appendix D.3; transfer and the supplementary perturbation stress test are reported in Appendices G.2 and G.3.

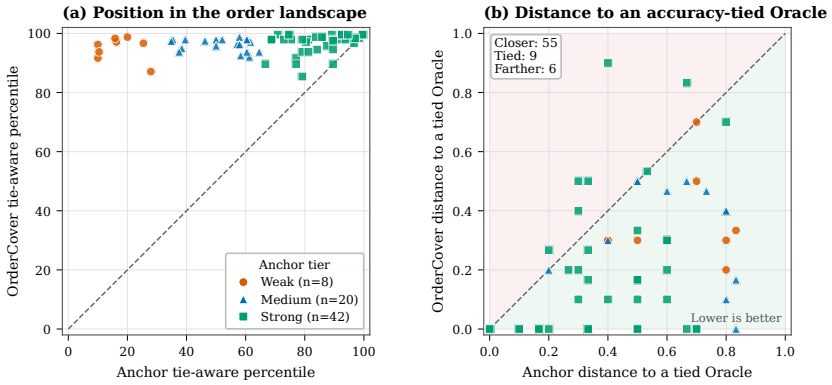


Figure 2: Movement from Anchor to ORDERCOVER in the 70 complete permutation landscapes. **(a)** Tie-aware landscape percentiles; points above the diagonal indicate movement to a better-scoring region. Colors and markers denote weak, medium, and strong Anchor tiers. **(b)** Minimum normalized Kendall distance to any accuracy-tied Oracle; points below the diagonal move closer to an Oracle.

8 LIMITATIONS

Our evidence covers relatively small order spaces ($M \leq 6$): the default budget evaluates 16 of 24 orders for $M = 4$, but only $36/720 = 5\%$ for $M = 6$. A matched-budget Kendall farthest-first ablation trails ORDERCOVER by only 0.08 points on average (Appendix B.5). Broader comparisons with permutation designs (Shao et al., 2026) remain open. Effectiveness also remains untested for larger, interdependent prompts such as agent instructions, tool specifications, and API schemas. Results depend on structuralization: alternative taxonomies and decomposition models generally preserve the coarse $M \leq 6$ regime, but not exact boundaries, selected orders, or gains. Exact-text preservation likewise does not ensure semantic permutability; prompts requiring cross-group constraints, or lacking meaningful movable components, fall outside the method's scope.

Empirical coverage is limited to ten controlled tasks and seven models under deterministic decoding. GPT-5.6 Terra appears only in the application and post-optimization suites; explicit reasoning modes are not studied; application prompts are evaluated on aligned held-out datasets rather than live deployments; and each optimizer result uses one saved artifact from one seed. MIPROv2 is evaluated in an instruction-only configuration with demonstration optimization disabled, so these results do not characterize its full search space. The transfer analysis holds the task and order universe fixed and therefore does not establish portability across tasks, templates, or structuralizations. Gains are also not guaranteed: matched Random remains competitive in some cells, the application-suite intervals include zero, and post-optimization order search can tie or slightly reduce held-out performance. Finally, selection requires a representative labeled development set and a one-time evaluation budget, although the deployed order adds no model calls or latency.

9 CONCLUSION

We formulated fixed-content prompt-order optimization and introduced ORDERCOVER, which selects structurally diverse candidate orders without model scores during candidate construction. Across 70 complete task–model landscapes, it improves Anchor by 3.35 percentage points, recovers 82.99% of the available headroom, and exceeds budget-matched Random by 0.46 points. The complete landscapes further show that this opportunity is not confined to weak starting orders. On prompts from real applications, ORDERCOVER improves Anchor in 12 of 15 cells; after GEPA or MIPROv2, it adds 1.48 points on average across 24 cells, with improvements in 19. The gains remain heterogeneous and are not guaranteed, but together these results establish component order as a distinct optimization axis that content optimization does not necessarily exhaust. For prompts with semantically movable components, ORDERCOVER provides a transparent, content-preserving post-processing step with no deployment-time model overhead.

AI USE STATEMENT

Generative AI tools were used to assist with code editing and debugging, and to improve the clarity and grammatical correctness of the manuscript. The research ideas, methodology, experimental design, and core implementation were developed by the authors. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content of this work.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd workshop on natural language processing for conversational AI*, pp. 38–45, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10136–10148, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Bryan Guan, Tanya Roosta, Peyman Passban, and Mehdi Rezagholizadeh. The order effect: Investigating prompt sensitivity to input order in llms. *arXiv preprint arXiv:2502.04134*, 2025.

Tianyi Huang, Nathan Huang, Justin Tang, Wenqian Chen, and Elsa Fan. Permutation-consensus listwise judging for robust factuality evaluation. In Simon Mille, Sebastian Gehrmann, Patrícia Schmidtová, Ondřej Dušek, Marzieh Fadaee, Kyle Lo, Enrico Santus, and Gabriel Stanovsky (eds.), *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 595–603, San Diego, California, USA, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-423-1. doi: 10.18653/v1/2026.gem-main.58. URL https://aclanthology.org/2026.gem-main.58/.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938. URL https://api.semanticscholar.org/CorpusID:120478295.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54928–54958, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f1cf02ce09757f57c3b93c0db83181e0-Paper-Conference.pdf.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Jerry Liu. LlamaIndex, 11 2022. URL https://github.com/jerryjliu/llama_index.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Blessings Mambwe. Prism: A predictive protocol for permutation optimization via landscape diagnostics, 2026. URL https://arxiv.org/abs/2608.08344.

Junyu Mao, Stuart E Middleton, and Mahesan Niranjan. Do prompt positions really matter? In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4102–4130, 2024.

Yuetian Mao, Junjie He, and Chunyang Chen. From prompts to templates: A systematic prompt template analysis for real-world llmapps. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pp. 75–86, 2025.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3845–3864, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.277. URL https://aclanthology.org/2023.eacl-main.277/.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023. URL https://arxiv.org/abs/2305.03495.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026b. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 2383–2392, 2016.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024.

Hui Shao, Yaping Wang, Dongdong Xiang, and Qian Xiao. Space-filling foldover designs for order-of-addition experiments under kendall tau distance criteria, 2026. URL https://arxiv.org/abs/2605.27248.

Maximilian Spliethöver, Tim Knebler, Fabian Fumagalli, Maximilian Muschalik, Barbara Hammer, Eyke Hüllermeier, and Henning Wachsmuth. Adaptive prompting: Ad-hoc prompt composition for social bias detection. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2421–2449, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.122. URL https://aclanthology.org/2025.naacl-long.122/.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims, 2020. URL https://arxiv.org/abs/2004.14974.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.340. URL https://aclanthology.org/2022.emnlp-main.340/.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Jinquan Zheng, Jia Yuan, Jiacheng Yao, Chenyang Gu, Pujun Zheng, and Guoxiu He. Mitigating selection bias in large language models via permutation-aware GRPO. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35125–35143, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1621. URL https://aclanthology.org/2026.acl-long.1621/.

Yujia Zheng, Tianhao Li, Haotian Huang, Tianyu Zeng, Jingyu Lu, Chuangxin Chu, Yuekai Huang, Ziyou Jiang, Qian Xiong, Yuyao Ge, et al. Are all prompt components value-neutral? understanding the heterogeneous adversarial robustness of dissected prompt in large language models. *arXiv preprint arXiv:2508.01554*, 2025.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts, 2024. URL https://arxiv.org/abs/2306.04528.

Type	Definition	Typical cues / examples
`Role`	Persona or identity specification for the model.	"You are a careful classifier." "Act as an expert tutor."
`InputDescription`	Text describing what input is provided or what fields the model will receive.	"You are given a question and four options." "Given a review sentence . . . "
`Directive`	The core instruction specifying what the model should do.	"Decide whether . . . " "Classify the statement as . . . "
`OutputFormatting`	Constraints on response form, schema, or allowed output tokens.	"Answer with only A, B, C, or D." "Return JSON with key `label`."
`LabelSemantics`	Definitions of what labels mean or when each label should be used.	"Label `positive` if the review expresses approval."
`Constraints`	Extra requirements or prohibitions beyond the core task.	"Use only the provided evidence."
`AuxiliaryHints`	Non-essential hints, reminders, or strategic advice.	"Think carefully about negation." "Pay attention to temporal order."
`Examples`	Worked demonstrations or illustrative input-output pairs.	Few-shot exemplars, demonstration blocks, or rubric examples.
`InstanceInput`	The task instance to be solved at inference time.	"Question: {{input}}" "Sentence: {{text}}"

Table 3: Operational component taxonomy used in the decomposition stage. Each prompt is segmented into a contiguous sequence of typed spans, and each span is assigned exactly one type. The taxonomy is designed to support faithful structuralization of benchmark-style prompts for grouping and permutation analysis.

A Prompt Decomposition and Grouping Details

This appendix provides implementation and reproducibility details that are intentionally abbreviated in Section 3.

A.1 Component Taxonomy and Decision Rules

Table 3 defines the nine span types used in the decomposition stage. We treat this inventory as a compact set of operational components for the benchmark-style prompts studied in this paper: it is adapted from Mao et al. (2025), but refined here to support type-pure segmentation, faithful grouping, and downstream order analysis.

A.2 Decomposition and Validation Protocol

The decomposition stage can be implemented with an LLM-assisted segmenter followed by deterministic validation. Throughout this paper, both LLMDecompose (Step 1 of this section) and LLMGroup (Section A.3) are instantiated with Llama-3.3-70B under deterministic decoding; this single choice is used for the structuralization pipeline and is independent of the evaluation models on which order search is run in Sections 6.1–6.3.

Step 1: Initial segmentation. Given a raw prompt, the segmenter outputs an ordered list of contiguous spans, each paired with one type from Table 3. The segmenter is instructed to prefer type-pure spans and to avoid inserting, deleting, or paraphrasing text.

Step 2: Structural validation. A deterministic checker verifies that (i) every span has a valid type from the taxonomy in Table 3, (ii) no span is fully contained inside another (nested duplicates are post-processed in Step 3), and (iii) concatenating the span texts in their proposed order reconstructs the original prompt exactly.

Step 3: Post-processing nested spans. If the validator detects a span whose character range is fully contained inside another span, the inner span is dropped before the sequence is passed to the grouping

stage. This is the only post-processing applied to the segmenter's output: text is never edited, span boundaries are never moved, and types are never relabelled. After dedup, the validator is reapplied; if type validity or exact-text reconstruction still fail, the prompt is flagged as "not decomposed" and excluded from order search rather than silently corrupted.

The exact LLMDECOMPOSE prompt is reproduced in A.4.

Empirical pass rate on the candidate pool. We ran the decomposition pipeline on all 49 tasks in our SNI candidate pool, including the 41 tasks ultimately not selected for evaluation. The LLM segmentation passed type validity and exact-text reconstruction on the first attempt for $49/49$ tasks, with no nested-span dedup required. Decomposition success was *not* a criterion for task selection: the eight evaluation tasks were chosen for structural diversity, and the rejected 41 tasks were dropped for redundancy or near-trivial order spaces, not for decomposition failure. The $49/49$ rate is therefore an untruncated population statistic over the full candidate pool, not a post-selection artifact.

We attribute this to two factors. First, the task pool consists of benchmark-style prompts with well-delimited type-pure spans that map cleanly onto the type taxonomy in Table 3. Second, the segmenter prompt explicitly forbids paraphrasing, content insertion, and content modification (A.4), so the model is steered toward exactly the conditions the validator checks. We do not interpret $49/49$ as evidence that the segmenter is universally clean: prompts with ambiguous boundaries, mixed-type spans, or unconventional formatting (interleaved instructions and demonstrations, non-Latin scripts, or markup-heavy inputs) may produce nested or overlapping segmentations and would require the dedup step above, or further protocol extensions.

A.3 COMPONENT GROUPING PROTOCOL

After decomposition, adjacent typed spans are bundled when they function operationally as a single candidate movement unit. In our implementation, the grouping stage follows three principles.

Contiguity. A group may contain only adjacent components in the original prompt. Non-contiguous spans are never merged.

Text preservation. Grouping changes only the granularity of movement. It never edits content, changes internal order, or duplicates text.

Operational cohesion. Adjacent components are grouped when separating them would create unnatural permutations or break a tightly coupled instruction block. Examples include a multi-line demonstration block whose example input and corresponding answer are initially segmented as separate spans, or a short label-definition block that is better treated as one unit during reordering.

In practice, the grouping stage is implemented with an LLM proposal followed by a deterministic check that the proposed groups are contiguous bundles of segmenter-output spans and that concatenating their texts in component order reconstructs the original prompt exactly. Each resulting group stores its member spans, raw text span, and a dominant type used only for later analysis. The exact grouping prompt is reproduced in A.4. On the same 49-task candidate pool, the grouping proposal passed these checks on the first attempt for all tasks; we treat this as evidence that the protocol is well matched to benchmark-style prompts, not as a universal guarantee.

A.4 EXACT LLM PROMPTS

For reproducibility, this subsection reproduces the exact prompts used by the LLM-assisted segmenter (LLMDECOMPOSE, A.2) and the LLM-assisted grouper (LLMGROUP, A.3). The decomposition prompt produces a JSON list of typed spans that is then handed to the deterministic validator described in A.2; the grouping prompt produces a JSON list of super-nodes that is checked for partition validity, contiguity (where applicable), and exact text preservation. Double braces (`{{ }}`) in the decomposition prompt escape literal braces inside a Python format-string template; the placeholders `{prompt_text}` and `{list of typed spans}` are filled in at call time.

LLMDECOMPOSE prompt.

```
You are a prompt decomposition model.

Goal:
Given a full prompt, split it into semantically meaningful components and assign exactly ONE
type to each component.
The output will be used to build a graph, so boundaries must be consistent and type-pure.

========================
COMPONENT TYPES
========================
1) Role: persona/identity only (e.g., "You are a ...", "Act as ...")
2) InputDescription: describes what is given, the task setup, topic, domain, or input
structure
3) Directive: tells the model what to do (e.g., decide, choose, classify, answer, generate)
4) OutputFormatting: specifies the required output form, allowed labels/tokens, or schema
5) LabelSemantics: defines what labels mean or when to use each label
6) Constraints: explicit requirements or prohibitions (e.g., "Do not...", "Only...",
"Must...")
7) AuxiliaryHints: advice, strategy, or hints (e.g., "Think step by step", "Consider...")
8) Examples: a completed example containing both input and a FILLED answer/label/output
9) InstanceInput: the actual input to be processed, including placeholders/slots; in few-shot
format, the final unfinished input with an EMPTY answer/label/output

========================
GLOBAL RULES
========================
- Assign every non-empty part of the prompt to exactly one component.
- Do not omit text.
- Do not create overlapping components.
- Preserve text exactly, including whitespace and newlines.
- Only Example and InstanceInput may contain mixed semantic subparts.
- All other components must be the smallest contiguous type-pure spans.

========================
IMPORTANT DISTINCTIONS
========================
- If a span matches input-setup language such as "you are given", "in this task", "you will
be shown", or describes the topic/domain/content presented, classify it as InputDescription,
not Role.
- If a span tells the model to perform an action, it is Directive.
- If a span specifies the answer form, labels, tokens, or schema, it is OutputFormatting.
- If a span explains what labels mean or when each label should be used, it is LabelSemantics.
- If a span contains placeholders or variables such as [[QUESTION_SLOT]], {{text}}, or their
immediate labels such as "Question:", classify it as InstanceInput.

========================
DECOMPOSITION RULES
========================

1) Extract Example and InstanceInput first when they are clearly present.
- A completed record with both input and a FILLED answer/label/output is one Example.
- The final unfinished record with an EMPTY answer/label/output is one InstanceInput.
- In line-based few-shot prompts, do not merge multiple examples into one component.

2) Decompose the remaining header left-to-right into the smallest type-pure spans.
- Split whenever the function changes.
- If one sentence or line contains multiple functions, split it into separate components.
- Example: "Think step by step and then finish your answer with 'the answer is (X)'" must be
split into:
  - AuxiliaryHints: "Think step by step and then "
  - OutputFormatting: "finish your answer with 'the answer is (X)'"

3) Use these cues:
- Role cues: "You are...", "Act as..."
- InputDescription cues: "You are given...", "The following are...", "In this task..."
- Directive cues: "Your task is...", "Your job is...", "Decide...", "Choose...",
"Classify...", "Answer...", "Generate..."
- OutputFormatting cues: "Answer with only...", "Return only...", "Respond with...", "Respond
in JSON...", "Indicate your answer by...", "finish your answer with..."
- LabelSemantics cues: "X means...", "Answer X if ... otherwise Y", "A <label> is..."

========================
ID RULES
========================
Use ids in appearance order without skipping numbers:
role1, inputDesc1, directive1, outputFmt1, labelSem1, constraint1, hint1, tool1, sys1,
example1..N, instanceInput1

========================
OUTPUT FORMAT
```

```
========================
Return ONLY this JSON object:
{{
  "components": [
    {{"id":"...","type":"...","text":"..."}}
  ]
}}

Rules:
- "text" must be an exact substring from the prompt.
- Preserve whitespace/newlines exactly. Use "\n" inside JSON strings.
- The first character of your response must be "{{"
- The last character of your response must be "}}"

========================
NOW DO THE TASK
========================
Given the following full prompt, return ONLY the JSON object.

<PROMPT>
{prompt_text}
</PROMPT>
```

LLMGROUP prompt.

```
You are a STRICT clustering algorithm for prompt components into super-nodes.
Your output will be programmatically VERIFIED. If you violate rules, it will be rejected.

## Goal
Cluster components into semantic-role super-nodes for fault-budget evaluation.
Do NOT minimize node count aggressively. Prefer meaningful semantic blocks.

## HARD RULES (MUST NEVER VIOLATE)
H1) InstanceInput isolation:
- Any component with type == "InstanceInput" MUST be alone in its own super-node named
"InstanceInput".
- It MUST NOT be merged with anything else.
H2) Example-only grouping:
- Components with type == "Examples" may ONLY be grouped with other "Example" components.
- Prefer merging ALL Example components into ONE super-node named "Examples".
- Examples MUST NOT be merged with any non-Example component.
H3) Partition:
- Every input component cid must appear in exactly one super-node.
- Do not drop or duplicate cids.

## RISK-AWARE RULE
- Prefer grouping components with the SAME type.
- Avoid mixing very different prompt component types in one super-node.

## REQUIRED semantic-role super-nodes (create these when the types exist)
Create the following super-nodes when corresponding component types are present.
Names must match exactly these tokens:
- "Directive"      : contains Directive components (main instruction/goal)
- "LabelSemantics" : contains LabelSemantics components (label mapping/semantics)
- "Constraints"    : contains Constraints components (requirements/allowed outputs)
- "OutputFormatting": contains OutputFormatting components (output schema/format)
- "InputDescription": contains InputDescription components (input definition)
- "AuxiliaryHints" : contains AuxiliaryHints components (optional guidance)
- "Examples"       : all Example components (ONLY Examples)
- "InstanceInput"  : the single InstanceInput component (alone)
- "Role"           : contains Role components (persona/identity)

## Assignment guidance
- If multiple components of the same role exist, put them in the same role node.
- Keep members minimal: do not add unrelated types to a role node.

## OUTPUT (STRICT JSON ONLY)
Output ONLY JSON. No extra text.
Each super-node object MUST contain EXACTLY: gid, name, members, why.
- sid: G1, G2, ...
- name: must be one of: Directive, LabelSemantics, Constraints, OutputFormatting,
InputDescription, AuxiliaryHints, Examples, InstanceInput, or Role
- why: <= 60 characters, short phrase (no sentences)
{
    "super_nodes": [
        {"gid": "G1", "name": "Directive", "members": ["C_0"], "why": "directive"}
    ]
}
```

Task	Spans K	Groups M	Merged $K-M$	Has demos
MultiRC answerability	8	6	2	✓
CrowS-Pairs stereotype category	6	4	2	✓
StereoSet race relation	7	5	2	✓
Jigsaw sexually explicit content	7	5	2	✓
SBIC sexual-offense classification	7	5	2	✓
NLI entailment	7	5	2	✓
MathQA	6	4	2	✓
Math MCQA	6	4	2	✓
ARC-Challenge	5	5	0	
MMLU-Pro	4	4	0	

Table 4: Per-task counts before and after grouping. Human-readable names are used throughout the paper; Table 6 gives their SNI identifiers. K is the number of typed spans returned by LLMDECOMPOSE; M is the number of grouped components fed to ORDERCOVER; $K - M$ is the number of merged adjacent same-type spans. The eight SNI tasks include demonstrations and have $K - M = 2$; ARC-Challenge and MMLU-Pro use zero-shot prompts and have $K - M = 0$.

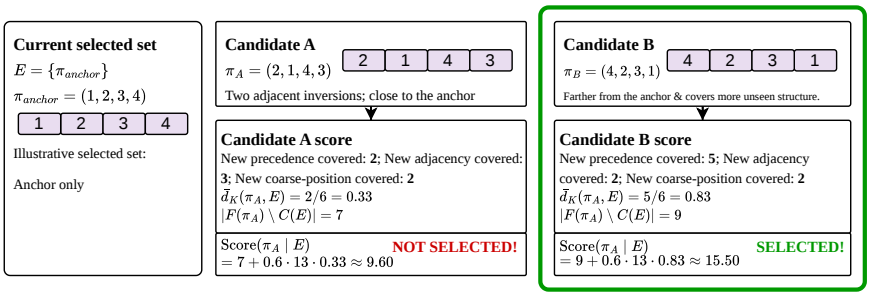


Figure 3: Illustrative greedy selection step on a 4-component example with λ=0.6 and $|F(\pi)|$=13. For clarity, the selected set is reduced to the Anchor alone; the full procedure first adds the deterministic seed set (Appendix B.2). Each candidate is scored by the new structural features it adds and its Kendall distance from already-selected orders. π_B is selected at this step; π_A remains in the pool and may be selected later.

```
## Components (ordered)
{list of typed spans}
```

Pre- and post-grouping counts. Table 4 reports, for each task, the number of typed spans K produced by LLMDECOMPOSE and the number of grouped components M produced by LLMGROUP. The pattern is consistent across the benchmark: whenever a task contains demonstrations, exactly one Examples block is merged ($K - M = 2$, namely three separate Example records merged into one Examples block); whenever a task does not contain demonstrations (ARC-Challenge and MMLU-Pro in our set), no merging occurs and $M = K$. The resulting order spaces Π_M contain $4! = 24$, $5! = 120$, or $6! = 720$ variants; all are enumerable, which is why ORDERCOVER runs on the full Π_M rather than a sampled candidate pool throughout the main paper.

B ADDITIONAL ORDERCOVER DETAILS

This appendix provides the formal notation and implementation details that support Section 4. The main text gives the reader-level description of what ORDERCOVER measures and how it selects the next order. Here we record the anchor-distance buckets, initialization details, and implementation notes in a compact form.

B.1 Position and Anchor-Distance Buckets

Coarse-position buckets. Let $r_\pi(i) = \pi^{-1}(i) - 1 \in \{0, \dots, M-1\}$ denote the zero-based position of component i in order π. We assign the component to position bucket

$$b_\pi(i) = \min \left\{ k \in \{1, \dots, A\} : \frac{r_\pi(i)}{M-1} \le \frac{k}{A} \right\}.$$

We set $A = 3$ throughout the experiments, giving coarse early, middle, and late regions of the component sequence. The value of A is fixed before any task–model evaluation.

Anchor-distance buckets. We use the normalized Kendall distance

$$\bar{d}_K(\pi, \rho) = \frac{d_K(\pi, \rho)}{\binom{M}{2}},$$

where $d_K(\pi, \rho)$ is the Kendall inversion distance. Thus, $\bar{d}_K(\pi, \rho) \in [0, 1]$ is the fraction of pairwise order relations on which the two permutations disagree.

To distribute the evaluation budget across both local and non-local variants, we partition candidates according to their distance from the Anchor order π_0. We use $D = 4$ fixed buckets:

$$\texttt{near} : [0, 0.10], \qquad \texttt{middle}_1 : (0.10, 0.35], \qquad \texttt{middle}_2 : (0.35, 0.65], \qquad \texttt{far} : (0.65, 1].$$

Given budget B, OrderCover allocates selections approximately evenly across non-empty buckets, with any remainder assigned preferentially to farther buckets. This balancing rule prevents the selected set from concentrating only near the Anchor while retaining both local and distant candidates.

B.2 Initialization and greedy selection

Let E denote the set of already-selected permutations. The structural patterns covered so far are

$$C(E) = \bigcup_{\rho \in E} F(\rho).$$

For a candidate permutation $\pi \notin E$, the greedy score is

$$\text{Score}(\pi \mid E) = |F(\pi) \setminus C(E)| + \lambda |F(\pi)| \min_{\rho \in E} \bar{d}_K(\pi, \rho),$$

where $\lambda = 0.6$ in all experiments.

The search is initialized with a small deterministic seed set consisting of the anchor order, all adjacent-swap neighbors of the anchor, up to four single-move neighbors, and the reverse order when available. These seeds ensure that the search begins with both local and distant structural variants before greedy selection continues. When scores tie, we break ties deterministically by permutation index.

B.3 Implementation notes

In the controlled suite, all tasks have $M \le 6$ and $\Pi(p) = \mathfrak{S}_M$, so the full valid order space is enumerable and we apply OrderCover directly to it rather than to a sampled candidate pool. Feature sets and rank maps can be precomputed for all permutations, making the structural overhead of OrderCover negligible relative to model inference.

For larger M, where exhaustive enumeration is infeasible, the same scoring rule applies unchanged to any finite candidate pool $\widehat{\Pi} \subseteq \Pi(p)$. The pool can be constructed by sampling globally and around the initial seeds, after which the same greedy selection is applied under budget $B \le |\widehat{\Pi}|$. The selector is independent of how the candidate pool is constructed.

B.4 Sensitivity to the Diversity Weight λ

We sweep $\lambda \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ at the default budget $B = M^2$ on the five-model subset of the controlled suite (Llama-3.1-8B, Llama-3.3-70B, Ministral-3-8B, Qwen3.5-9B, and Qwen3.5-35B;

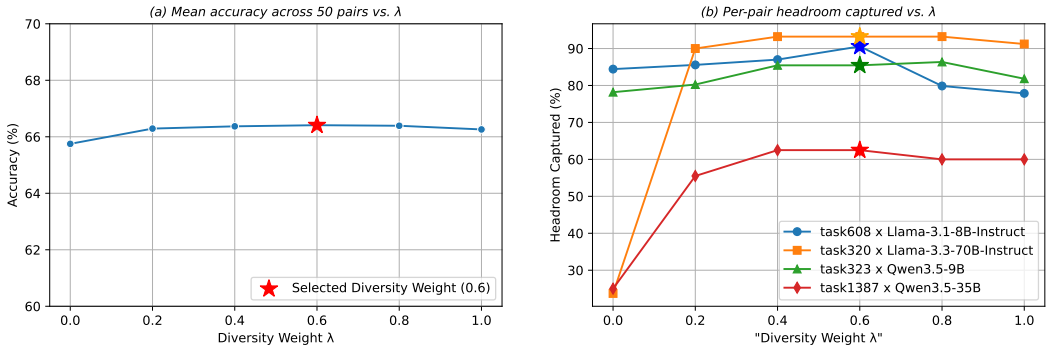


Figure 4: Sensitivity of ORDERCOVER to the diversity weight λ at the default budget $B{=}M^2$, on the five-model (50-pair) subset of the controlled suite. **(a)** Mean accuracy across the 50 pairs; the star marks the selected value $\lambda{=}0.6$. **(b)** Per-pair headroom-captured curves for four representative pairs spanning the gain spectrum. Performance is approximately flat for $\lambda \in [0.2, 1.0]$, with a shallow maximum at $\lambda{=}0.6$; removing the separation term ($\lambda{=}0$) markedly lowers headroom capture for individual pairs.

Method	Cov.	Kend.	Buckets	Avg. acc.	Gain	HC	W/T/L vs. OC
Random	–	–	–	66.92%	+2.89	71.01%	11/6/53
CoverageOnly	✓	–	–	67.05%	+3.02	76.07%	11/38/21
Kendall-FF	–	✓	–	67.30%	+3.27	82.37%	13/41/16
ORDERCOVER-noBucket	✓	✓	–	67.27%	+3.24	81.61%	12/40/18
ORDERCOVER	✓	✓	✓	67.38%	+3.35	82.99%	–
Oracle	–	–	–	68.00%	+3.97	100.00%	41/29/0

Table 5: Selector ablation on the 70-pair controlled suite. Cov. denotes coverage of precedence, adjacency, and coarse-position features; Kend. denotes nearest-neighbour Kendall separation; Buckets denotes balanced allocation across Anchor-distance buckets, distinct from the coarse-position features. Gain is the mean improvement over Anchor in percentage points. HC is mean Anchor-to-Oracle headroom captured over the 66 pairs with non-zero headroom. W/T/L compares the method in each row against the full ORDERCOVER selector.

50 task–model pairs), before Gemma-4-31B and gpt-oss-120b were added. Figure 4(a) shows that mean accuracy is approximately flat for $\lambda \in [0.2, 1.0]$, with a shallow empirical maximum at $\lambda = 0.6$, whereas $\lambda = 0$ is lower. Figure 4(b) shows that this drop at $\lambda = 0$ can be large for individual pairs, indicating that the separation term matters most when it is removed entirely rather than in its precise weight. We therefore select $\lambda = 0.6$ and keep it fixed for all subsequent experiments. Because the sweep averages over the 50-pair subset, its absolute accuracies are not directly comparable with the 70-pair results in Tables 1 and 5. The 50 subset pairs are in-suite with respect to this hyperparameter; the 20 pairs involving Gemma-4-31B and gpt-oss-120b, as well as the realistic and post-optimization suites, involve no tuning of λ.

B.5 SELECTOR ABLATION

We ablate structural coverage, nearest-neighbour Kendall separation, and balanced allocation across Anchor-distance buckets on the same 70 task–model pairs as the controlled evaluation. All selectors use the same order spaces and evaluation budget $B = M^2$.

The full selector has the highest mean accuracy and headroom capture among the budgeted methods. Its gains over Kendall-FF and the unbucketed hybrid are 0.08 and 0.11 percentage points, respectively, with many pair-level ties. These ablations support a modest aggregate benefit from combining the selection components; they do not establish that every component is necessary on every task–model pair.

Readable name	Source ID	Task form	M	Metric
MultiRC answerability	SNI `task050`	Binary contextual answerability	6	Accuracy
CrowS-Pairs stereotype category	SNI `task317`	Protected-attribute category classification	4	Accuracy
StereoSet race relation	SNI `task320`	Stereotype / anti-stereotype / unrelated	5	Accuracy
Jigsaw sexually explicit content	SNI `task323`	Binary content classification	5	Accuracy
SBIC sexual-offense classification	SNI `task608`	Binary safety-related text classification	5	Accuracy
NLI entailment	SNI `task1387`	Entailment / contradiction / neutral	5	Accuracy
MathQA	SNI `task1419`	Five-option quantitative reasoning	4	Accuracy
Math MCQA	SNI `task1678`	Five-option quantitative reasoning	4	Accuracy
ARC-Challenge	—	Multiple-choice science reasoning	5	Accuracy
MMLU-Pro	—	Multiple-choice multidisciplinary reasoning	4	Accuracy

Table 6: Task map for the controlled suite. M is the number of components.

B.6 Objective Grounding

The coverage term in OrderCover has a standard set-coverage interpretation. For a selected set of orders E, define

$$g(E) = \left| \bigcup_{\pi \in E} F(\pi) \right|.$$

This function is monotone because adding an order cannot remove an already covered feature. It is also submodular: as the selected set grows, the number of previously unseen features contributed by an additional order cannot increase. Therefore, if selection optimized only $g(E)$ under a cardinality budget, greedy maximization would inherit the standard guarantee for monotone submodular coverage.

The complete OrderCover procedure is not pure coverage maximization. It additionally rewards nearest-neighbour Kendall separation and balances selections across Anchor-distance buckets. Consequently, the submodularity of g does not imply an approximation guarantee for the full hybrid procedure, and we make no such claim. The three feature families should likewise be understood as model-independent descriptors of global, local, and positional structure rather than sufficient statistics for model accuracy. The effectiveness of the complete selection rule is evaluated empirically against budget-matched Random search in Section 6.1.

C Complete Experimental Details

C.1 Controlled Task Map

We use descriptive task names in the main text and figures. Table 6 maps each name to its stable source identifier and records the controlled component count. The identifiers are retained in this appendix, released artifacts, and machine-readable result files, but are not used as stand-alone task names in the main paper.

The seven evaluation models are Llama-3.1-8B, Llama-3.3-70B, Ministral-3-8B, Qwen3.5-9B, Qwen3.5-35B, Gemma-4-31B, and gpt-oss-120b, yielding 70 task–model pairs.

Task	SNI ID	Task form	Train	Dev	Test
MultiRC answerability	`task050`	Binary answerability	50	100	200
StereoSet race relation	`task320`	Three-way stereotype classification	50	100	200
NLI entailment	`task1387`	Natural-language inference	50	100	200
Math MCQA	`task1678`	Mathematical multiple-choice QA	50	100	200

Table 7: Frozen SNI splits for the post-optimization suite. Every task is evaluated with three models and two optimizers, yielding 24 cells. The development split is used by the optimizer and subsequently for order selection; the test split remains held out.

C.2 Selection, Uncertainty, and Cost Reporting

The controlled suite is a complete-grid characterization: every valid order is evaluated on its fixed task–model evaluation set, and Oracle is the best-scoring order in that grid. The realistic and post-optimization suites instead construct candidates without test labels, select one order using development data, freeze it, and evaluate it on the disjoint test split. OrderCover uses only permutation structure to construct its candidate set; it does not access labels, predictions, logits, gradients, or scores before the development-selection stage.

Controlled-suite Random is summarized over 10,000 resamples; realistic-suite Random is averaged over its recorded seeds. The post-optimization suite has no separately reported Random arm. When optimizer proposals are stochastic, we report the optimizer seed and preserve the optimized prompt artifact.

For controlled-suite aggregate intervals, we use 100,000 two-way bootstrap replicates. Each replicate samples the ten tasks and seven models independently with replacement and averages the resulting crossed cells. This treats both task and model choice as sources of variation while preserving the paired method comparison within each sampled cell. For the realistic suite, we use 100,000 hierarchical paired-bootstrap replicates. Each replicate samples the five applications and three models independently with replacement, then resamples the 200 test examples within every selected cell using identical indices for Anchor, OrderCover, and all ten Random runs. Random is averaged over its ten runs within each replicate, and the method differences are averaged over the resulting 15 crossed cells. For post-optimization comparisons, we use 100,000 nonparametric bootstrap replicates over the 24 task–model–optimizer cells. Each replicate resamples the cell-level paired differences with replacement and averages them; these are not example-paired intervals. Each result table states its number of cells, seeds, valid templates, and audit failures.

For the controlled-suite Oracle v^*, headroom captured by selected variant v is

$$\mathrm{HC}(v, m, t) = 100\,\frac{S(v, m, t) - S(v_{\mathrm{anchor}}, m, t)}{S(v^*, m, t) - S(v_{\mathrm{anchor}}, m, t)}, \tag{7}$$

defined only when the denominator is positive. We separately report one-time structuralization and model-evaluation cost, as well as deployment-time overhead.

C.3 Post-optimization Evaluation on Frozen SNI Splits

The post-optimization suite crosses the four SNI tasks in Table 7 with Ministral-3-8B, Gemma-4-31B, and GPT-5.6 Terra, and with both GEPA and MIPROv2, yielding 24 cells. Each task uses a disjoint 50/100/200 train/development/test split.

For each task, GEPA or MIPROv2 compiles a task-specific DSPy classifier using the training set and development validation set. We save the optimized program, optimizer settings and seed, optimizer details, and the rendered optimized prompt. The explicit order-search path extracts `classifier.signature.instructions` from that frozen program. It splits the instruction string at Markdown headings or paragraph boundaries; long numbered sections may be split further, while all text within a resulting block is preserved. The original instance template and output schema are appended unchanged to every variant.

This is an instruction-order experiment, not a reordering of the complete DSPy program. In particular, when MIPROv2 supplies demonstrations, their placement is not changed by the explicit OrderCover

ID	Application	Original repository	Evaluation data / metric	M
T01120	Steamship	steamship-core/python-client	BANKING77 / label accuracy	4
T01055	LlamaIndex	run-llama/llama_index	SciFact / label accuracy	5
T01496	DataCopilot	Modulos/data_copilot	WikiSQL / SQLite execution accuracy	4
T00561	CASALIOY	su77ungr/CASALIOY	SQuAD / token F1	5
T01969	Indexify	diptanu/indexify	SQuAD / token F1	5

Table 8: Provenance and evaluation coverage for the realistic prompt suite. Repository links identify the original application sources; template IDs and exact prompt text are retained because repositories may later be renamed or removed. M is the number of components.

evaluation. For a frozen instruction prompt with M blocks, we form all $M!$ instruction permutations, select a structure-only set of $B = \min(M^2, M!)$ candidates (including the original instruction order), and choose its best member by development accuracy. The test set is evaluated only for the selected order. We report the original SNI Anchor, the optimizer's original instruction order, and the corresponding GEPA→ORDERCOVER or MIPROv2→ORDERCOVER arm. Per-run artifacts include the components, candidate indices, selected order, development variant summary, predictions, and result metadata.

We use these optimizers to study residual order headroom on a fixed componentized suite, not to reproduce their original benchmark tables.

C.4 REALISTIC APPLICATION-TEMPLATE SUITE

Table 8 lists five application prompts from the collection of Mao et al. (2025), including LlamaIndex (Liu, 2022). Every application is evaluated with Ministral-3-8B, Gemma-4-31B, and GPT-5.6 Terra. We use BANKING77 (Casanueva et al., 2020), SciFact (Wadden et al., 2020), WikiSQL (Zhong et al., 2017), and SQuAD (Rajpurkar et al., 2016) for task-appropriate held-out evaluation.

For each template, we score all $M!$ valid orders on the 100-example development set ($4! = 24$ or $5! = 120$) to support retrospective comparison. Each budgeted selector constructs $B = M^2$ candidates (16 or 25, including the Anchor) without model scores, then chooses the best order using only the development scores of those candidates. Thus B is the matched selection budget per run; collecting the full development grid requires $M!$ order evaluations. Only the order chosen by each run is evaluated on the disjoint test set. Random uses ten independently sampled candidate sets, while ORDERCOVER selects its set using structural coverage and Kendall separation. We report Anchor, Random, and ORDERCOVER, but no Oracle arm.

D FULL CONTROLLED-SUITE RESULTS

Every controlled task–model cell contains 1,000 examples. Table 9 reports the complete 10×7 grid at the default budget $B = M^2$. Random is the mean best-found accuracy over 10,000 uniformly resampled candidate sets of size B, sampled without replacement and always including Anchor. ORDERCOVER is deterministic for a fixed componentization and budget. Oracle is the maximum over the complete $M!$ order grid. Scores are displayed to two decimal places, while all aggregates, headroom values, and win/tie/loss counts use the unrounded values.

Table 9: Complete controlled-suite results at $B = M^2$. ΔA and ΔR are ORDERCOVER's percentage-point differences from Anchor and Random, respectively.

Task	Model	Anchor	Random	OC	Oracle	ΔA	ΔR
MultiRC answerability	Llama-3.1-8B	73.40	75.19	75.80	76.80	+2.40	+0.61
MultiRC answerability	Ministral-3-8B	63.70	75.20	74.80	78.30	+11.10	-0.40
MultiRC answerability	Llama-3.3-70B	72.80	75.76	76.30	78.30	+3.50	+0.54
MultiRC answerability	Qwen3.5-9B	71.10	73.65	74.30	75.30	+3.20	+0.65
MultiRC answerability	Qwen3.5-35B	69.20	69.20	69.20	71.30	0.00	0.00
MultiRC answerability	Gemma-4-31B	71.20	74.67	75.60	78.10	+4.40	+0.93
MultiRC answerability	gpt-oss-120b	62.90	64.50	65.90	69.40	+3.00	+1.40

Continued on next page

Table 9 continued

Task	Model	Anchor	Random	OC	Oracle	△A	△R
CrowS-Pairs stereotype category	Llama-3.1-8B	73.00	74.89	75.50	75.50	+2.50	+0.61
CrowS-Pairs stereotype category	Ministral-3-8B	71.80	77.36	78.40	78.40	+6.60	+1.04
CrowS-Pairs stereotype category	Llama-3.3-70B	78.20	80.50	80.80	80.80	+2.60	+0.30
CrowS-Pairs stereotype category	Qwen3.5-9B	71.90	74.50	74.70	74.70	+2.80	+0.20
CrowS-Pairs stereotype category	Qwen3.5-35B	73.10	74.90	75.20	75.20	+2.10	+0.30
CrowS-Pairs stereotype category	Gemma-4-31B	82.30	84.74	84.80	84.80	+2.50	+0.06
CrowS-Pairs stereotype category	gpt-oss-120b	70.90	76.33	78.10	78.40	+7.20	+1.78
StereoSet race relation	Llama-3.1-8B	66.40	71.53	72.10	72.60	+5.70	+0.57
StereoSet race relation	Ministral-3-8B	66.20	74.02	75.10	75.10	+8.90	+1.08
StereoSet race relation	Llama-3.3-70B	81.00	84.83	86.50	86.90	+5.50	+1.67
StereoSet race relation	Qwen3.5-9B	69.50	69.50	69.50	69.50	0.00	0.00
StereoSet race relation	Qwen3.5-35B	66.60	71.65	72.30	73.20	+5.70	+0.65
StereoSet race relation	Gemma-4-31B	78.80	85.04	85.60	86.20	+6.80	+0.56
StereoSet race relation	gpt-oss-120b	54.10	62.06	63.40	66.50	+9.30	+1.34
Jigsaw sexually explicit content	Llama-3.1-8B	81.50	82.00	81.70	82.40	+0.20	-0.30
Jigsaw sexually explicit content	Ministral-3-8B	80.10	83.19	84.20	85.30	+4.10	+1.01
Jigsaw sexually explicit content	Llama-3.3-70B	81.70	82.70	83.00	83.00	+1.30	+0.30
Jigsaw sexually explicit content	Qwen3.5-9B	67.40	74.42	76.80	78.40	+9.40	+2.38
Jigsaw sexually explicit content	Qwen3.5-35B	70.10	70.90	72.00	72.00	+1.90	+1.10
Jigsaw sexually explicit content	Gemma-4-31B	79.50	80.72	81.30	81.30	+1.80	+0.58
Jigsaw sexually explicit content	gpt-oss-120b	60.50	69.91	73.80	76.00	+13.30	+3.89
SBIC sexual-offense classification	Llama-3.1-8B	59.60	80.77	81.70	84.00	+22.10	+0.93
SBIC sexual-offense classification	Ministral-3-8B	72.10	75.79	76.40	76.90	+4.30	+0.61
SBIC sexual-offense classification	Llama-3.3-70B	84.80	85.80	85.30	87.10	+0.50	-0.50
SBIC sexual-offense classification	Qwen3.5-9B	67.10	76.15	77.00	78.40	+9.90	+0.85
SBIC sexual-offense classification	Qwen3.5-35B	80.70	81.90	81.60	82.40	+0.90	-0.30
SBIC sexual-offense classification	Gemma-4-31B	86.80	87.59	87.70	87.90	+0.90	+0.11
SBIC sexual-offense classification	gpt-oss-120b	76.90	77.90	78.90	79.20	+2.00	+1.00
NLI entailment	Llama-3.1-8B	50.60	53.10	53.40	53.40	+2.80	+0.30
NLI entailment	Ministral-3-8B	60.00	60.90	61.20	61.30	+1.20	+0.30
NLI entailment	Llama-3.3-70B	66.20	67.90	68.20	68.60	+2.00	+0.30
NLI entailment	Qwen3.5-9B	56.10	60.37	61.20	61.80	+5.10	+0.83
NLI entailment	Qwen3.5-35B	65.60	66.18	66.60	67.20	+1.00	+0.42
NLI entailment	Gemma-4-31B	71.30	72.66	72.40	73.20	+1.10	-0.26
NLI entailment	gpt-oss-120b	52.60	52.60	52.60	52.60	0.00	0.00
MathQA	Llama-3.1-8B	22.80	25.53	26.10	26.10	+3.30	+0.57
MathQA	Ministral-3-8B	35.10	36.83	37.40	38.30	+2.30	+0.57
MathQA	Llama-3.3-70B	43.10	43.70	44.00	44.00	+0.90	+0.30
MathQA	Qwen3.5-9B	34.80	36.00	35.30	37.20	+0.50	-0.70
MathQA	Qwen3.5-35B	44.50	46.20	46.50	46.50	+2.00	+0.30
MathQA	Gemma-4-31B	64.30	64.50	64.60	64.60	+0.30	+0.10
MathQA	gpt-oss-120b	25.80	27.85	27.90	28.00	+2.10	+0.05
Math MCQA	Llama-3.1-8B	21.10	27.99	28.50	28.50	+7.40	+0.51
Math MCQA	Ministral-3-8B	24.20	29.92	30.30	30.30	+6.10	+0.38
Math MCQA	Llama-3.3-70B	41.90	44.55	45.00	45.00	+3.10	+0.45
Math MCQA	Qwen3.5-9B	35.90	38.10	37.00	39.10	+1.10	-1.10
Math MCQA	Qwen3.5-35B	44.40	46.00	45.90	46.00	+1.50	-0.10
Math MCQA	Gemma-4-31B	64.20	64.73	65.30	65.60	+1.10	+0.57
Math MCQA	gpt-oss-120b	27.30	30.63	31.10	31.10	+3.80	+0.47
ARC-Challenge	Llama-3.1-8B	80.50	82.20	82.50	82.90	+2.00	+0.30
ARC-Challenge	Ministral-3-8B	90.50	91.32	91.60	91.70	+1.10	+0.28
ARC-Challenge	Llama-3.3-70B	93.60	93.93	93.90	94.00	+0.30	-0.03
ARC-Challenge	Qwen3.5-9B	94.00	94.25	94.60	94.70	+0.60	+0.35
ARC-Challenge	Qwen3.5-35B	96.00	96.50	96.40	96.50	+0.40	-0.10
ARC-Challenge	Gemma-4-31B	98.20	98.20	98.20	98.20	0.00	0.00
ARC-Challenge	gpt-oss-120b	91.00	91.78	91.60	91.90	+0.60	-0.18
MMLU-Pro	Llama-3.1-8B	31.30	32.77	33.10	33.10	+1.80	+0.33
MMLU-Pro	Ministral-3-8B	43.90	45.36	45.70	45.70	+1.80	+0.34
MMLU-Pro	Llama-3.3-70B	51.20	51.20	51.20	51.20	0.00	0.00
MMLU-Pro	Qwen3.5-9B	52.40	53.38	53.70	53.70	+1.30	+0.32
MMLU-Pro	Qwen3.5-35B	60.80	61.00	61.00	61.00	+0.20	0.00
MMLU-Pro	Gemma-4-31B	63.40	69.46	69.90	69.90	+6.50	+0.44
MMLU-Pro	gpt-oss-120b	46.70	47.16	47.50	47.50	+0.80	+0.34

D.1 TASK- AND MODEL-LEVEL RECURRENCE

Tables 10 and 11 summarize the same pair-level results along each axis. Every model and every task has a positive mean gain over both Anchor and Random, but the magnitude is not uniform. Llama-3.1-8B has the largest model-level mean gain over Anchor, while gpt-oss-120b has the largest

Model	Mean ΔA	Mean ΔR	Max ΔA	W/T/L	O match
Llama-3.1-8B	5.02	0.44	22.10	9/0/1	5
Ministral-3-8B	4.75	0.52	11.10	9/0/1	4
Llama-3.3-70B	1.97	0.33	5.50	7/1/2	5
Qwen3.5-9B	3.39	0.38	9.90	7/1/2	3
Qwen3.5-35B	1.57	0.23	5.70	5/2/3	4
Gemma-4-31B	2.54	0.31	6.80	8/1/1	5
gpt-oss-120b	4.21	1.01	13.30	8/1/1	3

Table 10: Model-level controlled-suite summary. W/T/L compares ORDERCOVER with Random using unrounded scores.

Task	Mean ΔA	Mean ΔR	Max ΔA	W/T/L
MultiRC answerability	3.94	0.53	11.10	5/1/1
CrowS-Pairs stereotype category	3.76	0.61	7.20	7/0/0
StereoSet race relation	5.99	0.84	9.30	6/1/0
Jigsaw sexually explicit content	4.57	1.28	13.30	6/0/1
SBIC sexual-offense classification	5.80	0.39	22.10	5/0/2
NLI entailment	1.89	0.27	5.10	5/1/1
MathQA	1.63	0.17	3.30	6/0/1
Math MCQA	3.44	0.17	7.40	5/0/2
ARC-Challenge	0.71	0.09	2.00	3/1/3
MMLU-Pro	1.77	0.25	6.50	5/2/0

Table 11: Task-level controlled-suite summary across seven models. W/T/L compares ORDERCOVER with Random using unrounded scores.

mean advantage over Random. At the task level, StereoSet race relation has the largest mean gain over Anchor, while Jigsaw sexually explicit content has the largest mean advantage over Random. ARC-Challenge is near saturation for several models and correspondingly has the smallest average order gain.

D.2 AGGREGATE UNCERTAINTY AND ANCHOR QUALITY

The two-way bootstrap described in Appendix C.2 gives a 95% interval of $[2.35, 5.99]$ points for Oracle−Anchor, $[1.94, 5.11]$ for ORDERCOVER−Anchor, and $[0.21, 0.83]$ for ORDERCOVER−Random. These intervals quantify variation over both the task and model axes rather than only over the 70 observed cells.

We also test whether the available headroom is simply a consequence of weak Anchor performance. Across the 70 cells, Anchor accuracy has Pearson correlation $r = -0.11$ and Spearman correlation $\rho = -0.20$ with Oracle−Anchor headroom. The corresponding correlations with ORDERCOVER−Anchor are $r = -0.12$ and $\rho = -0.20$, while those with ORDERCOVER−Random are $r = 0.02$ and $\rho = -0.09$. These weak relationships do not imply that saturation is irrelevant, as ARC-Challenge contains several near-ceiling cells. They do show that low Anchor accuracy is not the sole source of the observed order opportunity. Because absolute accuracy mixes task and model difficulty with order quality, Appendix D.3 complements these correlations with each Anchor's within-landscape percentile.

D.3 PERMUTATION-LANDSCAPE POSITION AND ORACLE DISTANCE

For an order π in a complete landscape of size $N = |\Pi|$, let $H(\pi)$ be the number of orders scoring strictly higher and $T(\pi)$ the number tied with it, including π. We define the tie-aware rank and percentile as

$$R(\pi) = H(\pi) + \frac{T(\pi) + 1}{2}, \qquad P(\pi) = 100\frac{N - R(\pi) + 1/2}{N}. \tag{8}$$

Higher percentiles are better. This definition assigns all score-tied orders the midpoint of the ranks they occupy and is comparable across the 24-, 120-, and 720-order landscapes. We define weak,

Anchor tier	Pairs	Anchor pct.	OC pct.	ΔA	ΔR	W/T/L vs. Random
Weak	8	16.98	94.95	+4.43	+0.80	7/0/1
Medium	20	52.10	96.18	+5.46	+0.57	15/1/4
Strong	42	83.65	96.86	+2.14	+0.34	31/5/6
All	70	67.02	96.45	+3.35	+0.46	53/6/11

Table 12: Controlled results stratified by the Anchor order's tie-aware percentile in its complete permutation landscape. Pct. denotes mean landscape percentile; ΔA and ΔR are ORDERCOVER's mean percentage-point gains over Anchor and budget-matched Random. Tiers use fixed percentile boundaries rather than sample quantiles.

Anchor tier	Pairs	Anchor pct.	OC pct.	Δ pct.	d^*_A	d^*_{OC}	Δd^*	C/T/F
Weak	8	16.98	94.95	77.97	0.642	0.367	0.275	7/1/0
Medium	20	52.10	96.18	44.08	0.547	0.223	0.323	17/3/0
Strong	42	83.65	96.86	13.21	0.361	0.175	0.187	31/5/6
All	70	67.02	96.45	29.43	0.446	0.210	0.236	55/9/6

Table 13: Landscape position and minimum normalized Kendall distance to an accuracy-tied Oracle, grouped by Anchor tier. C/T/F counts pairs where ORDERCOVER is closer to, tied with, or farther from the nearest Oracle than Anchor.

medium, and strong Anchors using fixed percentile boundaries at 33.3 and 66.7, rather than empirical quantiles whose meaning would change with the evaluated sample.

For Oracle proximity, let $\mathcal{O}$ be the complete set of orders tied for the maximum score. We report

$$d^*(\pi) = \min_{\rho \in \mathcal{O}} \bar{d}_K(\pi, \rho), \tag{9}$$

where $\bar{d}_K$ is normalized Kendall distance. Taking the minimum over $\mathcal{O}$ avoids selecting an arbitrary representative when the landscape has multiple Oracle orders. Positive distance reduction $d^*(\pi^0) - d^*(\pi^{\mathrm{OC}})$ means that the selected order is closer to at least one accuracy-tied Oracle than Anchor.

The tier results separate starting-order quality from task accuracy. Weak and medium Anchors show the largest percentile movement, but strong Anchors also move from 83.65 to 96.86 on average. All six increases in Oracle distance occur in this strong group. The two-way bootstrap used for the main controlled comparisons gives a 95% interval of $[20.33, 40.25]$ for mean percentile improvement and $[0.145, 0.329]$ for mean normalized distance reduction.

For MMLU-Pro with Llama-3.3-70B, five of the 24 permutations share the maximum accuracy of 51.2%, including both Anchor and the ORDERCOVER-selected order. Both therefore receive tie-aware rank 3, percentile 89.58, and zero minimum distance to the accuracy-tied Oracle set.

Table 14 shows that selected-order percentile remains high as the evaluated share of the complete space decreases. At $M = 6$, ORDERCOVER evaluates 36 of 720 orders and reaches a mean percentile of 96.84. The remaining distance to an Oracle is larger than for $M = 4$ and $M = 5$, which is consistent with searching a smaller fraction of the largest space.

All six farther cases in Table 15 begin with strong Anchor orders and nevertheless improve over Anchor in both accuracy and landscape percentile; four also exceed Random. Performance is therefore not monotonic in Kendall distance. This is consistent with multiple high-performing regions and with Kendall distance weighting every pairwise inversion equally, irrespective of which component relations matter for a particular model. The converse boundary also occurs in the full grid: an unchanged selected score can indicate either that Anchor is already Oracle-tied or that better orders remain outside the budgeted candidate set. Moreover, ORDERCOVER falls below expected matched-budget Random in 11 pairs, only two of which are among the six farther cases. We therefore report score headroom, landscape rank, and Oracle distance as related but distinct diagnostics; proximity alone does not explain selector advantage over Random.

Finally, normalized distance reduction has Pearson correlation $r = 0.14$ and Spearman correlation $\rho = 0.19$ with absolute ORDERCOVER−Anchor gain. On the 66 pairs with defined positive

M	**Pairs**	$M!$	B	**Evaluated**	**Anchor pct.**	**OC pct.**	$d_A^* \to d_{OC}^*$
4	28	24	16	66.7%	66.29	95.39	$0.417 \to 0.119$
5	35	120	25	20.8%	67.49	97.21	$0.460 \to 0.251$
6	7	720	36	5.0%	67.56	96.84	$0.495 \to 0.371$

Table 14: Landscape results by component count. Evaluated is $B/M!$. The seven $M = 6$ cells are one task evaluated with all seven models, so this row is descriptive rather than a broad scaling result.

Task	**Model**	**Anchor pct.**	**OC pct.**	d_A^*	d_{OC}^*	**ΔA**	**ΔR**
MultiRC answerability	gpt-oss-120b	84.17	94.51	0.200	0.267	+3.00	+1.40
Math MCQA	Gemma-4-31B	79.17	93.75	0.333	0.500	+1.10	+0.57
SBIC sexual-offense classification	Gemma-4-31B	80.42	97.92	0.400	0.900	+0.90	+0.11
MathQA	Qwen3.5-9B	79.17	85.42	0.667	0.833	+0.50	-0.70
Jigsaw sexually explicit content	Ministral-3-8B	80.42	97.08	0.300	0.400	+4.10	+1.01
SBIC sexual-offense classification	Qwen3.5-35B	89.58	97.50	0.300	0.500	+0.90	-0.30

Table 15: The six pairs for which ORDERCOVER is farther from every accuracy-tied Oracle than Anchor. All six improve over Anchor in both accuracy and landscape percentile; four also exceed Random.

headroom, the corresponding correlations with headroom captured are $r = 0.36$ and $\rho = 0.42$. These associations support using Oracle distance as a descriptive landscape diagnostic, but not as a causal model or a substitute for evaluating candidate orders.

E REALISTIC PROMPT TEMPLATES

Table 16 gives the raw held-out scores underlying panel (a) of Table 2. Development and test sets contain 100 and 200 examples, respectively. Random is the mean test score of the orders selected by ten independently sampled, budget-matched candidate sets. All selection is performed on development data before the selected order is frozen for test evaluation.

E.1 OUTPUT VALIDITY

The saved final rows contain no backend failures. The selected QA, relevance, and SQL runs contain no parse failures; SQL execution failures are retained as incorrect predictions rather than excluded. For Steamship, the Ministral-3-8B ORDERCOVER predictions have a 1% parse-failure rate. With GPT-5.6 Terra, the corresponding rates are 1.5% for Anchor and ORDERCOVER and 0.8% for Random. These failures remain included in the reported accuracy, so no condition benefits from post-hoc filtering.

F CONTENT-OPTIMIZER INTEGRATION DETAILS

F.1 COMPLETE RESULTS

Table 17 gives all 24 cells in the complete post-optimization evaluation grid. All rows use 200 held-out test examples and contain no parse, format, or recorded evaluation failures.

Table 17: Post-optimization held-out results. Δ is Optimized+ORDERCOVER minus Optimized, in percentage points.

Task	**Model**	**Opt.**	M	B	**Anchor**	**Optimized**	**Opt.+OC**	Δ
MultiRC answerability	Ministral-3-8B	GEPA	5	25	64.5	71.0	74.0	+3.0
MultiRC answerability	Ministral-3-8B	MIPROv2	5	25	64.5	71.5	74.5	+3.0
MultiRC answerability	Gemma-4-31B	GEPA	5	25	74.5	83.5	84.0	+0.5
MultiRC answerability	Gemma-4-31B	MIPROv2	5	25	74.5	73.0	73.5	+0.5

Continued on next page

Table 17 continued

Task	Model	Opt.	M	B	Anchor	Optimized	Opt.+OC	Δ
MultiRC answerability	GPT-5.6 Terra	GEPA	5	25	75.5	83.0	84.0	+1.0
MultiRC answerability	GPT-5.6 Terra	MIPROv2	6	36	75.5	79.0	80.5	+1.5
StereoSet race relation	Ministral-3-8B	GEPA	6	36	68.0	66.5	71.5	+5.0
StereoSet race relation	Ministral-3-8B	MIPROv2	6	36	68.0	67.0	68.0	+1.0
StereoSet race relation	Gemma-4-31B	GEPA	6	36	79.0	85.5	85.5	+0.0
StereoSet race relation	Gemma-4-31B	MIPROv2	4	16	79.0	79.0	83.0	+4.0
StereoSet race relation	GPT-5.6 Terra	GEPA	6	36	83.5	82.5	84.0	+1.5
StereoSet race relation	GPT-5.6 Terra	MIPROv2	4	16	83.5	83.0	85.0	+2.0
NLI entailment	Ministral-3-8B	GEPA	4	16	59.5	59.5	59.5	+0.0
NLI entailment	Ministral-3-8B	MIPROv2	6	36	59.5	55.0	58.0	+3.0
NLI entailment	Gemma-4-31B	GEPA	5	25	71.0	72.0	73.5	+1.5
NLI entailment	Gemma-4-31B	MIPROv2	5	25	71.0	73.5	73.0	-0.5
NLI entailment	GPT-5.6 Terra	GEPA	6	36	69.5	70.5	71.0	+0.5
NLI entailment	GPT-5.6 Terra	MIPROv2	4	16	69.5	70.0	72.5	+2.5
Math MCQA	Ministral-3-8B	GEPA	6	36	26.5	39.0	41.5	+2.5
Math MCQA	Ministral-3-8B	MIPROv2	6	36	26.5	37.5	39.5	+2.0
Math MCQA	Gemma-4-31B	GEPA	6	36	67.0	68.0	68.5	+0.5
Math MCQA	Gemma-4-31B	MIPROv2	3	6	67.0	70.0	70.0	+0.0
Math MCQA	GPT-5.6 Terra	GEPA	6	36	73.0	78.0	78.0	+0.0
Math MCQA	GPT-5.6 Terra	MIPROv2	3	6	73.0	73.0	73.5	+0.5

Conditional analysis by optimizer outcome. Relative to the corresponding Anchor, the optimizer output has higher held-out accuracy in 15 cells, equal accuracy in three, and lower accuracy in six. In the 15 improving cells, subsequent ORDERCOVER selection adds 1.20 percentage points on average, with 11 improvements, three ties, and one loss. In the three tied cells, it adds 1.50 points on average, with two improvements and one tie. In the six cells where the optimizer falls below Anchor, ORDERCOVER improves all six optimizer outputs by 2.17 points on average and restores four to the Anchor score or above. The only decrease after order selection is -0.5 points, corresponding to one example on a 200-example test set. These groups are defined descriptively from the observed held-out outcomes and are not treated as prespecified statistical strata.

F.2 OPTIMIZER AND EVALUATION CONFIGURATION

GEPA uses the target model in each row as both task model and reflection/proposal model. We use `auto=medium`, one thread, seed 12345, reflection temperature 1.0 with an 8,192-token cap, task temperature 0 with a 64-token cap, `use_merge=false`, and disabled caching. The runner records a nominal `max_metric_calls=150`; under automatic medium mode, however, no explicit maximum is passed to GEPA, and its automatic budget schedule controls the search.

MIPROv2 likewise uses the row's target model as both task and proposal model. We use `auto=medium`, seed 12345, proposal temperature 1.0, initialization temperature 0.5, an 8,192-token proposal cap, and a 64-token task cap. Setting `num_trials` and `num_candidates` to zero delegates both quantities to the automatic schedule. Demonstrations are disabled: the maximum numbers of bootstrapped and labeled demonstrations are both zero. Minibatching uses 35 examples with a full evaluation every five steps. Program-, data-, tip-, and few-shot-aware proposal contexts are all enabled. Runs use one thread, allow at most ten errors, and disable caching.

For both optimizers, the optimizer and ORDERCOVER share the frozen 100-example development set, whereas the 200-example test set is untouched until the final order has been selected. Target evaluation uses temperature 0, a 64-token cap, no retries, and one evaluation thread. All optimizer and ORDERCOVER seeds are 12345. These choices condition the experiment on one saved prompt artifact per optimizer rather than estimating variation over optimizer seeds.

F.3 SELECTED ORDERS AND BUDGETS

Table 19 records the instruction blocks seen by ORDERCOVER. Orders use one-based indices relative to the original optimized instruction, whose order is $(1, \ldots, M)$. The selected development score is never below the original because the original optimized order is included in every candidate set. Complete component text and short component maps are retained in the released per-run artifacts.

Application	Model	Metric	Anchor	Random	OC	ΔA	ΔR
CASALIOY	Gemma-4-31B	F1	70.62	72.85	73.10	+2.48	+0.25
CASALIOY	Ministral-3-8B	F1	64.03	68.39	68.90	+4.87	+0.51
CASALIOY	GPT-5.6 Terra	F1	69.34	71.71	71.76	+2.42	+0.05
Indexify	Gemma-4-31B	F1	84.09	85.70	86.32	+2.23	+0.62
Indexify	Ministral-3-8B	F1	75.58	75.39	75.58	+0.00	+0.19
Indexify	GPT-5.6 Terra	F1	85.65	84.52	86.48	+0.83	+1.96
Steamship	Gemma-4-31B	Acc.	77.00	78.55	80.00	+3.00	+1.45
Steamship	Ministral-3-8B	Acc.	67.50	68.40	68.50	+1.00	+0.10
Steamship	GPT-5.6 Terra	Acc.	86.00	85.25	86.00	+0.00	+0.75
LlamaIndex	Gemma-4-31B	Acc.	82.00	83.55	83.50	+1.50	-0.05
LlamaIndex	Ministral-3-8B	Acc.	67.50	88.40	89.50	+22.00	+1.10
LlamaIndex	GPT-5.6 Terra	Acc.	84.00	86.35	87.50	+3.50	+1.15
DataCopilot	Gemma-4-31B	Exec.	64.00	62.00	63.50	-0.50	+1.50
DataCopilot	Ministral-3-8B	Exec.	52.00	57.50	58.00	+6.00	+0.50
DataCopilot	GPT-5.6 Terra	Exec.	64.00	62.00	64.50	+0.50	+2.50

Table 16: Complete realistic-suite results. Scores and differences are percentages or percentage points. OC denotes ORDERCOVER; ΔA and ΔR denote differences from Anchor and Random.

Slice	Cells	Anchor	Optimized	Opt.+OC	Δ [95% CI]
GEPA	12	67.63	71.58	72.92	$+1.33\ [+0.58, +2.21]$
MIPROv2	12	67.63	69.29	70.92	$+1.63\ [+0.88, +2.38]$
Ministral-3-8B	8	54.63	58.38	60.81	$+2.44\ [+1.50, +3.38]$
Gemma-4-31B	8	72.88	75.56	76.38	$+0.81\ [+0.06, +1.81]$
GPT-5.6 Terra	8	75.38	77.38	78.56	$+1.19\ [+0.63, +1.75]$
All cells	24	67.63	70.44	71.92	$+1.48\ [+0.94, +2.06]$

Table 18: Aggregate post-optimization accuracy and 95% nonparametric cell-bootstrap intervals. Intervals use 100,000 resamples of the completed cells; they are not example-paired intervals.

Table 19: Per-run component count, evaluation budget, selected order, and development accuracy before and after order selection.

Task	Model	Opt.	M	B	Selected order	Dev. orig.→sel.
MultiRC answerability	Ministral-3-8B	GEPA	5	25	$(1,2,3,5,4)$	.73 → .75
MultiRC answerability	Ministral-3-8B	MIPROv2	5	25	$(1,2,3,5,4)$	.73 → .75
MultiRC answerability	Gemma-4-31B	GEPA	5	25	$(1,2,4,5,3)$	.80 → .83
MultiRC answerability	Gemma-4-31B	MIPROv2	5	25	$(1,3,5,4,2)$	.75 → .76
MultiRC answerability	GPT-5.6 Terra	GEPA	5	25	$(2,4,1,3,5)$	.79 → .83
MultiRC answerability	GPT-5.6 Terra	MIPROv2	6	36	$(1,3,2,4,5,6)$	.73 → .76
StereoSet race relation	Ministral-3-8B	GEPA	6	36	$(1,2,3,4,6,5)$	.66 → .70
StereoSet race relation	Ministral-3-8B	MIPROv2	6	36	$(1,3,6,4,2,5)$	.64 → .68
StereoSet race relation	Gemma-4-31B	GEPA	6	36	$(1,2,3,5,4,6)$	.87 → .89
StereoSet race relation	Gemma-4-31B	MIPROv2	4	16	$(1,4,2,3)$	.83 → .88
StereoSet race relation	GPT-5.6 Terra	GEPA	6	36	$(1,3,5,2,6,4)$	.87 → .90
StereoSet race relation	GPT-5.6 Terra	MIPROv2	4	16	$(1,3,2,4)$	.86 → .91
NLI entailment	Ministral-3-8B	GEPA	4	16	$(1,4,2,3)$	.60 → .62
NLI entailment	Ministral-3-8B	MIPROv2	6	36	$(6,3,1,5,4,2)$	.54 → .63
NLI entailment	Gemma-4-31B	GEPA	5	25	$(1,4,5,2,3)$	.72 → .74
NLI entailment	Gemma-4-31B	MIPROv2	5	25	$(2,3,4,1,5)$	.70 → .72
NLI entailment	GPT-5.6 Terra	GEPA	6	36	$(5,2,6,3,4,1)$	.69 → .73
NLI entailment	GPT-5.6 Terra	MIPROv2	4	16	$(2,4,3,1)$	.70 → .73
Math MCQA	Ministral-3-8B	GEPA	6	36	$(1,4,6,2,5,3)$	.29 → .38
Math MCQA	Ministral-3-8B	MIPROv2	6	36	$(6,2,5,1,4,3)$	.32 → .36
Math MCQA	Gemma-4-31B	GEPA	6	36	$(1,4,6,2,5,3)$	.61 → .68
Math MCQA	Gemma-4-31B	MIPROv2	3	6	$(2,3,1)$	.66 → .70

Task	Model	Optimizer	M	B	Selected order	Dev. orig.→sel.
Math MCQA	GPT-5.6 Terra	GEPA	6	36	$(1, 2, 3, 4, 5, 6)$	.75 → .75
Math MCQA	GPT-5.6 Terra	MIPROv2	3	6	$(3, 1, 2)$	.71 → .78

All budgets follow $B = \min(M^2, M!)$.

G Additional Analyses, Robustness, and Failure Cases

G.1 Structuralization Sensitivity

This analysis separates two questions. The taxonomy comparison asks whether an external descriptive taxonomy and our operational taxonomy imply the same practical component-count regime. The decomposer comparison then holds the taxonomy and structuralization instructions fixed and changes only the model used to construct the grouped components. Our taxonomy adapts the functional categories of Mao et al. (2025) into exact-text movable units: it may split or merge descriptive roles when necessary to preserve literal text and form coherent reorderable components.

Alternative-taxonomy regime check. Table 20 compares component-count distributions on the deployed-template corpus. The external taxonomy is reported both without and with its residual `others` category. The latter is conservative because the residual bucket need not correspond to one coherent movable function. The OrderCover estimate uses a stratified 200-template sample and weights each stratum back to the corpus distribution. Six samples failed to parse and are excluded from the corresponding component-count estimates.

Alternative-decomposer stability. The frozen set contains the ten controlled prompts, five additional deployed QA/RAG prompts used in the stability audit, and 194 deployed templates from the successful OrderCover-taxonomy sample. The baseline reuses the saved Llama-3.3-70B structuralizations. Each alternative run keeps the raw prompt, decomposition and grouping instructions, taxonomy, and validators fixed. Table 21 reports exact component-count agreement, tolerance within one or two components, agreement on the binary $M \leq 6$ regime, and role-set Jaccard similarity.

Disagreement audit. All crossings of the $M \leq 6$ boundary occur in the 194-template deployed sample: 12 for gpt-oss-120b and seven for Qwen3.6-35B. Large count changes mostly reflect granularity. For example, one prompt changes from $M = 4$ to $M = 8$ because the alternative model separates five input placeholders, while preserving the same role set. Qwen3.6-35B is systematically more compressive in the largest disagreements: all 18 cases with $|\Delta M| > 2$ have negative ΔM. One controlled prompt is severely under-segmented by Qwen ($M = 5 \to 1$), although the other alternative recovers $M = 5$ and both remain within the enumerable regime. Finally, the sole gpt-oss-120b parse failure contains only model reasoning rather than a valid structuralization; we count it as an operational failure. These cases motivate the calibrated claim in Section 7.2: the coarse functional regime is stable, but exact boundaries and downstream selected permutations are not claimed to be invariant.

G.2 Cross-Model Transfer Protocol and Aggregate Results

We evaluate transfer only within the controlled suite, where every task has a complete permutation landscape for all seven models. Before any lookup, we verify for each task that all seven landscapes contain exactly the same order universe and are aligned to the same 1,000 example identifiers. Hence, a source order is located in a target landscape without changing the task, examples, component decomposition, or set of valid orders.

For OrderCover transfer, we use the source model's saved selected order. For the Oracle reference, we collect every source-landscape order tied for maximum source score and average the target score and percentile across that complete tied set. No target score or rank is used to choose among source ties. We report three target-side quantities. The first is the tie-aware target percentile defined in Appendix D.3. The second is transferred minus target Anchor, $s_t(\pi_s) - s_t(\pi_{A,t})$. The third is regret

Taxonomy pass	**Parsed**	**Mean M**	**Median M**	**P90 / P99**	$M \leq 6$
Mao/FSE, excluding `others`	2,149/2,163	3.78	4	6 / 7	92.88%
Mao/FSE, including `others`	2,149/2,163	4.29	4	7 / 8	84.41%
OrderCover, weighted corpus estimate	194/200	3.23	3	5 / 7	97.06%

Table 20: Component-count regime under an external descriptive taxonomy and the OrderCover operational taxonomy. Exact counts differ, but both taxonomies place the large majority of deployed prompts in or below the $M \leq 6$ regime.

Alternative model	**Prompt subset**	**Parsed**	**Exact M**	$\lvert\Delta M\rvert \leq 1$	$\lvert\Delta M\rvert \leq 2$	$M \leq 6$ **agr.**	**Role Jac.**
gpt-oss-120b	All prompts	208/209	41.3%	83.7%	96.6%	93.8%	0.675
Qwen3.6-35B	All prompts	209/209	42.1%	77.0%	91.4%	96.7%	0.652
gpt-oss-120b	Ten controlled prompts	10/10	70.0%	100.0%	100.0%	100.0%	0.849
Qwen3.6-35B	Ten controlled prompts	10/10	40.0%	80.0%	90.0%	100.0%	0.721
gpt-oss-120b	Five QA/RAG audit prompts	5/5	0.0%	80.0%	100.0%	100.0%	0.470
Qwen3.6-35B	Five QA/RAG audit prompts	5/5	20.0%	80.0%	80.0%	100.0%	0.480

Table 21: Structuralization agreement with the Llama-3.3-70B baseline. "$M \leq 6$ agr." is agreement on whether the resulting order space lies in the studied enumerable regime; "Role Jac." is mean role-set Jaccard similarity.

relative to target-specific ORDERCOVER, $s_t(\pi_{\mathrm{OC},t}) - s_t(\pi_s)$; positive regret therefore means that transfer trails target-specific selection.

Across the 42 off-diagonal directed model pairs (420 task-level transfers), the transferred ORDERCOVER orders attain a mean target-landscape percentile of 60.71, and 40 of 42 directed cells remain above the target landscape midpoint (Figure 5). Thus, selected orders exhibit a recurring cross-model rank signal rather than behaving like arbitrary permutations.

This rank signal does not make an order directly reusable without target-model selection. The transferred ORDERCOVER orders fall 1.64 percentage points below the target Anchor on average and trail target-specific ORDERCOVER by 4.99 points; only 8 of 42 directed cells exceed the target Anchor in mean accuracy. Transferring source-Oracle orders raises the mean target percentile only to 62.14, while remaining 1.53 points below the target Anchor and 4.88 points behind target-specific ORDERCOVER. Transfer is also directional, as the asymmetric cells in the heatmaps show.

The source-Oracle reference improves mean target percentile by only 1.44 points over transferring the source ORDERCOVER order. It ranks higher in 25 of the 42 directed cells and lower in 17, confirming that source optimality does not induce a uniformly stronger target order. In accuracy terms, both transfer rules remain below the target Anchor and target-specific ORDERCOVER on average. These results distinguish rank recurrence from performance portability: an order that is unusually strong for one model often remains above median for another, but selecting on the target model is still necessary for reliable improvement.

G.3 ROBUSTNESS OF CLEAN-SELECTED ORDERS

This supplementary analysis reuses the robustness experiment from the five-model version of the controlled suite. It covers the same ten tasks and Llama-3.1-8B, Llama-3.3-70B, Ministral-3-8B, Qwen3.5-9B, and Qwen3.5-35B, yielding 50 task–model pairs.

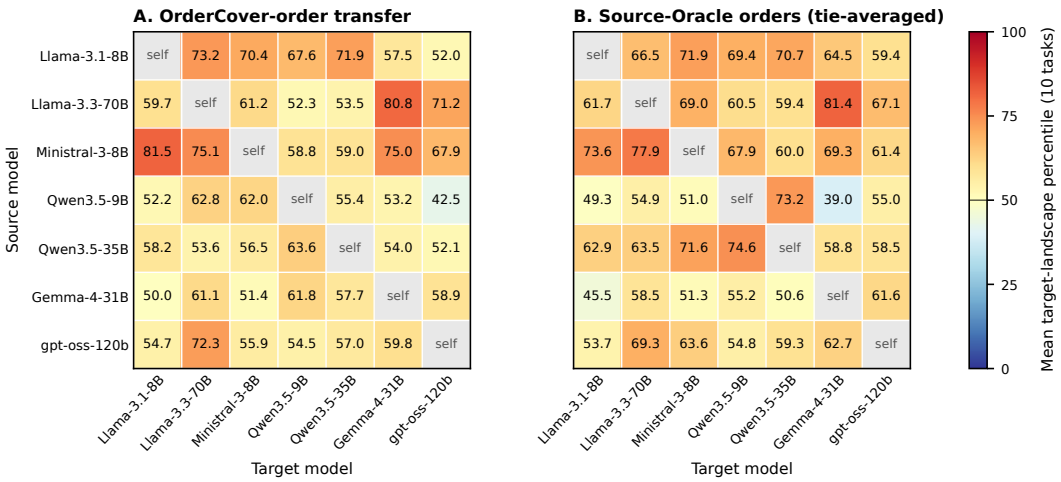


Figure 5: Same-task cross-model transfer of prompt orders. Rows denote source models and columns denote target models; diagonal cells are marked "self" and excluded from transfer summaries. Each off-diagonal cell is the mean tie-aware percentile in the target model's complete landscape across ten tasks, with 50 denoting the landscape midpoint. **(a)** Transfer of the source model's saved OrderCover order. **(b)** Transfer of all source-Oracle-tied orders, averaging their target percentiles without target-based tie-breaking. Most transferred orders remain above the target median, but this rank-level transfer does not imply competitive target-model accuracy.

Transferred source order	**Pairs**	**Target pct.**	**Δ target Anchor**	**Regret vs. target OC**	**Pct. > 50**
OrderCover order	42	60.71	−1.64	4.99	40/42
Source-Oracle orders (tie-averaged)	42	62.14	−1.53	4.88	39/42

Table 22: Aggregate same-task cross-model transfer over off-diagonal directed model pairs. Target pct. is the mean target-landscape percentile across pairs and tasks. Accuracy differences and regret are in percentage points. The diagonal is excluded because it is self-selection rather than transfer.

G.3.1 Fixed Variants and Perturbation Scope

For each task–model pair, we carry forward one fixed clean-selected variant from Anchor, OrderCover, and clean Oracle. We also retain the concrete Random comparator used in the original robustness experiment: the highest-clean-scoring order found across five independently sampled budget-M^2 Random candidate sets. We denote this comparator **Random-5**. Unlike the controlled-suite Random summary, which averages best-found scores across 10,000 matched-budget resamples, Random-5 identifies a concrete order that can be perturbed. Because it effectively selects from $5M^2$ clean evaluations, this robustness comparison is conservative and is not a matched-budget replication of the controlled result.

All four variants are selected once using clean accuracy and then held fixed. We perturb these selected prompts without rerunning OrderCover, re-optimizing Oracle, or resampling Random. The question is therefore whether a clean-selected order remains effective under perturbation, not which order would be optimal for an attacked prompt.

We use four PromptRobust-style perturbation worlds (Zhu et al., 2024):

- **Character**: local insertion, deletion, replacement, repetition, or transposition of characters.
- **Word**: contextual replacement of words or short phrases.
- **Sentence**: insertion of locally distracting or extraneous sentence-level material.
- **Semantic**: larger-span re-expression that aims to preserve the component's intended meaning.

Method	Clean acc.	Attacked acc.	Robust KC
Anchor	63.08%	61.20%	91.77%
Random-5	65.87%	63.07%	91.68%
ORDERCOVER	66.40%	63.48%	91.21%
Clean Oracle	66.99%	63.70%	90.93%

Table 23: Supplementary robustness summary over the original 50 task–model pairs. Attacked accuracy and Robust KC average over the four perturbation worlds and all feasible fault budgets. Random-5 is the concrete clean-selected Random variant defined above; it is not the matched-budget Random expectation reported for the controlled suite.

Perturbations are applied locally to grouped prompt components rather than to the flattened prompt. Only prompt-side components whose text is fixed across evaluation instances are eligible. `InstanceInput`, the test instance, its answer candidates, and its gold label are not edited. Placeholders, field names, and dataset slot markers remain unchanged, and an operator may neither move text across component boundaries nor change the selected component order.

G.3.2 FAULT BUDGETS AND METRICS

The fault budget f is the number of eligible grouped components perturbed simultaneously. For a selected prompt variant v, let $\mathcal{G}_{\text{elig}}(v)$ be its eligible components and

$$\mathcal{S}_{v,f} = \{U \subseteq \mathcal{G}_{\text{elig}}(v) : |U| = f\}.$$

We evaluate only feasible budgets $f \in \{1, \ldots, |\mathcal{G}_{\text{elig}}(v)|\}$. For perturbation world w, attacked accuracy is

$$A_{\text{rob}}(v, m, t, w, f) = \frac{1}{|\mathcal{S}_{v,f}|} \sum_{U \in \mathcal{S}_{v,f}} S(\text{Attack}(v, U, w), m, t). \tag{10}$$

We additionally report absolute drop $\Delta_{\text{rob}} = S(v, m, t) - A_{\text{rob}}$ and performance-drop rate

$$\text{PDR}(v, m, t, w, f) = 1 - \frac{A_{\text{rob}}(v, m, t, w, f)}{S(v, m, t)},$$

when the clean score is non-zero.

Attacked accuracy measures whether the clean order advantage survives in absolute task performance. We separately report *Keep-Correct* (KC), the fraction of examples answered correctly by the clean prompt that remain correct after perturbation. KC is conditional on a method-specific clean-correct set and is therefore not a replacement for attacked accuracy: a method with a lower clean score may have an easier or more stable conditional subset.

We first average over attacked component subsets for a fixed (v, m, t, w, f), then over feasible world–budget cells within each task–model–method triple, and finally over the 50 task–model pairs. Per-world summaries instead average over feasible budgets within a fixed world before averaging across pairs.

G.3.3 AGGREGATE AND PER-WORLD RESULTS

Table 23 summarizes the original 50-pair experiment. ORDERCOVER attains 63.48% attacked accuracy, remaining 2.28 points above Anchor and 0.41 points above Random-5; the clean Oracle order is only 0.22 points higher. ORDERCOVER exceeds Anchor in attacked accuracy on 39 of 50 pairs and exceeds Random-5 on 29, with 21 losses. Thus, most of the clean ordering advantage is retained on average, but the smaller advantage over Random is heterogeneous.

KC shows a different ordering. Anchor retains 91.77% of its clean-correct examples, slightly above Random-5, ORDERCOVER, and clean Oracle. The orders with the highest attacked accuracy are therefore not the most conservative on examples they initially solve. This does not reverse the attacked-accuracy result, because the methods condition KC on different clean-correct subsets; rather, it shows that overall attacked performance and conditional stability capture different behavior.

World	Method	Attacked acc.	Robust KC
Character	Anchor	59.66%	87.64%
	Random-5	60.74%	87.17%
	ORDERCOVER	61.72%	87.21%
	Clean Oracle	62.15%	87.58%
Word	Anchor	61.49%	93.20%
	Random-5	62.74%	93.00%
	ORDERCOVER	64.01%	92.85%
	Clean Oracle	64.29%	92.36%
Sentence	Anchor	61.35%	92.24%
	Random-5	62.33%	91.30%
	ORDERCOVER	63.75%	91.45%
	Clean Oracle	63.64%	90.54%
Semantic	Anchor	62.31%	94.02%
	Random-5	63.22%	93.89%
	ORDERCOVER	64.41%	93.31%
	Clean Oracle	64.73%	93.23%

Table 24: Per-world robustness results over the original 50 task–model pairs. Values first average over feasible fault budgets within each pair and then across pairs.

Table 24 gives the per-world breakdown. ORDERCOVER remains above Anchor in attacked accuracy in all four worlds, with margins of 2.06, 2.52, 2.40, and 2.10 points for character, word, sentence, and semantic perturbations, respectively. Anchor has the highest KC in every world. The distinction between attacked accuracy and conditional stability is therefore not driven by a single perturbation family.

G.3.4 INCREASING FAULT BUDGETS AND BRITTLE REVERSALS

To compare increasing fault budgets on identical support, we use the 41 pairs for which all four methods have measurements for $f \in \{0, 1, 2, 3\}$ in every perturbation world. Attacked accuracy generally decreases as more components are perturbed. The ORDERCOVER–Anchor and ORDERCOVER–Random-5 gaps contract with f but remain positive on average throughout this common-support range. At $f = 2$, the ORDERCOVER–Anchor attacked-accuracy gap is positive in all four worlds.

Pair-level behavior is not universal. On SBIC sexual-offense classification with Llama-3.1-8B, the 22.1-point clean advantage over Anchor remains a 16.41-point advantage after averaging over perturbation worlds and feasible budgets. StereoSet race relation with Llama-3.3-70B similarly retains a 5.38-point attacked-accuracy advantage over Anchor. Conversely, on SBIC sexual-offense classification with Qwen3.5-9B, the approximately 9.9-point clean gain becomes a -2.77-point attacked-accuracy difference. The stress test therefore identifies a minority of brittle clean wins rather than implying that clean order gains are invariant to perturbation.

Scope of the robustness claim. This analysis supports only a narrow conclusion: orders selected on clean data usually preserve their relative attacked-accuracy advantage on the original five-model subset. It does not evaluate adaptive order search under perturbation, robustness on the two newly added models, input corruption, or adversarially optimized attacks.