ORCH3D: GLOBAL COMPUTE ORCHESTRATION FOR MULTI-STAGE 3D GENERATION

Anonymous authors
Paper under double-blind review



Figure 1: Scene and object comparisons of Orch3D and the DPCache-based DP baseline.

ABSTRACT

Modern 3D generators are increasingly multi-stage, making efficient orchestration of their growing inference computation critical for practical deployment. However, existing acceleration methods largely optimize individual stages or operators in isolation, overlooking dependencies across both the generative pipeline and sparse execution. We find that stage-wise optima are non-compositional: changing an upstream schedule can reverse downstream schedule preferences, showing that computation must be optimized over the complete generation process. Based on this finding, we introduce **Orch3D**, a global compute orchestration framework built on two complementary components: **Pareto Compute Optimizer (PaCO)**, which jointly optimizes cross-stage compute allocation and within-stage sampling schedules through allocation-conditioned Pareto search, and **Packed Sparse Window Attention (PSWA)**, a structure-preserving sparse execution scheme that converts irregular window partitions into reusable packed attention schedules while preserving the original attention semantics. PaCO determines where and how much computation should be spent, while PSWA reuses window structure and cached boundaries to reduce the cost of retained computation. Under an equal 90-minute search budget, complete-policy joint search attains feasible deployment policies in every evaluated search seed, whereas the evaluated factorized procedure does not; frozen joint policies also improve reference-model fidelity on held-out scenes. PSWA adds semantics-preserving execution acceleration; together Orch3D reaches $1.895\times$–$3.776\times$ end-to-end speedup after transferring frozen policies to SA-3DAO without retuning.

1 INTRODUCTION

Efficient inference is becoming a central bottleneck for high-fidelity 3D generation. Modern 3D generators increasingly rely on multi-stage diffusion pipelines to progressively construct comple-

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.

mentary 3D representations (Xiang et al., 2025; Zhao et al., 2025; SAM 3D Team et al., 2025), while their sparse representations further introduce highly irregular computation inside each stage. Yet most acceleration techniques are still designed for isolated diffusion trajectories or individual operators (Sabour et al., 2024; Li et al., 2023; Zhang et al., 2026; Haghighi & Alahi, 2026; Chung et al., 2026; Yang et al., 2026; Cui et al., 2026). As 3D generation becomes increasingly multi-stage and sparse, this local view leaves the end-to-end computation fundamentally under-optimized.

The core difficulty is that multi-stage acceleration is no longer a separable scheduling problem. Every upstream scheduling decision changes the intermediate state seen downstream, thereby changing the downstream quality landscape itself; a schedule that is optimal under one upstream policy may become inferior under another. Joint optimization must therefore search simultaneously over how computation is distributed across stages and where evaluations are placed within every trajectory, producing a large coupled combinatorial space in which stage-local objectives and transition costs are no longer reliable. Meanwhile, sparse 3D execution introduces a second structural inefficiency: highly imbalanced window occupancy makes regularized attention execution spend computation on semantically masked cross-window pairs and associated masking/bookkeeping rather than only on valid within-window interactions. Simply combining existing stage-wise schedulers with generic sparse operators therefore fails to address either source of global inefficiency.

Our key finding makes the scheduling difficulty explicit: changing an upstream sampling policy can reverse the ranking of downstream schedules. As illustrated in Figure 2, even when the two stages are individually optimized, their composition need not be globally optimal. This cross-stage preference reversal shows that stage-wise optima are non-compositional: the best end-to-end policy cannot, in general, be recovered by combining independently optimized schedules. The correct optimization unit is therefore not an individual diffusion trajectory, but the complete multi-stage execution policy.

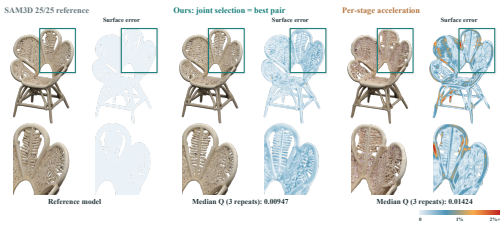

Figure 2: Stage-wise optima need not compose. Independently preferred SS and SLaT schedules can form a suboptimal policy.

Based on this finding, we introduce **Orch3D**, a framework for global compute orchestration over both cross-stage sampling structure and irregular sparse computation. **Pareto Compute Optimizer (PaCO)** directly searches complete multi-stage policies, jointly optimizing cross-stage compute allocation and within-stage sampling placement across different quality and latency budgets. **Packed Sparse Window Attention (PSWA)** turns the model-defined sparse-window partition into a reusable packed schedule: it reuses token order, packs Q/K/V by window, caches cumulative boundaries, executes only window-local attention, and restores the sparse layout. This removes masked cross-window work without changing the model, sparse structure, or sampling policy. PaCO casts joint scheduling as an allocation-conditioned Pareto search problem. Stage-wise compute allocation defines distinct compute regimes, while timestep placement is optimized within and across these regimes using direct end-to-end fidelity and measured latency. This formulation avoids constructing stage-local surrogate objectives whose validity changes with the upstream policy, and a single offline search produces a repertoire of globally optimized schedules for different deployment budgets. PSWA complements this search by exposing the preserved window boundaries directly to execution.

On SAM 3D Objects, we observe clear cross-stage preference reversals that motivate complete-policy optimization. Under the same search budget, complete-policy joint search improves over the evaluated factorized procedure, while fixed-policy comparisons isolate complementary gains from structure-aware sparse orchestration with PSWA. Together, Orch3D achieves end-to-end speedups from $1.895\times$ to $3.776\times$, and the resulting frozen policies transfer to SA-3DAO without retuning. Our contributions are threefold:

- We identify cross-stage preference reversal in multi-stage 3D generation, revealing non-compositional stage-wise optima and motivating global complete-policy search.
- We formulate multi-stage acceleration as complete-policy joint optimization over cross-stage compute allocation and within-stage sampling placement, and introduce **Pareto Compute Optimizer (PaCO)**, an allocation-conditioned Pareto search using complete-rollout feedback and measured latency.

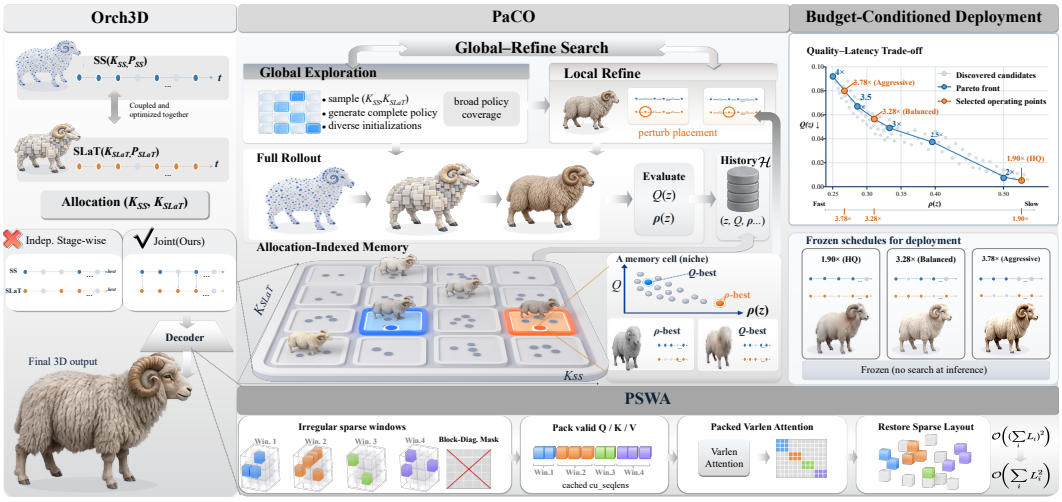

Figure 3: Overview of **Orch3D**. PaCO performs policy-level compute orchestration by jointly searching cross-stage compute allocation and within-stage sampling schedules using complete-rollout quality and measured latency. PSWA performs operator-level orchestration through a pack–execute–restore path that reuses the sparse-window partition, caches cumulative boundaries, evaluates only window-local attention, and restores the original sparse layout. The selected policy is frozen after offline search and executed without online adaptation.

- We introduce **Packed Sparse Window Attention (PSWA)**, a structure-preserving sparse execution scheme that reuses irregular window partitions as packed execution boundaries, caches recurring sequence metadata, and restores sparse-token layout exactly after window-local attention.

2 Related Work

3D generation and sampling acceleration. 3D generation spans diffusion priors (Ho et al., 2020; Poole et al., 2023), view/multi-view methods (Liu et al., 2023; Shi et al., 2024), feed-forward reconstruction (Hong et al., 2024; Xu et al., 2024), and staged systems (Xiang et al., 2025; Zhao et al., 2025; SAM 3D Team et al., 2025). Acceleration uses faster solvers (Song et al., 2021; Lu et al., 2022; Karras et al., 2022), timestep optimization (Li et al., 2023; Sabour et al., 2024; Zhu et al., 2026; Zhang et al., 2026), and path planning with reuse (Cui et al., 2026). PaCO evaluates complete cross-stage policies with measured end-to-end feedback, indexes placement subspaces by stage allocation, uses fidelity/latency extrema as refinement parents, and retains the observed repertoire for budget-conditioned deployment. Descriptor-indexed archives and cell-wise Pareto sets follow MAP-Elites/MOME (Mouret & Clune, 2015; Pierrot et al., 2022).

Caching and efficient attention. Repeated denoiser/DiT evaluation (Peebles & Xie, 2023) motivates DeepCache/FasterCache (Ma et al., 2024; Lyu et al., 2025), TaylorSeer (Liu et al., 2025), and adaptive 3D caching (Haghighi & Alahi, 2026; Chung et al., 2026; Yang et al., 2026). Swin localizes attention (Liu et al., 2021), while FlashAttention accelerates exact attention (Dao et al., 2022; Dao, 2024). PSWA instead targets sparse-3D execution, converting model-defined irregular windows and their token order into a reusable packed schedule with invariant metadata.

3 Method

Orch3D accelerates multi-stage 3D generation through global compute orchestration. Rather than treating sampling and kernel execution as isolated optimization targets, Orch3D coordinates computation at two complementary granularities. At the policy level, Pareto Compute Optimizer (PaCO) jointly determines how computation is distributed across generative stages and where evaluations are placed along their sampling trajectories. At the operator level, Packed Sparse Window Attention (PSWA) reorganizes irregular sparse-window execution to avoid masked dense cross-window

work without approximating the underlying model computation. The execution backend stays fixed throughout search, so PaCO optimizes realized deployment latency.

3.1 Global Compute Orchestration

For an M-stage generator, stage m retains K_m solver intervals at ordered positions P_m. We treat the full allocation-and-placement choice as one policy,

$$z = \{(K_m, P_m)\}_{m=1}^{M}, \tag{1}$$

because every downstream stage consumes an intermediate state produced by the preceding schedule. This coupling appears empirically as preference reversal: for downstream schedules u, v, two upstream policies a, b can satisfy

$$Q(a,u) < Q(a,v), \qquad Q(b,u) > Q(b,v), \tag{2}$$

so a single upstream-independent ranking cannot represent both conditions. The reversal also imposes a nonzero worst-case error on additive stage-local scores; Proposition 1 and its measurement-error conditions are given in Appendix O.1.

Let $Q_{\mathcal{D}}(z)$ be Official25-relative terminal discrepancy and $\rho_{\mathcal{D}}(z)$ paired normalized end-to-end latency on development set $\mathcal{D}$. Orch3D evaluates complete rollouts and optimizes

$$\min_{z \in \mathcal{Z}} \big(Q_{\mathcal{D}}(z), \rho_{\mathcal{D}}(z)\big), \tag{3}$$

without a stage-additive quality model. For SAM 3D, $M = 2$ (SS and SLaT). Metric definitions and timing normalization are in Appendix A; the schedule-grid construction is specified in Appendix D.1.

3.2 PaCO: Pareto Compute Optimizer

PaCO organizes complete policies by the stage-wise allocation descriptor

$$\phi(z) = (K_1, \ldots, K_M), \tag{4}$$

while policies inside one cell differ only in timestep placement. For completed observations $\mathcal{H}_c$ in allocation cell c, PaCO retains the within-cell quality–latency repertoire

$$\mathcal{A}_c = \mathrm{ND}_{(Q,\rho)}(\mathcal{H}_c). \tag{5}$$

This partitions the search space without factorizing evaluation: every proposal still receives complete-policy Q and end-to-end latency.

Search alternates global proposals, which introduce new allocations and placements, with allocation-preserving refinement. Each occupied cell contributes up to two parents—its best-fidelity and least-latency observations—and a local move edits one or two internal timesteps while preserving allocation. Cell-conditioned memory therefore keeps refinement support across observed compute regimes, while global proposals continue to expand coverage. Initialization and anchor definitions are in Appendix D.1; legal mutation, deduplication, and the frozen fallback are in Appendix D.2, with support analysis in Appendix D.4.

The two anchors guide refinement, whereas deployment queries the full eligible pool or its recomputed Pareto repertoire, which can select non-anchor policies. After search, one protocol-eligible repertoire supports every latency target. For budget $\bar{\rho}$,

$$\hat{z}(\bar{\rho}) = \operatorname*{lex\,argmin}_{\substack{z \in \mathcal{V} \\ \rho(z) \le \bar{\rho}}} \big(Q(z), \rho(z), k(z)\big), \tag{6}$$

and the selected schedule is frozen for all inputs. Recomputing the final cell-wise Pareto repertoire preserves this selector for any budget; the proof is in Appendix O.3.

3.3 PSWA: Packed Sparse Window Attention

Each retained evaluation still contains irregular sparse-window attention. Let L_i be the valid-token count in window i and $L = \sum_i L_i$. The masked dense reference attends over the concatenated axis, so its quadratic work contains both useful within-window pairs and masked cross-window pairs:

$$L^2 = \sum_i L_i^2 + 2 \sum_{i<j} L_i L_j. \tag{7}$$

PSWA treats the sparse-window partition itself as an execution schedule. The existing sparse path supplies forward/inverse token permutations and valid window lengths L_i; PSWA reuses them to gather Q/K/V into window-contiguous segments, caches recurring cumulative boundaries, executes only within those segments, and restores sparse-token order with the inverse permutation. Relative to the masked-dense reference, attention work becomes $\mathcal{O}(\sum_i L_i^2)$ rather than $\mathcal{O}(L^2)$.

PSWA therefore neither repartitions nor densifies the sparse representation: the pack–execute–restore path uses preserved window boundaries as the isolation mechanism and evaluates exactly the same within-window blocks as the block-diagonal reference. Appendix E.1 gives the packed construction and exact-arithmetic identity; Appendix E.2 gives the fixed-policy timing and numerical fidelity checks.

3.4 Unified Orchestration

PaCO and PSWA act at complementary levels rather than as jointly optimized variables. The execution backend is fixed during each search: PaCO selects complete sampling policies using measured end-to-end latency, while PSWA reduces the realized cost of executing those policies. Because deployment is latency-constrained, backend-specific execution time determines which frozen policies are feasible at a target budget. Appendix E.5 formalizes this budget-conditioned selection effect, and Appendix E.4 verifies it on a frozen policy pool. Deployment then uses one static policy with no online adaptation.

4 Experiments

4.1 Experimental Setup

We evaluate ORCH3D primarily on the frozen SAM 3D Objects generator (SAM 3D Team et al., 2025). PaCO performs offline complete-policy search on LVIS (Gupta et al., 2019) using terminal discrepancy $\mathcal{E}$ and measured end-to-end latency. The resulting policies are frozen before SA-3DAO evaluation; no SA-3DAO ground-truth metric is used for search, selection, or retuning. We compare against uniform and random timestep schedules and representative diffusion-acceleration methods, including Fast3Dcache, TaylorSeer, Fast-SAM3D, PFDiff, SeaCache, and DPCache. PSWA is evaluated separately with the model, schedule, and model-evaluation count fixed, isolating its infrastructure-level contribution. Unless otherwise stated, SA-3DAO speedups use the same clean SAM 3D Objects reference of 8.729788 s/object. Complete metric definitions and timing protocols are given in Appendix A.

Beyond end-to-end quality and speed, our experiments ask: **(i)** whether downstream schedule rankings depend on upstream policies, **(ii)** whether nominal sampling compute predicts realized latency, and **(iii)** whether execution acceleration can alter which policy is selected under a deployment budget.

4.2 Finding I: Stage-Wise Optima Are Non-Compositional

We first examine the assumption underlying stage-wise acceleration, namely whether independently optimized stage-wise schedules compose into a strong end-to-end policy.

Cross-stage preference reversal. Under an upstream-independent ranking, the answer is no. Across two independently selected six-scene cohorts, changing the upstream SS schedule reverses the preregistered preference between downstream SLaT schedules in $4/6$ scenes in each cohort. Importantly, this is not merely a change in the magnitude of downstream error: the *ranking* of downstream schedules itself changes with the upstream policy. A SLaT schedule preferred under one SS trajectory can become inferior under another.

This supports Eq. (2): a fixed upstream-independent ranking cannot be reused unchanged. Conditional or sequential solvers are not excluded (Appendix B.3); PaCO instead evaluates complete policies directly.

Joint versus factorized optimization. We next test this coupling under the same 90-minute wall-clock budget. The frozen factorized arm spends 48 stage-local references before validating at most 16 pairs selected by its additive predictor (Appendix B.2). At the registered $\rho \leq 0.4$ ($\geq 2.5\times$) target,

Table 1: Evidence for non-compositional multi-stage scheduling.

A. Preference reversal	
Readout	Result
Cohort 1	$4/6$ scenes
Cohort 2	$4/6$ scenes
Rate	66.7%

B. Finite-budget feasibility		
Readout	Joint	Factorized
Feasible @ 30 min	**3/3**	0/3
Feasible @ 60 min	**3/3**	0/3
Feasible @ 90 min	**3/3**	0/3

C. Holdout generalization	
Readout	Result
Holdout (12 scenes)	**11/12 better**
95% CI ($\Delta\mathcal{E}$)	$[0.0148,\ 0.0537]$
CI excludes 0	Yes

Note. Factorized pair validation starts at 4863.7–5014.2 s (90–93% of the budget); complete-policy search validates deployable pairs directly. Scope: this preregistered baseline only.

joint complete-policy search finds a feasible policy in all $3/3$ fresh seeds, whereas the factorized procedure finds none.

Held-out validation. We further freeze the jointly and independently selected policies and evaluate them on a separate 12-scene holdout. Joint policies obtain lower terminal discrepancy in $11/12$ scenes, with a paired 95% confidence interval of $[0.0148, 0.0537]$ for

$$\mathcal{E}_{\text{Independent}} - \mathcal{E}_{\text{Joint}}.$$

Together, these results show that downstream preferences are conditional, the evaluated factorized procedure spends most of its budget before pair validation, and the resulting joint policies generalize better than that baseline beyond the development cohort.

The full composition grid and finite-budget timing details are provided in Appendices B.3 and B.2.

4.3 Finding II: Nominal Compute Is Not Realized Compute

Joint scheduling raises a second question, namely how "compute" should be defined. A common approximation is to use the number of model evaluations, but we find that NFE alone does not determine the actual deployment cost of a multi-stage 3D generator.

Placement matters even at fixed allocation. At the same SS12/SLaT3 allocation, changing only timestep placement reduces mean runtime from 2.8622 to 2.6974 s/object, a 5.76% latency reduction. The stage allocation, model, and execution backend are unchanged. Thus, two policies with the same number of evaluations in every stage can still incur different realized costs.

Allocation matters even at fixed total NFE. The same phenomenon appears on an independent multi-stage generator, TRELLIS. Three policies all execute the same 42 realized model evaluations, but reallocating those evaluations across stages changes end-to-end latency from 2.443 to 2.699 seconds, a 10.5% range. In other words, an evaluation in one stage is not cost-equivalent to an evaluation in another.

These results motivate a central design choice in PaCO: policies are optimized using *measured end-to-end latency*, rather than NFE or stage allocation as a surrogate. Global compute orchestration must determine not only how much nominal computation is retained, but where that computation occurs and what it actually costs on the deployed system.

4.4 PaCO: Searching the Coupled Policy Space

The previous findings motivate complete-policy search; we next evaluate PaCO's search performance and the contribution of allocation-conditioned memory separately.

Comparison with generic multi-objective search. Fair NSGA-II starts from the same measured B16 bootstrap and uses the same objectives and logical proposal budget. Each slot is one proposal opportunity: Fair duplicates reuse stored measurements, while PaCO rejects them before evaluation. Thus matching is by proposal opportunities, not fresh rollouts or GPU time. At $\geq 3.5\times$ and $\geq 4\times$, PaCO obtains lower terminal discrepancy in all three reported runs. The lower-speed region is mixed and the aggregate four-tier comparison remains unresolved at $n = 3$; this supports high-speed search behavior, not universal or equal-compute superiority. The runs comprise one Stage-A configuration-selection run and two with that configuration frozen (Appendix D.5).

Table 2: Nominal sampling compute does not uniquely determine realized latency.

Fixed allocation			**Fixed total NFE**		
Placement	Time (s)	Δ	Alloc.	E2E (s)	Rel.
Uniform	2.8622	–	16/8	**2.443**	1.000×
Searched	**2.6974**	−5.76%	12/12	2.561	1.048×
Saved	0.1648	5.76%	8/16	2.699	1.105×

Table 3: PaCO search evidence: B64 PaCO/Fair; B128 4G+4R memory factorial.

PaCO/Fair (3/3 wins)			**Alloc. memory** (5/6 fav.)		
Seed	$\mathcal{E}$@3.5×	$\mathcal{E}$@4×	Readout	Full	−Alloc.
S1	**.0740**/.0811	**.0740**/.1055	Parent cells	**18.67**	3.83
S2	**.1114**/.1193	**.1174**/.1634	H_{norm}	**.866**	.349
S3	**.0693**/.0775	**.0693**/.0775	Portfolio $J\downarrow$	**1.0054**	1.0869

Global exploration and refinement. Pure Global uses only global proposals, retaining complete-policy evaluation and budget-conditioned selection. Its reported B64 tier winners exactly match PaCO's in all three paired seeds; the B128 contrast remains unresolved (Appendix D.7). The aggressive-tier gains over Fair NSGA-II come from global proposals (Appendix D.5). The following factorial separately examines parent-memory design.

Role of allocation-conditioned memory. PaCO's local mutation preserves allocation, making the parent pool structurally important: refinement from a given parent pool can access only the allocation cells represented in that pool. Allocation-conditioned memory therefore serves as a persistent map of which compute regimes remain accessible to local search.

We isolate memory design in a preregistered six-seed B128 factorial: all arms use 4G+4R and share the measured B16 bootstrap and global proposal stream. Deployment remains 7G+1R. Allocation-conditioned memory increases mean parent-cell coverage from 3.83 to 18.67 and normalized allocation entropy from 0.349 to 0.866. The four-tier portfolio score simultaneously decreases from 1.0869 to 1.0054, with the favorable direction appearing in $5/6$ seeds. Preregistered terminal-J main effects are -0.0574 for allocation conditioning (95% CI $[-0.1173, 0.0025]$) and -0.0352 for latency anchors (95% CI $[-0.0786, 0.0083]$). Both meet the preregistered $5/6$-seed directional criterion; both intervals include zero. These terminal results complement coverage (Appendix D.3).

Search-budget behavior. PaCO also shows clear diminishing returns with search budget. B64 captures 86.3%, 85.2%, and 85.0% of the observed B16→B128 median-discrepancy reduction at the 2.5×, 3×, and 3.5× tiers, respectively. Doubling the budget to B128 continues to improve median quality, but the additional gains are comparatively modest. We therefore retain B64 as the frozen deployment budget and report the full budget analysis in Appendix D.6.

4.5 Finding III: Execution Efficiency Changes the Deployable Policy

PaCO determines which evaluations are retained, while deployment feasibility depends on the realized cost of executing them. This makes backend-specific latency part of budget-conditioned policy selection, even though the execution backend is fixed during search.

PSWA accelerates fixed policies without changing their computation. Across the dense control and three frozen accelerated schedules, PSWA provides 1.1209×–1.1616× end-to-end speedup while keeping the model, schedule, valid tokens, window partition, and model-evaluation count fixed, thereby testing the pack–execute–restore pathway under fixed sparse structure.

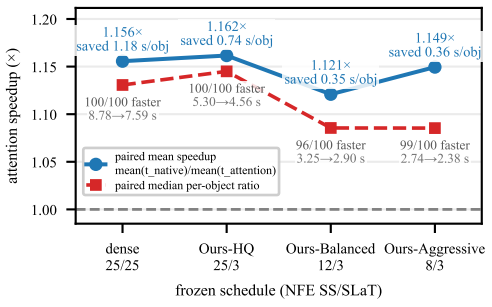

Figure 4: PSWA speedup with fixed model, policy, and NFE across four schedules.

Schedule	Time (s)	Speedup	Faster objects
Dense	8.78 → 7.59	1.156×	100/100
HQ	5.30 → 4.56	1.162×	100/100
Balanced	3.25 → 2.90	1.121×	96/100
Aggressive	2.74 → 2.38	1.150×	99/100

Table 4: Fixed-policy PSWA acceleration.

Target	Native eligible	PSWA eligible	Selected $\mathcal{E}$
2.5×	9	11	.041 → .041
3.0×	3	**8**	.076 → **.041**
3.5×	0	**4**	−→ .076

Table 5: Execution backend changes the budget-feasible policy set without rerunning search.

Table 6: Controlled decomposition and quality retention.

(a) Controlled decomposition of policy- and operator-level acceleration.

Component	Median gain	Speedup
PSWA on dense policy	11.78%	1.1699×
Schedule optimization only	35.10%	1.5773×
PSWA on compressed policy	14.59%	1.1958×
Full stack	**44.44%**	**1.8860×**

(b) Quality-retaining deployment under a 5% seven-metric gate.

Point	Pass	Time (s)↓	Speedup↑
DPCache-HQ	7/7	8.5910	1.0162×
DPCache-Fast	2/7	4.5060	1.9374×
ORCH3D-HQ	7/7	4.6068	1.8950×
ORCH3D-Balanced	**7/7**	**2.6592**	**3.2829×**
ORCH3D-Aggressive	5/7	2.3117	3.7763×

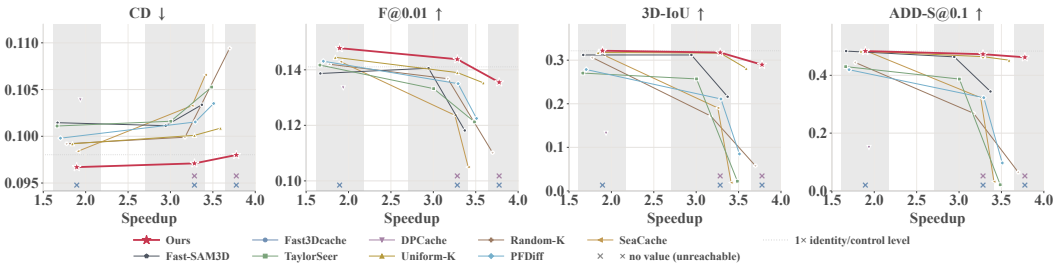

Figure 5: Quality–speed trade-offs on SA-3DAO. ORCH3D-HQ/Balanced/Aggressive operate at 1.8950×, 3.2829×, and 3.7763×, respectively.

Because PSWA preserves the underlying sparse-window operation, the resulting quality changes are small and attributable to finite-precision execution and iterative numerical propagation. For dense/sparse schedules, CD changes by $+0.77\%/+0.14\%$, F@0.01 by $-0.89\%/-0.35\%$, and 3D-IoU by $+0.01\%/+0.02\%$. Appendix E.2 reports the full fixed-policy fidelity readout, while Appendix E.1 gives the exact-arithmetic operator identity.

Execution changes policy feasibility. More importantly, PSWA can change which policy should be deployed under a fixed latency target. We freeze 15 previously evaluated policies and all corresponding quality values, then remeasure only their execution latency. At 3×, PSWA expands the feasible set from 3 to 8 policies. A lower-discrepancy policy that reaches only 2.749× under native execution reaches 3.294× with PSWA and becomes feasible, changing the selected discrepancy from 0.07566 to 0.04126. At 3.5×, the native backend admits no policy, whereas PSWA admits four.

PSWA therefore does not improve the quality of a fixed policy; it enlarges the latency-feasible set and can allow a lower-discrepancy frozen policy to be selected. Orch3D uses this backend-specific measured latency when selecting policies for deployment.

Full-stack decomposition. A controlled same-session decomposition separates the contributions of schedule optimization and packed execution. Schedule optimization alone yields 1.5773×, while PSWA yields 1.1699× on the dense policy and 1.1958× on the compressed policy. Combining the searched schedule and PSWA reaches 1.8860× in this controlled setting.

4.6 END-TO-END QUALITY–EFFICIENCY TRADE-OFFS

The three findings above translate into a substantially broader deployment frontier. Table 29 (Appendix J) reports three representative policies selected on LVIS and transferred to SA-3DAO without retuning. ORCH3D-HQ reaches 1.8950×, ORCH3D-Balanced reaches 3.2829×, and ORCH3D-Aggressive reaches 3.7763×.

Figure 5 shows the corresponding quality–speed trade-offs. Near 3.28×, ORCH3D-Balanced records CD/F@0.01 of 0.0971/0.1438, compared with 0.1001/0.1390 for the nearby Uniform-K operating point. The remaining four metrics are reported in Appendix F.1.

A stricter seven-metric retention test provides a more deployment-oriented view. Requiring every primary SA-3DAO metric to remain within 5% of the clean reference, ORCH3D-Balanced passes all seven metrics at 3.2829×. DPCache-HQ also passes 7/7 but reaches only 1.0162×, whereas DPCache-Fast reaches 1.9374× while passing only 2/7. Thus, Orch3D shifts not only the maximum-speed point but the quality-feasible deployment frontier.



Figure 6: Representative qualitative comparison on SA-3DAO. ORCH3D better preserves global geometry and fine structures under aggressive acceleration. Extended comparisons are given in Appendix F.2.

Overall, the final $1.8950\times$–$3.7763\times$ speed range is best understood as the consequence of three structural findings: stage-wise quality is conditional, realized compute is not determined by nominal NFE, and execution efficiency changes the budget-feasible policy set. PaCO and PSWA address these properties at complementary levels, turning them into practical end-to-end acceleration; Appendix J gives the full operating-point table and compact ablation and sensitivity summaries.

The evidence is intentionally separated by claim. The joint-search study supports complete-policy feedback relative to the preregistered factorized baseline under a fixed wall-clock budget, without ruling out every conditional or sequential solver. The PaCO memory study establishes broader refinement support and favorable terminal trends, while the default paired control remains tied at B64 and unresolved at B128. PSWA is evaluated with frozen schedules and a fixed execution backend, so its evidence concerns semantics-preserving execution acceleration and budget-conditioned deployment selection rather than joint backend search. These boundaries make the end-to-end gains directly attributable to the measurements reported in each controlled study.

5 CONCLUSION

We present ORCH3D, a global compute orchestration framework for multi-stage 3D generation. Our study reveals that stage-wise schedules are non-compositional, nominal sampling compute does not uniquely determine realized latency, and execution efficiency can change which policy is selected under a deployment budget. Guided by these findings, PaCO jointly orchestrates compute allocation and sampling placement across stages, while PSWA reduces the cost of irregular sparse-window execution without approximating the underlying computation. Together, they produce frozen policies spanning $1.8950\times$–$3.7763\times$ end-to-end acceleration on SA-3DAO without retuning. More broadly, our results suggest treating efficient multi-stage generation as a complete-pipeline compute-orchestration problem: optimize policies against measured system cost, preserve the execution semantics of retained computation, and freeze the selected policy for deployment. This separation keeps offline search complexity out of inference while making the deployed operating point explicit and reproducible. A single offline search also yields a reusable family of budget-conditioned policies, so deployment can switch operating points without online search or model retraining. Appendix K summarizes the scope and evidence boundaries governing interpretation of all reported search, execution, and deployment results.

Reproducibility statement

Schedules use explicit endpoint lists and hashes, and runs retain provenance, metrics, timing receipts, and call traces. Appendix D specifies proposal and mutation logic; Appendix A defines metrics and timing; Appendix I records provenance and audit reconstruction; Appendix O gives the supporting derivations. Anonymized code and artifact manifests accompany the submission.

AI use statement

Generative AI tools were used to aid or polish manuscript writing. The authors reviewed the AI-assisted text and take responsibility for the final manuscript.

References

Jiwoo Chung, Sangeek Hyun, MinKyu Lee, Byeongju Han, Geonho Cha, Dongyoon Wee, Youngjun Hong, and Jae-Pil Heo. Seacache: Spectral-evolution-aware cache for accelerating diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14283–14294, 2026.

Bowen Cui, Yuanbin Wang, Huajiang Xu, Biaolong Chen, Aixi Zhang, Hao Jiang, Zhengzheng Jin, Xu Liu, and Pipei Huang. Denoising as path planning: Training-free acceleration of diffusion models with dpcache. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 43632–43642, 2026.

Tri Dao. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 16344–16359, 2022.

Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5356–5364, 2019.

Yasaman Haghighi and Alexandre Alahi. Sencache: Accelerating diffusion model inference via sensitivity-aware caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14295–14304, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large Reconstruction Model for Single Image to 3D. In *International Conference on Learning Representations (ICLR)*, 2024.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 26565–26577, 2022.

Lijiang Li, Huixia Li, Xiawu Zheng, Jie Wu, Xuefeng Xiao, Rui Wang, Min Zheng, Xin Pan, Fei Chao, and Rongrong Ji. Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7105–7114, 2023.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863, 2025.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9298–9309, 2023.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022, 2021.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Zhengyao Lyu, Chenyang Si, Junhao Song, Zhenyu Yang, Yu Qiao, Ziwei Liu, and Kwan-Yee K. Wong. Fastercache: Training-free video diffusion model acceleration with high quality. In *International Conference on Learning Representations (ICLR)*, 2025.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating Diffusion Models for Free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15762–15772, 2024.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Thomas Pierrot, Guillaume Richard, Karim Beguir, and Antoine Cully. Multi-objective quality diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 139–147, 2022. doi: 10.1145/3512290.3528823.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: Optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42947–42975, 2024.

SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J. Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jia-Wei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images. *arXiv preprint arXiv:2511.16624*, 2025.

Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. MVDream: Multi-view Diffusion for 3D Generation. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations (ICLR)*, 2021.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2025.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models. *arXiv preprint arXiv:2404.07191*, 2024. URL `https://arxiv.org/abs/2404.07191`.

Mengyu Yang, Yanming Yang, Chenyi Xu, Chenxi Song, Yufan Zuo, Tong Zhao, Ruibo Li, and Chi Zhang. Fast3dcache: Training-free 3d geometry synthesis acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27030–27040, 2026.

Travis Zhang, Christian Belardi, Justin Lovelace, Jin Peng Zhou, Saebyeol Shin, Carla P. Gomes, and Kilian Q. Weinberger. Optimize your sampling: Tuned diffusion sampling with bayesian optimization. *arXiv preprint arXiv:2608.18040*, 2026.

Zibo Zhao et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

Aihua Zhu, Rui Su, Qinglin Zhao, Li Feng, Meng Shen, and Shibo He. Hierarchical schedule optimization for fast and robust diffusion model sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 13907–13915, 2026.

A Evaluation Protocol and Metrics

This section defines the evaluation quantities used throughout the paper. We separate three types of measurements: (i) the terminal discrepancy $\mathcal{E}$ used during PaCO search, (ii) measured end-to-end latency used for compute orchestration, and (iii) downstream SA-3DAO metrics used only for frozen-policy evaluation. No SA-3DAO ground-truth metric is used to construct or retune the reported Orch3D policies.

A.1 Development-Time Terminal Discrepancy

Let V_c and V_r denote the candidate and paired-reference mesh vertices after applying their predicted scene poses. Let ℓ_r be the largest side length of the posed reference bounding box and $\mathbf{b}_r$ its lower corner. We normalize both meshes in the shared reference coordinate system as

$$\bar{V} = (V - \mathbf{b}_r)/\ell_r. \tag{8}$$

The scene-space geometry discrepancy is

$$d_{\text{scene}} = \frac{1}{|\bar{V}_c|} \sum_{x \in \bar{V}_c} \min_{y \in \bar{V}_r} \|x - y\|_2^2 + \frac{1}{|\bar{V}_r|} \sum_{y \in \bar{V}_r} \min_{x \in \bar{V}_c} \|y - x\|_2^2. \tag{9}$$

We additionally measure normalized extent and centroid deviations,

$$e_{\text{extent}} = \frac{1}{3} \sum_{k=1}^{3} \frac{|e_{c,k} - e_{r,k}|}{\max(|e_{r,k}|, 10^{-6}\ell_r)}, \tag{10}$$

$$e_{\text{centroid}} = \frac{\|\boldsymbol{\mu}_c - \boldsymbol{\mu}_r\|_2}{\ell_r}, \tag{11}$$

and define the per-object terminal discrepancy

$$\mathcal{E}_o = \sqrt{d_{\text{scene}}} + e_{\text{extent}} + e_{\text{centroid}}. \tag{12}$$

Historical search campaigns retain their frozen aggregation rules for reproducibility. Fresh scene-grouped comparisons use

$$\mathcal{E}_{\text{scene}}(z) = \text{median}_g \, \text{median}_{o \in g} \, \mathcal{E}_o(z), \tag{13}$$

whereas the historical B16/B32/B64/B128 trajectories use the pooled-object median. We never mix these aggregations within one controlled comparison.

A.2 Measured Latency and Speedup

For object o, let

$$\bar{t}_o(z) = \frac{1}{n_{\text{rep},o}(z)} \sum_{j=1}^{n_{\text{rep},o}(z)} t_{o,j}(z) \tag{14}$$

denote its warmed mean runtime under policy z.

For paired LVIS search studies, normalized latency is

$$\rho^{\beta}(z) = \frac{1}{N} \sum_{o=1}^{N} \frac{\bar{t}_o(z)}{b_o^{\beta}}, \qquad \beta \in \{\text{on}, \text{cert}\}, \tag{15}$$

and normalized speedup is

$$S^{\beta}(z) = 1/\rho^{\beta}(z). \tag{16}$$

The online denominator is available during search, whereas the certified denominator incorporates the final timing audit. The latter changes only the final readout and never retroactively changes proposal history.

For SA-3DAO, we report mean absolute runtime

$$T_{\mathrm{mean}}(z) = \frac{1}{N} \sum_{o} \bar{t}_o(z) \tag{17}$$

and compute speedup from the common clean SAM 3D Objects reference,

$$S_{\text{SA-3DAO}}(z) = \frac{8.729788}{T_{\mathrm{mean}}(z)}. \tag{18}$$

This common-reference convention makes cross-method deployment readouts directly comparable.

A.3 SA-3DAO METRICS

SA-3DAO deterministic shape metrics. Each predicted and ground-truth mesh is normalized independently by centering its vertex bounding box and scaling its longest extent to length two. We then estimate a rigid point-to-point ICP transform from separate fixed-seed 10k surface samples, using at most 50 iterations and a tolerance of 10^{-7}, and apply that transform to the predicted shape. CD, F@0.01, V-IoU, and NC are evaluated deterministically on fixed-seed 1M-point surface samples. Let P and G denote the aligned predicted and ground-truth samples and let $d(x, A) = \min_{a \in A} \|x - a\|_2$. We compute

$$\mathrm{CD}(P, G) = \frac{1}{2} \left[\frac{1}{|P|} \sum_{p \in P} d(p, G) + \frac{1}{|G|} \sum_{g \in G} d(g, P) \right]. \tag{19}$$

$$\mathrm{Prec}_\tau = \frac{1}{|P|} \sum_{p \in P} \mathbf{1}[d(p, G) < \tau]. \tag{20}$$

$$\mathrm{Rec}_\tau = \frac{1}{|G|} \sum_{g \in G} \mathbf{1}[d(g, P) < \tau]. \tag{21}$$

$$\mathrm{F}_\tau = \frac{2\,\mathrm{Prec}_\tau\,\mathrm{Rec}_\tau}{\mathrm{Prec}_\tau + \mathrm{Rec}_\tau}, \quad \tau = 0.01. \tag{22}$$

with $\mathrm{F}_\tau = 0$ when the denominator is zero. For voxel IoU, let $\mathcal{V}_{64}(P)$ and $\mathcal{V}_{64}(G)$ be the occupied cells obtained by clipping the normalized aligned samples to $[-1, 1]^3$ and mapping them into a 64^3 grid. Then

$$\text{V-IoU}(P, G) = \frac{|\mathcal{V}_{64}(P) \cap \mathcal{V}_{64}(G)|}{|\mathcal{V}_{64}(P) \cup \mathcal{V}_{64}(G)|}. \tag{23}$$

For normal consistency, each sampled point carries the unit normal of its sampled mesh face. Let $j(p)$ be the nearest ground-truth sample to p and $i(g)$ the nearest predicted sample to g, using the same nearest-neighbor assignments as CD. After rotating the predicted normals by the ICP rotation, the orientation-insensitive symmetric score is

$$\mathrm{NC}(P, G) = \frac{1}{2} \left[\frac{1}{|P|} \sum_{p \in P} |\mathbf{n}_p^\top \mathbf{n}_{j(p)}| + \frac{1}{|G|} \sum_{g \in G} |\mathbf{n}_g^\top \mathbf{n}_{i(g)}| \right]. \tag{24}$$

These four values are deterministic under the frozen sampling seeds and ICP protocol. They follow the published benchmark metric definitions, but they are local public-split reproductions rather than official server scores because the official evaluator and several implementation details are not public.

SA-3DAO auxiliary EMD estimate. The auxiliary EMD column in Table 29 uses a finite-iteration entropic transport approximation. The other metrics also use the explicitly stated finite surface-sampling protocol. We draw separate fixed-seed sets P_E and G_E of 1024 surface points and transform P_E by the same shape ICP alignment. With uniform empirical marginals $a_i = 1/|P_E|$ and $b_j = 1/|G_E|$, squared Euclidean ground cost $C_{ij} = \|p_i - g_j\|_2^2$, and transport polytope $\Pi(a, b)$, the entropically regularized plan is

$$\pi_\epsilon^\star = \arg \min_{\pi \in \Pi(a,b)} \langle \pi, C \rangle + \epsilon \sum_{i,j} \pi_{ij} (\log \pi_{ij} - 1). \tag{25}$$

$$\epsilon = 0.01. \tag{26}$$

Writing $\widehat{\pi}_\epsilon$ for the finite numerical iterate, rather than the exact optimum $\pi_\epsilon^\star$, the reported auxiliary distance is

$$\widehat{\mathrm{EMD}} = \left(\sum_{i,j} \widehat{\pi}_{\epsilon,ij} C_{ij} \right)^{1/2}. \tag{27}$$

We solve this plan in the log domain for at most 200 Sinkhorn iterations with marginal tolerance 10^{-5} and retain the realized iteration count and marginal error as diagnostics. The hat emphasizes that this is a finite-iteration entropic approximation. The official sampling count, regularization strength, iteration budget, and convergence rule are not public, so this column is not used as an official leaderboard score or as a primary claim.

SA-3DAO deterministic layout metrics. Canonical surface samples are transformed into camera space using the predicted and ground-truth similarity poses. The reported layout scores use fixed-seed 100k surface samples. Let B_P and B_G be the axis-aligned bounding boxes of the posed point sets. The 3D IoU score is

$$\mathrm{IoU}_{3D} = \frac{\mathrm{Vol}(B_P \cap B_G)}{\mathrm{Vol}(B_P \cup B_G)}. \tag{28}$$

For posed point clouds P^S, G^S, let D_G denote the evaluator's deterministic ground-truth point-cloud diameter. The symmetric normalized ADD-S score is

$$\text{ADD-S} = \frac{1}{2D_G} \left[\frac{1}{|P^S|} \sum_{p \in P^S} d(p, G^S) + \frac{1}{|G^S|} \sum_{g \in G^S} d(g, P^S) \right]. \tag{29}$$

The per-object success indicator is $\mathbf{1}[\text{ADD-S} < 0.1]$, and ADD-S@0.1 is its mean over the pose-annotated objects. These layout metrics are deterministic under the fixed sampling and pose protocol. SA-3DAO shape metrics are averaged over 100 objects and layout metrics over the 93 objects with pose annotations.

We report four shape metrics (CD, F@0.01, V-IoU, and NC), one auxiliary finite-iteration EMD estimate, and three posed/layout metrics (3D-IoU, ADD-S, and ADD-S@0.1). Shape metrics are evaluated over all 100 objects; layout metrics use the 93 objects with pose annotations. The deterministic sampling, ICP, and finite-iteration EMD definitions remain exactly as in the original evaluation protocol.

A.4 Portfolio and Tail Readouts

For a target speedup s, method m, search seed r, and budget B, let

$$\mathcal{E}_{m,r,s}(B) = \min_{\substack{z \in \mathcal{V}_{m,r,B} \\ \rho^{\mathrm{cert}}(z) \le s^{-1}}} \mathcal{E}(z). \tag{30}$$

Unattained tiers remain unattained rather than receiving an artificial penalty value.

The normalized multi-tier portfolio score J is used only for controlled search comparisons. Lower J is better. Because its normalization depends on the comparison set and common attained tiers, J values from different experiments are not directly interchangeable.

B Finding I: Cross-Stage Coupling and Non-Compositionality

The first main finding of the paper is that a downstream schedule does not possess a fixed quality ranking independent of the upstream trajectory. This section provides the full empirical and structural evidence behind

that claim.

B.1 Cross-Stage Preference-Reversal Protocol

We test downstream schedule preference under multiple frozen upstream trajectories. A reversal occurs when two downstream schedules u and v satisfy

$$\mathcal{E}(a,u) < \mathcal{E}(a,v), \qquad \mathcal{E}(b,u) > \mathcal{E}(b,v) \tag{31}$$

for two upstream policies a and b.

Across two independently selected six-scene cohorts, the preregistered downstream preference reverses in $4/6$ scenes in each cohort. The important observation is not merely that upstream schedules influence downstream error magnitude; they change the *ordering* of candidate downstream schedules.

Table 7: Controlled evidence: feasible search attainment and the 12-scene holdout.

(a) Feasible search			(c) 12-scene holdout	
Time	Joint	Fact.	Readout	Value
30 min	**3/3**	0/3	Scenes	12
60 min	**3/3**	0/3	Joint better	**11/12**
90 min	**3/3**	0/3	95% CI of ΔQ	**[.0148,.0537]**
Gate	$\rho \leq 0.4$		CI excludes 0	**yes**

This appendix documents the composition diagnostic summarized in Section 4.2. It adds no new rollout: every number is recomputed offline from the audited six-scene confirmation grid of the preregistered 2×3 schedule study, using only its canonical measurement of each cell. Repeated cells enter the readout through their canonical measurement only; the repeats are used exclusively to size the repeat-noise envelope. The cohort is six mixed indoor ScanNet scenes with three masks each, and all six cells share one execution stack and the same SS23/SLaT3 allocation family.

The compact three-way readout is reported in Table 8 in the main paper; the full six-cell grid and selector variants below provide the robustness details.

B.2 Finite-Budget Joint versus Factorized Search

The 90-minute experiment compares complete-policy search with the frozen factorized procedure under the same wall-clock budget. The factorized arm is frozen before launch: it first evaluates 24 SS-reference schedules with SLaT25 fixed and 24 SLaT-reference schedules with SS25 fixed, then predicts complete compositions using $\widehat{Q} = Q_{\mathrm{SS}} + Q_{\mathrm{SLaT}}$ and $\widehat{\rho} = \rho_{\mathrm{SS}} + \rho_{\mathrm{SLaT}} - 1$. It selects at most 16 pairs with the preregistered 8 quality-priority / 4 speed-priority / 4 boundary-coverage rule; only full SS–SLaT rollouts are eligible deployment policies. All 48 stage-local references and pair validations are charged to the same 5400-s budget, and no observations are shared across arms.

Joint search reaches the registered $\rho \leq 0.4$ ($\geq 2.5\times$) deployment target in all three seeds. The factorized arm reaches none within the cutoff. Its first complete-pair validation begins after 4863.7, 4960.5, and 5014.2 seconds, corresponding to 90.1%, 91.9%, and 92.8% of the total search budget.

This diagnostic exposes a practical cost of factorization: substantial budget is spent producing stage-local evidence before the method can evaluate the object ultimately used at deployment—the complete policy.

B.3 Frozen Composition Diagnostic

The frozen 2×3 grid isolates whether independently preferred components necessarily compose into the strongest complete policy.

Table 8: Controlled evidence: frozen composition diagnostic.

(i) Composition diagnostic		
Rule	Policy	Q / Time
Fact.	SB/L3	.0236 / 75.91
J-Lat	SB/L1	.0235 / **74.23**
J-Qual	SB/L2	**.0221** / 78.54
Gain	–	$Q: -6.6\%$

Table 9: The six complete policies of the frozen grid. Q is reported under the grid's frozen pooled-object median aggregation and, as a robustness check, under a median-of-scene-medians aggregation; clean latency is the six-scene total. Both aggregations rank SB–L2 first. They disagree on the remaining order: the stage-marginal composition SB–L3 ranks third (pooled) versus second (scene-median), the latency-matched cell SB–L1 ranks second versus third, and SA–L1 and SA–L3 swap ranks 5/4 for 4/5.

Cell	Q (pooled) ↓	Q (scene-median) ↓	Clean latency (s)	Rank (pooled)	Rank (scene-med.)
SA–L1	0.02522	0.02833	72.49	5	4
SA–L2	0.02659	0.03160	75.81	6	6
SA–L3	0.02463	0.02962	74.11	4	5
SB–L1	0.02346	0.02103	74.23	2	3
SB–L2	**0.02206**	**0.01809**	78.54	**1**	**1**
SB–L3	0.02362	0.01928	75.91	3	2

Under the pooled-object readout the stage marginals are $M_{\mathrm{SS}}(\mathrm{SA}) = 0.02548$ versus $M_{\mathrm{SS}}(\mathrm{SB}) = 0.02305$, and $M_{\mathrm{SLaT}}(\mathrm{L1}) = 0.02434$, $M_{\mathrm{SLaT}}(\mathrm{L2}) = 0.02432$, $M_{\mathrm{SLaT}}(\mathrm{L3}) = 0.02413$. The stage-marginal rule therefore composes to SB–L3, the third-ranked cell of the pooled readout. The joint quality-best cell SB–L2 lowers Q by 6.61% relative to that composition, while a joint selection restricted to the composition's own clean-latency budget returns SB–L1, which lowers both terminal error and measured latency under the same pooled readout.

Under the pooled-object readout, stage-marginal composition selects SB–L3 with $\mathcal{E} = 0.02362$, whereas direct complete-policy selection finds SB–L2 with $\mathcal{E} = 0.02206$, a 6.61% reduction. The diagnostic is intentionally limited: several conditional selectors recover SB–L2, so the experiment does not show that every factorized rule must fail. Instead, it illustrates the precise point established by preference reversal: the outcome depends on how downstream preference is conditioned on the upstream policy. The sequential conditional selector therefore shows that conditioning can recover the grid optimum; this diagnostic rejects a fixed upstream-independent stage ranking, not conditional optimization itself.

Table 10: Frozen factorized-selector variants evaluated on the same six cells. Anchored variants choose each stage at a fixed anchor cell (u_0, v_0); the $(\mathrm{SA}, \mathrm{L1})$ row is the rule previously frozen in the confirmation analysis. The sequential row is a conditional (greedy) rule rather than a stage-marginal one, and it reaches the grid minimum. This is why the diagnostic describes specific frozen local-preference rules rather than factorized selection in general.

Selector variant	Anchor	Composed cell	Q (pooled) ↓	Equals joint best
Stage-marginal (main effect)	—	SB–L3	0.02362	no
Additive surrogate argmin	—	SB–L3	0.02362	no
Anchored	$(\mathrm{SA}, \mathrm{L1})$	SB–L3	0.02362	no
Anchored	$(\mathrm{SA}, \mathrm{L2})$	SB–L3	0.02362	no
Anchored	$(\mathrm{SA}, \mathrm{L3})$	SB–L3	0.02362	no
Anchored	$(\mathrm{SB}, \mathrm{L1})$	SB–L2	**0.02206**	yes
Anchored	$(\mathrm{SB}, \mathrm{L2})$	SB–L2	**0.02206**	yes
Anchored	$(\mathrm{SB}, \mathrm{L3})$	SB–L2	**0.02206**	yes
Sequential conditional	—	SB–L2	**0.02206**	yes

The additive variant fits the balanced two-way main-effect surrogate $\mu + a_u + b_v$ and takes its grid argmin; its maximum absolute residual on these six cells is 1.05×10^{-3}, and because a balanced main-effect fit is separable its argmin coincides with the stage-marginal composition by construction. Anchored variants that select the SLaT preference at the reference cell return SB–L3, whereas anchoring that preference at the selected SS cell returns SB–L2. The fixed anchor therefore determines whether a factorized readout coincides with the grid minimum, which is why we do not present this diagnostic as evidence against factorized selection in general.

The difference between the composition and the joint quality-best cell is 1.56×10^{-3} in Q, about $191\times$ the aggregate repeat envelope of the pooled-object median (8.17×10^{-6}, the larger of the two repeated cells). The latency-matched difference is 1.62×10^{-4}, about $19.8\times$ that aggregate envelope but $0.78\times$ the largest per-object repeat change (2.07×10^{-4}); its latency difference of 1.68 s is $3.1\times$ the aggregate latency repeat envelope (0.54 s). The grid protocol defines no paired significance test for these compositions, so we report multiples of the observed repeat envelope and make no significance claim.

Scene-wise signs are mixed and are reported as measured. Per-scene medians over the three objects of each scene give $Q(\mathrm{SB}--\mathrm{L2}) < Q(\mathrm{SB}--\mathrm{L3})$ in 3/6 scenes, with 3/6 exceeding the per-object repeat envelope in that direction and 3/6 exceeding it in the opposite direction; $Q(\mathrm{SB}--\mathrm{L1}) < Q(\mathrm{SB}--\mathrm{L3})$ holds in 2/6 scenes, with four scenes exceeding the per-object envelope in the opposite direction. Re-running the latency-matched selection separately per scene, using that scene's own SB–L3 clean latency as its budget, returns SB–L1 in 2/6 scenes, SB–L3 in 3/6, and SA–L3 in 1/6. A three-object scene median has a noise scale comparable to these differences, whereas the pooled aggregate over 18 objects resolves them; the aggregate readout used in the main text is thus a cohort-level statement, not a per-scene one. Under a median-of-scene-medians aggregation the joint quality-best cell remains SB–L2 (6.14% lower Q), but the latency-matched selection returns SB–L3 itself, removing that gain. No absolute quality-safety conclusion against OFFICIAL25 is drawn here, and the preregistered cross-stage rank reversals, the 90-minute Joint-versus-Factorized study, and the 12-scene holdout remain the primary evidence.

B.4 Why Preference Reversal Breaks Additive Stage Models

Preference reversal also gives a simple structural obstruction to an upstream-independent additive surrogate. Define the positive reversal margins

$$\Delta_a = \mathcal{E}(a,v) - \mathcal{E}(a,u) > 0, \qquad \Delta_b = \mathcal{E}(b,u) - \mathcal{E}(b,v) > 0. \tag{32}$$

For any additive approximation

$$\widetilde{\mathcal{E}}(s,t) = f(s) + g(t), \tag{33}$$

the uniform approximation error over $\{a,b\} \times \{u,v\}$ satisfies

$$\max_{(s,t)} \left| \mathcal{E}(s,t) - \widetilde{\mathcal{E}}(s,t) \right| \geq \frac{\Delta_a + \Delta_b}{4}. \tag{34}$$

The result does not imply that dynamic programming is impossible in a larger state space. It shows instead that a fixed stage-separable objective cannot exactly represent the observed preference structure. Complete-policy evaluation avoids introducing this approximation.

C Finding II: Realized Compute Beyond NFE

The second main finding is that nominal evaluation count does not uniquely determine the realized cost of a multi-stage 3D generator. We isolate this effect both within SAM 3D and on TRELLIS.

C.1 Fixed-Allocation Placement Diagnostic

We hold the allocation fixed at SS12/SLaT3 and compare uniform placement with the frozen searched placement under the same execution backend. Changing placement alone reduces runtime from 2.8622 to 2.6974 s/object, corresponding to a 5.76% latency reduction.

Table 11: Controlled evidence: fixed-allocation timestep-placement ablation.

(g) Fixed SS12/SLaT3		
Metric	U/U→S/S	Δ
Time ↓	2.862→**2.697**	−5.76%
CD ↓	.0999→.0980	†
3D-IoU ↑	.3253→.3176	†
Q CIs	cross 0	

We isolate timestep placement at the frozen SS12/SLaT3 allocation. Uniform placement (U/U) and the frozen search-selected placement (S/S) use the same candidate execution stack and are measured in one balanced, same-GPU window. The placement change alone reduces mean latency from 2.8622 to 2.6974 s/object, corresponding to a $1.0611\times$ pairwise speed ratio or a 5.76% reduction in seconds/object.

Table 11 in the main paper gives the compact readout; the paired bootstrap intervals are $[-0.00131, 0.00594]$ for CD and $[-0.02208, 0.00728]$ for 3D-IoU.

The two preregistered quality intervals cross zero, so this ablation is not used to rank the placements by quality. Its cost conclusion is nevertheless direct: even after stage allocation, model, and execution stack are fixed, the retained timestep locations measurably change end-to-end runtime. This is why PaCO evaluates complete policies with measured latency rather than treating NFE or allocation alone as the deployment cost.

The preregistered CD and 3D-IoU confidence intervals cross zero, so this

experiment is not used to claim a quality ordering between the two placements. Its systems conclusion is independent of that ambiguity: identical stage allocation and NFE do not imply identical realized cost.

C.2 TRELLIS: EQUAL NFE, DIFFERENT REALIZED COST

We repeat the cost diagnostic on TRELLIS (Xiang et al., 2025), an independent two-stage 3D generator. Three policies each execute exactly 42 realized model evaluations but distribute them differently across stages.

Table 12: Controlled evidence: TRELLIS equal-NFE stage reallocation.

(h) TRELLIS: 42 evals		
Alloc.	SS / SLaT (s)	E2E (s)
16/8	1.310 / .889	**2.443**
12/12	.990 / 1.321	2.561
8/16	.673 / 1.785	2.699
Range	–	+10.5%

A2: stage-allocation-dependent realized cost. To test whether evaluation count alone is a reliable deployment-cost proxy on an independent two-stage 3D generator, we run the preregistered A2 diagnostic on TRELLIS (Xiang et al., 2025) under one fixed execution backend. The three allocations in Table 12 all execute the same 42 realized model evaluations; only the cross-stage allocation differs. End-to-end latency nevertheless increases monotonically from 2.4430 to 2.6993 seconds as computation is shifted from SS toward SLaT, a 10.5% increase from the fastest to the slowest allocation. The direction is consistent on all 8 objects and is substantially larger than the repeated-timing envelope. The stage breakdown indicates the main reason: an SLaT interval is consistently more expensive than an SS interval under this backend, so equal evaluation counts are not equal realized compute costs.

Table 12 in the main paper gives the fixed-count latency readout; the corresponding sparse-coordinate totals are 10.79k, 11.03k, and 11.46k.

The sparse-coordinate count is monotone in the aggregate (10.79k $\rightarrow$ 11.03k $\rightarrow$ 11.46k), but only 5/8 objects follow the preregistered object-wise direction. We therefore do not attribute the latency gap primarily to downstream token workload. Under the frozen decision rule, the stronger A2 downstream-workload endpoint remains INCONCLUSIVE. The supported conclusion is narrower and directly relevant to our cost model: stage allocation changes realized latency even at identical model-evaluation count, so complete schedules must be judged by measured end-to-end cost rather than nominal NFE alone. Absolute TRELLIS seconds are not compared against SAM 3D because the implementations use different execution stacks; only same-backend paired comparisons are interpreted.

A1: fixed-count placement interaction. A1 holds both stages at K12. Under its recorded CFG accounting, each stage executes 12 solver intervals and 9 additional branch evaluations, hence 42 realized evaluations per complete policy. Two SS placements (E12, L12) are crossed with three SLaT placements (E12, U12, L12), using $n_{\mathrm{rep}} = 3$ repeats.

For a fixed object, upstream arm, and downstream pair, let d_j be the paired terminal-error difference in repeat j. The frozen resolution rule uses

$$\bar{d} = \frac{1}{n_{\mathrm{rep}}} \sum_j d_j. \tag{35}$$

$$\sigma_d = \left[\frac{\sum_j (d_j - \bar{d})^2}{n_{\text{rep}} - 1} \right]^{1/2}. \tag{36}$$

$$\text{resolved} = \mathbf{1}[\sigma_d > 0 \ \wedge \ |\bar{d}| > 2\sigma_d]. \tag{37}$$

Here σ_d is repeat standard deviation, not standard error. This is a registered noise-resolution heuristic, not an automatic 95% significance test. A resolved reversal requires opposite signs and resolved sides under both SS arms. Aggregate repeat variation was small in this diagnostic; taking a median does not mathematically force it to zero. The frozen PASS rule additionally requires at least one object-level reversal to clear its own envelope. The verdict and thresholds are unchanged.

The 16/8, 12/12, and 8/16 allocations require 2.4430, 2.561, and 2.6993 seconds end-to-end, respectively. Thus, changing allocation alone produces a 10.5% latency range at identical total NFE.

The supported conclusion is deliberately narrow: evaluation counts are not interchangeable across stages, so deployment cost should be measured rather than inferred from nominal NFE.

C.3 TRELLIS Placement Interaction

The A1 TRELLIS diagnostic fixes both stages at K12 and varies only their placement patterns. The aggregate preferred downstream placement changes with the upstream SS path, but no object-level reversal satisfies the registered resolution criterion. We therefore retain this result as INCONCLUSIVE rather than using it as cross-model confirmation of Finding I.

Table 13: TRELLIS A1 fixed-count placement diagnostic. All six cells execute the same 42 realized model evaluations. The aggregate best downstream placement changes with the upstream SS placement, but the object-level reversal is unresolved under the frozen noise rule, so the verdict is INCONCLUSIVE. Lower values are better; $\succ$ denotes preference, not a numerical greater-than relation.

Upstream SS	L1 (E12)	L2 (U12)	L3 (L12)	Best-to-worst
SS-A (E12)	0.008547	**0.008519**	0.008641	$L2 \succ L1 \succ L3$
SS-B (L12)	**0.008761**	0.008764	0.008767	$L1 \succ L2 \succ L3$

Although the aggregate ordering changes, every pair has 0/8 object-level resolved reversals. The only individually resolved side is $L2$ versus $L3$ under SS-B ($d = -4.9 \times 10^{-5}$, $\sigma = 2.4 \times 10^{-5}$); the corresponding SS-A comparison is not resolved, so it does not constitute a resolved reversal. The mechanism diagnostics also show that the sparse-coordinate count is identical across the three SLaT arms for a fixed SS path (10,268 for SS-A and 10,161 for SS-B). This is expected because the three SLaT arms share the same upstream trajectory and interval count, and it means A1 tests placement-dependent quality ordering rather than workload coupling. We therefore classify A1 as INCONCLUSIVE, keep A2's stronger workload mechanism unresolved, do not launch TRELLIS Phase B, and stop this external study here.

This negative result is useful for separating the paper's claims: cross-stage quality reversal is strongly established on SAM 3D, whereas TRELLIS is used here primarily to validate the broader cost observation that equal NFE does not imply equal latency.

D PaCO Search Protocol and Analysis

Finding I motivates complete-policy search; this section specifies PaCO and examines how its allocation-conditioned memory supports that search.

D.1 Search Space and Initialization

For SAM 3D, an allocation cell is

$$c = (K_{\text{SS}}, K_{\text{SLaT}}). \tag{38}$$

The frozen domain contains 24 cells grouped into high-quality, balanced, and aggressive bands.

Algorithm 1 specifies the common proposal/evaluation skeleton. The allocation domain $\mathcal{C}$, distinct from development set $\mathcal{D}$, has three frozen bands:

$$\mathcal{C}_{\text{high}} = \{(25,3),(24,4),(23,4),(21,6),(20,7),(18,8),(17,9),(16,10)\}, \tag{39}$$

$$\mathcal{C}_{\text{bal}} = \{(13,3),(12,3),(11,4),(10,5),(10,6),(9,6),(11,5),(12,4)\}, \tag{40}$$

$$\mathcal{C}_{\text{agg}} = \{(8,3),(7,3),(6,3),(6,4),(5,4),(7,4),(7,5),(8,4)\}. \tag{41}$$

Their union contains 24 cells. A global K-interval path samples $K-1$ distinct internal indices from $\{1,\dots,24\}$, sorts them, and adds endpoints 0 and 25. The implemented batch schedule uses $B = 16 + 8M$, including the reported budgets 16, 32, 64, and 128. Global, parent-selection, and mutation random streams are separately keyed.

For each occupied allocation cell c, refinement uses the two lexicographic extrema

$$e_Q(c) = \operatorname*{lex\,argmin}_{z \in \mathcal{H}_c} \big(Q(z), \rho(z), \mathrm{id}(z)\big), \tag{42}$$

$$e_\rho(c) = \operatorname*{lex\,argmin}_{z \in \mathcal{H}_c} \big(\rho(z), Q(z), \mathrm{id}(z)\big). \tag{43}$$

Duplicate anchors are merged before parent sampling.

A K-interval schedule selects $K-1$ internal grid indices from $\{1,\dots,24\}$ and adds endpoints 0 and 25. The B16 bootstrap samples complete policies across the allocation bands before continuation search.

D.2 Global Exploration and Dual-Anchor Refinement

For every occupied allocation c, PaCO defines two refinement anchors,

$$e_{\mathcal{E}}(c) = \operatorname*{lex\,argmin}_{z \in \mathcal{H}_c} \big(\mathcal{E}(z), \rho(z), \mathrm{id}(z)\big), \tag{44}$$

$$e_\rho(c) = \operatorname*{lex\,argmin}_{z \in \mathcal{H}_c} \big(\rho(z), \mathcal{E}(z), \mathrm{id}(z)\big). \tag{45}$$

Global proposals explore new allocations and placements, while local mutations modify placements around these anchors without changing the parent allocation.

Legal mutation and frozen fallback. For a stage list P, index i is mutable when $\{p_{i-1}+1,\dots,p_{i+1}-1\} \setminus \{p_i\}$ is nonempty. In the original coordinate-mutation planner, a dense unmodifiable stage is replaced by the other stage, the requested count is clipped to the number of mutable indices, and that many indices are sampled. Replacements are drawn from their original-neighbor intervals; joint assignments must leave the entire list strictly increasing. The other stage and the allocation remain unchanged. Thus "local" means low-dimensional and within-allocation, not necessarily a numerical ± 1 move.

The FRESH9-derived planner uses a separately frozen finite legal-neighbor fallback: it tries the requested parent, stage, and count first, then the alternative count/stage and other eligible parents in the prescribed order. Legal simultaneous neighbors are enumerated, scanned from a seeded offset, and checked against all prior and current-batch keys. The realized parent, stage, count, and fallback flag are retained. This is not identical to an unbounded independent-coordinate rejection sampler; a change of parent is an explicit fallback, not unreported within-parent refinement. Global proposal rejection is bounded at 5,000 attempts per slot. Exhaustion and invalid outputs never trigger an unregistered substitution.

Every batch uses only pre-batch observations. Pareto-interior policies are retained as an empirical repertoire, but parent selection uses only the two extrema. Final certified fronts and deployment choices are derived on the final eligible pool and denominator; they do not retroactively alter the search history.

The implementation uses only information available before each batch. All newly proposed policies are evaluated through the complete SS–SLaT–decoder pipeline.

Algorithm 1 PaCO search skeleton with ordered history and explicit deduplication.

Require: seed ξ, $B = 16 + 8M$, allocation domain $\mathcal{C}$, frozen allocator Π
$\mathcal{H} \leftarrow [\,]$; $\mathcal{K} \leftarrow \emptyset$ ▷ history and observed policy keys
sample $3/3/2$ allocations without replacement from the three bands
for each of the eight sampled allocations **do**
  sample two distinct complete policies; execute each on $\mathcal{D}$
  append $(z, Q, \rho^{\mathrm{on}}, \{\bar{t}_o\})$ to $\mathcal{H}$; add $k(z)$ to $\mathcal{K}$
end for
while $|\mathcal{H}| < B$ **do**
  $(m_G, m_R) \leftarrow \Pi(\mathcal{H})$, with $m_G + m_R = 8$
  $\mathcal{B} \leftarrow [\,]$; $\mathcal{K}_{\mathcal{B}} \leftarrow \emptyset$; $\mathcal{P} \leftarrow [\,]$
  sample eight distinct allocations from $\mathcal{C}$; retain the first m_G
  for each retained allocation **do**
    sample a complete policy with key outside $\mathcal{K} \cup \mathcal{K}_{\mathcal{B}}$
    append it to $\mathcal{B}$; add its key to $\mathcal{K}_{\mathcal{B}}$
  end for
  if $m_R > 0$ **then**
    for each observed cell c in fixed order **do**
      compute $e_Q(c), e_\rho(c)$ by Eqs. (42)–(43)
      append distinct anchor identities to $\mathcal{P}$
    end for
    shuffle $\mathcal{P}$ with the batch-parent RNG
    for $j = 1, \ldots, m_R$ **do**
      choose the next parent cyclically from $\mathcal{P}$
      request a stage and an edit count in $\{1, 2\}$ using their fixed RNG streams
      obtain an unseen legal offspring by the frozen snapshot's mutation/fallback rule
      append it to $\mathcal{B}$; record requested and realized operator metadata
      add its key to $\mathcal{K}_{\mathcal{B}}$; stop with a recorded error on exhaustion
    end for
  end if
  fully evaluate $\mathcal{B}$ on $\mathcal{D}$; append completed observations in batch order
  $\mathcal{K} \leftarrow \mathcal{K} \cup \mathcal{K}_{\mathcal{B}}$
  derive online niche fronts from $\mathcal{H}$; update Π from completed feedback
end while
return $\mathcal{H}$ and $\{\mathcal{A}_{c,\mathrm{on}}\}$

D.3 Why Allocation-Conditioned Memory Matters

PaCO's local mutation preserves allocation, so the parent pool determines which allocation subspaces remain reachable by refinement.

The preregistered six-seed factorial uses 4G+4R at B128, separately from the frozen 7G+1R deployment configuration. Within each seed, all four arms share the measured B16 bootstrap, global proposal stream, development cohort, and Official25 timing-reference basis. The crossed factors are allocation-conditioned versus global parent memory and quality/latency dual anchors versus quality-only anchors: A uses allocation-conditioned dual anchors, B allocation-conditioned quality anchors, C global dual anchors, and D global quality anchors. Requested mutation stages and edit counts use shared random streams; parent-dependent legal-neighbor fallbacks can change the realized operators, as reported below. This experiment evaluates parent-memory design under active refinement; it does not replace the default-mixture Pure Global control in Section 4.4.

In the six-seed refinement-active factorial, allocation-conditioned arms retain approximately 18–20 parent cells, whereas global-parent variants collapse to roughly 2–4. The normalized parent-allocation entropy changes accordingly from approximately 0.11–0.35 to 0.86–0.89.

Both preregistered directional rules are met: allocation conditioning lowers the normalized score by 0.0574 and the latency-extreme factor by 0.0352 on average, each with 5/6 seeds in the predicted direction. These are absolute differences in J, not relative percentages. Both paired confidence

Table 14: Controlled evidence: allocation-conditioned memory arms.

(d) PaCO memory			
Arm	$J \downarrow$	Cells	H_{norm}
Full	**1.0054**	18.67	.866
− Alloc.	1.0869	3.83	.349
− Lat.	1.0647	19.33	.892
− Both	1.0980	1.67	.110

Table 15: Preregistered six-seed refinement-active factorial at B128. $\Delta_{\text{niche}} = (J_A + J_B)/2 - (J_C + J_D)/2$ and $\Delta_\rho = (J_A + J_C)/2 - (J_B + J_D)/2$; negative values are the preregistered favorable directions.

Seed	J_A	J_B	J_C	J_D	Δ_{niche}	Δ_ρ
2026091821	1.0005	1.0005	1.0000	1.0005	+0.0003	-0.0003
2026091822	1.0000	1.0668	1.0915	1.0915	-0.0581	-0.0334
2026091823	1.0315	1.0000	1.1694	1.1694	-0.1537	+0.0158
2026091824	1.0000	1.0889	1.0248	1.0905	-0.0132	-0.0773
2026091825	1.0003	1.0537	1.0576	1.0576	-0.0306	-0.0267
2026091826	1.0000	1.1782	1.1782	1.1782	-0.0891	-0.0891
Mean	–	–	–	–	$\mathbf{-0.0574}$	$\mathbf{-0.0352}$
95% CI	–	–	–	–	$[-0.1173, 0.0025]$	$[-0.0786, 0.0083]$
Predicted direction	–	–	–	–	**5/6**	**5/6**

intervals include zero. The descriptive difference-in-differences interaction is $\Delta_{\text{int}} = J_A - J_B - J_C + J_D = -0.0483$, with 95% CI $[-0.1248, 0.0283]$; it is not a significant interaction claim.

Table 16: Mechanism readout averaged over the six factorial seeds. Niche-conditioned memory keeps refinement distributed across many allocation regimes, while global parent pools collapse to a small set. Latency-incumbent hits count latency-parent offspring that become a new online tier incumbent.

Arm	Refine cells	Parent cells	H_{norm}	Repair fallback	Latency hits
A: niche+Q/ρ	18.67	18.67	0.866	1.67	1/101
B: niche+Q	19.33	19.33	0.892	1.17	–
C: global+Q/ρ	3.83	3.83	0.349	12.50	7/166
D: global+Q	1.67	1.67	0.110	23.33	–

Parent-linkage provenance. The parent-cell and entropy readouts use the parent decisions frozen in the per-batch proposal plans rather than the copied identifiers stored in reused observations. The shared observation pool can return a previously measured schedule together with the creating arm's candidate metadata; direct within-arm ID matching therefore drops 14 of 1,344 logical refinement-parent links across the four factorial arms. Re-resolving the frozen plan-declared parents against each arm's own pre-batch history recovers 1,344/1,344 links with no schedule-key mismatch. This bookkeeping repair changes Arm A's mean parent-cell coverage from 18.50 to 18.67 and its entropy from 0.865 to 0.866, and changes Arm C's entropy from 0.350 to 0.349, without altering portfolio scores or rollout data.

The mechanism separation is substantially larger than the portfolio effect: niche-conditioned arms use roughly 18–20 parent/refinement cells, whereas global-parent arms use only 2–4 on average. The corresponding normalized parent-allocation entropy is 0.86–0.89 versus 0.11–0.35. Direct improvements from latency-extreme parents are sparse (1/101 T-parent offspring in A and 7/166 in C), so the negative Δ_ρ should not be interpreted as evidence that latency anchors help primarily through immediate tier-incumbent replacement.

Protocol notes. Each seed uses its own internally consistent Official25 pre/post denominator and all four factorial arms share that seed-level basis. The campaign used three timed repeats per object before the preregistered timing-compression amendment and one timed repeat afterward; the row-level `timed_repeats` field makes this mixed basis auditable. The pre-amendment study observed 0/32 tier-winner changes under one-repeat timing on the measured early pool; the amendment requires a final full-pool stability recheck before submission. Seed 2026091822 uses the completed pre-card-move unit: all four arms contain 128 rows and share that unit's own Official25 pre/post measurements; only its report phase was missing and was reconstructed read-only with the same

analyzer. The partially repeated new-card rows are not used. The six seeds were measured on six separate GPUs.

Pure-refinement stress diagnostic. We use a second controlled mechanism diagnostic to isolate the search function of the allocation-indexed parent pool. This experiment is not a generic GA or NSGA-II benchmark; instead, it matches the complete-policy evaluator and B64 budget while changing the parent-pool organization. The cleanest contrast is PaCO-R8 versus EA-Top2. Both begin from the same bit-identical B16 bootstrap, allocate all 48 continuation proposals to refinement (0G+8R per batch), and share the requested mutation stream. The only intended search difference is the parent pool: PaCO-R8 recomputes minimum-error and minimum-normalized-latency anchors within every occupied allocation cell, while EA-Top2 uses only the two global extrema.

Table 17: Allocation-collapse mechanism diagnostic at B64 on LVIS. "Refine cells" counts distinct $(K_{\mathrm{SS}}, K_{\mathrm{SLaT}})$ cells that generated the 48 post-bootstrap refinement proposals. ΔPortfolio is the preregistered normalized tier-portfolio difference PaCO-R8 minus EA-Top2; negative values mean the global two-elite arm is worse.

Search seed	PaCO-R8 cells	EA-Top2 cells	ΔPortfolio
2026091621	8	2	-0.1148
2026091622	8	2	-0.1150
2026091623	7	2	-0.0343
Mean	N/A	N/A	-0.0880

The restricted global-parent arm can refine only the allocations present in its parent pool because mutation preserves allocation. The niche-based arm instead supplies anchors from every occupied cell. This is a structural difference in parent availability, not by itself proof of deployment superiority. The mean score difference is $\Delta J = J_{\text{PaCO-R8}} - J_{\text{EA-Top2}} = -0.0880$ with paired 95% CI $[-0.2035, 0.0275]$ at $n = 3$. All seeds have the same direction and attain all five evaluated tiers; the interval nevertheless includes zero.

A second diagnostic removes continuation-stage global exploration while retaining niche memory (PaCO-R8 versus the historical 7G+1R operating point). The descriptive portfolio direction favors the operating point in 2/3 seeds, but the comparison also spans different GPUs. We do not treat it as a controlled performance effect or a relative percentage improvement. Finally, increasing mutation magnitude after the global parent pool has already collapsed to two cells does not reliably recover the lost portfolio: two seeds tie exactly and one favors the higher-mutation arm. Together, these diagnostics motivate the combination used by PaCO: persistent global proposals provide breadth, while allocation-conditioned anchors preserve structured depth for refinement.

The portfolio effect is smaller than the support effect. Allocation conditioning gives the favorable direction in $5/6$ seeds, but the confidence interval crosses zero. We therefore interpret this experiment primarily as evidence that the memory preserves structured local-search support, not as a universal claim that it improves every finite-budget portfolio.

D.4 Refinement-Support Analysis

For stage m with K_m intervals and grid endpoint T_m, the number of legal placements is

$$\binom{T_m - 1}{K_m - 1}. \tag{46}$$

An allocation cell therefore contains

$$|\mathcal{Z}_c| = \prod_{m=1}^{M} \binom{T_m - 1}{K_m - 1} \tag{47}$$

legal placement combinations before proposal restrictions.

Because PaCO's local mutation preserves allocation,

$$\mathcal{N}_R(z) \subseteq \mathcal{Z}_{\phi(z)}. \tag{48}$$

Hence, if an allocation disappears from the available parent pool, local refinement cannot re-enter that subspace without global exploration or an explicit fallback through another retained parent. This structural property explains the large parent-coverage difference observed empirically.

D.5 FAIR NSGA-II BASELINE

Fair NSGA-II is a preregistered proposal-budget-matched multi-objective search baseline using the same complete-policy objectives, development cohort, timing-certification protocol, measured B16 bootstrap, and logical proposal budget as PaCO. It uses random-key encoding, tournament selection, crossover, mutation, and immigrants rather than PaCO's proposal mechanism. New policy identities receive complete rollouts; previously measured identities reuse their stored observations while consuming a logical slot. Consequently, the comparison controls initial evidence and proposal count, not the number of fresh rollouts or total GPU time. Random-key encoding avoids the earlier stress diagnostic's illegal-policy repair channel.

Representation and exact decoding. A chromosome is

$$g = \left(c, r^{\mathrm{SS}}_{1:24}, r^{\mathrm{SLaT}}_{1:24}\right), \tag{49}$$

where the allocation gene c indexes one of the same 24 legal $(K_{\mathrm{SS}}, K_{\mathrm{SLaT}})$ cells used by PaCO and the remaining 48 genes lie in $[0, 1]$. For a stage with K solver intervals, decoding takes the positions of the $K - 1$ largest interior keys, sorts those positions, and brackets them with endpoints 0 and 25. Thus every chromosome maps to a legal ordered path without repair. The shared B16 policies are converted to chromosomes by a deterministic encoder whose selected interior positions receive larger keys than all unselected positions; the implementation verifies $\mathrm{decode}(\mathrm{encode}(z)) = z$ for every bootstrap policy. Fair NSGA-II therefore starts from exactly the same 16 measured complete policies as PaCO, not from a fresh random population.

Generation mechanics. The selected Fair configuration uses $\mu = 16$ population members and $\lambda = 8$ logical offspring slots per generation. Parent selection is binary tournament on nondominated rank and crowding distance. With probability $p_{\mathrm{cross}} = 0.60$, uniform crossover independently chooses every gene, including the allocation gene, from one of the two parents with equal probability; otherwise one parent is copied before mutation. Each of the 48 random-key genes mutates independently with probability $p_{\mathrm{key}} = 1/48$ by adding a uniform perturbation in $[-0.05, 0.05]$ and clipping to $[0, 1]$. Independently, the allocation gene changes to another legal cell with probability $p_{\mathrm{alloc}} = 0.10$. Two of the eight slots are fresh random-key immigrants. A duplicate decoded policy is neither repaired nor replaced: it consumes its slot and reuses the stored complete-rollout observation. The offspring retains its own chromosome, parent identifiers, and proposal provenance even when its measurement is reused. Environmental selection applies NSGA-II nondominated sorting and crowding to the parent and offspring records, retaining $\mu = 16$ members. Thus all objective values come from complete-policy evaluation, but not every logical offspring triggers a new rollout.

Logical budget, measurement reuse, and deployment readout. Generation 0 contains the same 16 measured policies for both searches. Budgets are logical history lengths: B32 contains the bootstrap plus two eight-slot generations, and B64 contains the bootstrap plus six. The observation pool is keyed by complete-policy identity. A cache hit avoids a new rollout but still occupies one offspring slot; reused timing remains associated with the GPU and reference basis that produced the measurement. The three Fair B64 reports contain 49, 47, and 51 distinct stored quality receipts, including inherited bootstrap evidence, rather than 64 fresh evaluations. These counts are cardinalities of recorded quality-receipt paths, not independent statistical sample sizes.

Table 18: Controlled evidence: PaCO versus Fair NSGA-II at the aggressive tiers.

	(b) PaCO vs. Fair	
Seed	$Q@\geq 3.5\times$	$Q@\geq 4\times$
S1	**.0740**/.0811	**.0740**/.1055
S2	**.1114**/.1193	**.1174**/.1634
S3	**.0693**/.0775	**.0693**/.0775
Wins	**3/3**	**3/3**

Algorithm 2 Fair NSGA-II (c1): matched logical slots and explicit measurement reuse.

Require: measured bootstrap $\mathcal{H}_{16}$, legal domain $\mathcal{C}$, observation pool $\mathcal{M}$
$P \leftarrow \mathrm{encode}(\mathcal{H}_{16})$ with measured objectives; $\mathcal{H} \leftarrow \mathcal{H}_{16}$
for $t = 1, \ldots, 6$ **do**
  compute nondominated rank and crowding distance on P
  $C \leftarrow [\,]$; $O \leftarrow [\,]$
  for $i = 1, \ldots, 6$ **do**
    select two parents from P by rank–crowding binary tournaments
    produce u by c1 uniform crossover and random-key/allocation mutation
    append u and its proposal provenance to C
  end for
  append two fresh random-key immigrants to C
  for each chromosome u in C **do**
    $z \leftarrow \mathrm{decode}(u)$; look up the coupled-schedule key $k(z)$
    if $k(z)$ has a stored observation in $\mathcal{M}$ **then**
      reuse that observation with its measurement-source and timing provenance
    else
      execute the full SS–SLaT–decoder rollout; store its observation in $\mathcal{M}$
    end if
    attach the observation to this offspring's own identity, chromosome, and provenance
    append the offspring record to O and $\mathcal{H}$ ▷ one logical slot, including reuse
  end for
  $P \leftarrow \mathrm{NSGAIISelect}(\mathrm{concat}(P, O), 16)$
end for
return ordered 64-slot history $\mathcal{H}$

Table 19: Recorded Fair NSGA-II B64 accounting, including the shared bootstrap. Repeated references to the same quality receipt consume logical slots without an additional full rollout.

Search seed	Logical slots	Distinct receipts	Repeated receipt slots
2026091621	64	49	15
2026091622	64	47	17
2026091623	64	51	13

The comparison therefore matches proposal opportunities, not newly executed rollout count or wall-clock cost. The deployment readout uses all protocol-valid observations in the budget prefix, not just the final 16-member population. With the certified timing basis, the tier selector is

$$z_s^\star(B) = \operatorname*{lex\,argmin}_{z \in \mathcal{H}_{1:B}:\, S^{\mathrm{cert}}(z) \geq s} \big(Q(z), \rho^{\mathrm{cert}}(z), \mathrm{id}(z)\big). \tag{50}$$

Duplicate records do not provide new measurement evidence or extra weight in the tier minimum. The s threshold concerns deployment efficiency relative to Official25; it is not a ratio of search times.

Frozen tuning protocol. The five configurations and their ranking rule were frozen before this campaign's candidate measurements. Stage A evaluates all five on search seed 2026091621 and selects c1 by the lowest four-tier portfolio score over $\{2.5, 3, 3.5, 4\}\times$. Stage B keeps c1 fixed on seeds 2026091622 and 2026091623; no hyperparameter is changed in response to the method comparison. The reported three-seed result includes c1's Stage-A selection run, so it comprises one configuration-selection seed and two additional seeds, not three untouched confirmation seeds. The three operator-diagnostic arms are reported separately and do not enter the baseline performance contrast.

All formal chains use same-card Official25 pre/post timing references; the certified drift range is 0.65–2.37%, below the frozen 5% gate, and the executed-command audit finds no prohibited Fast-SAM3D-only flags.

The grid varies crossover probability, immigrant count, and key-mutation rate. Shared settings are $p_{\mathrm{alloc}} = 0.10$, $\mu = 16$, and $\lambda = 8$; all five Stage-A scores appear below.

Table 20: Preregistered Fair NSGA-II configuration grid and mechanical Stage-A selection on one development seed. The selected configuration is subsequently frozen for the three-seed comparison; this table is configuration selection, not a multi-seed performance claim.

Config	p_{cross}	Immigrants	p_{key}	Mean $R \downarrow$
c1	0.60	2	$1/48$	**1.063801**
c2	0.75	2	$1/48$	1.133463
c3	0.90	1	$1/48$	1.120899
c4	1.00	1	$1/48$	1.109264
c5	0.90	1	$2/48$	1.175778

For the three-seed aggregate comparison, including the Stage-A selection seed, define $\Delta R = R_{\text{PaCO}} - R_{\text{Fair}}$, so positive values favor Fair NSGA-II. At B64 the three values are -0.1522, $+0.0670$, and -0.0460, giving mean -0.043742 and paired 95% CI $[-0.3161, 0.2286]$. Thus the aggregate portfolio difference is unresolved at $n = 3$; the interval does not establish either superiority or equivalence.

Table 18 in the main paper reports the aggressive-tier B64 comparison; the lower-speed region is mixed and remains described here.

The lower-speed region is mixed. At the tighter $\geq 4\times$ deployment threshold, Fair NSGA-II retains its shared B16 bootstrap incumbent in all three seeds, while the B64 PaCO/Pure-Global trajectories replace it with post-bootstrap global proposals. The table below reports this within-trajectory reduction in development Q under the registered logical-slot budget. It is not a search-time acceleration or an additional set of independent replications.

Table 21: Search gain at the $\geq 4\times$ tier from the identical B16 bootstrap to B64. Lower Q is better. Fair NSGA-II retains the shared bootstrap incumbent in all three seeds, while the PaCO/Pure-Global trajectory improves it in all three.

Seed	Shared B16 Q	PaCO/Global B64 Q	Gain	Fair B64 Q
2026091621	0.10546	**0.07396**	**29.9%**	0.10546
2026091622	0.16337	**0.11739**	**28.1%**	0.16337
2026091623	0.07750	**0.06932**	**10.6%**	0.07750

The observed aggressive-tier improvements were supplied by global proposals. This provenance is consistent with the usefulness of continuing global exploration in these runs, but the comparison also changes encoding, variation operators, selection, and duplicate handling. It does not isolate a causal effect of one component, establish equal-compute superiority, or show that archive-guided refinement outperforms Pure Global.

Single-seed operator diagnostics. Three additional arms modify one mechanism of `c1` on seed 2026091621. They are diagnostics, not evolutionary-search baselines: `nodx` removes crossover, `samecell` restricts crossover to parents from the same allocation cell, and `niche` fills the truncated front cell-first before crowding.

Table 22: Fair NSGA-II operator diagnostics on one search seed. "Beats both" counts crossed offspring whose terminal discrepancy is lower than both parents. These arms explain search behavior and are not used for the PaCO-versus-NSGA-II performance claim.

Arm	Duplicate rate	Crossed	Beats both	Pop. cells	Archive cells
fair_c1	0.5000	22	3	7	18
nodx_c1	0.7500	0	0	8	12
samecell_c1	0.6875	8	0	6	17
niche_c1	0.5625	18	5	8	14

Removing crossover or restricting it to one allocation cell raises the duplicate rate and eliminates offspring that beat both parents on this seed. The cell-first niche variant retains cross-cell diversity and produces five such offspring. These diagnostics are consistent with the broader observation that allocation structure changes the effective search neighborhood, but they are not pooled with the three-seed baseline comparison.

At the aggressive $\geq 3.5\times$ and $\geq 4\times$ deployment tiers, PaCO obtains lower terminal discrepancy in all three evaluated seeds. The lower-speed region is mixed, and the aggregate multi-tier portfolio

Table 23: Within-arm search-budget scaling of the frozen historical $\alpha = 0.875$ trajectory on LVIS. Entries are mean best-observed median Q over three search seeds. The diagnostic rows make the B64 selection explicit: "gain captured" is the fraction of the total B16→B128 median-Q reduction already achieved by B64, and "extra gain" is the relative median-Q reduction obtained by doubling the budget from B64 to B128. Tail change reports the relative increase in worst-object/CVaR90 at B128 for the aggressive regimes.

Search budget	$Q@\geq 2\times$	$Q@\geq 2.5\times$	$Q@\geq 3\times$	$Q@\geq 3.5\times$	$Q@\geq 4\times$
B16	0.01162	0.06599	0.06976	0.10255	0.12015
B32	0.00764	0.06007	0.06087	0.06717	0.09525
B64 (deployment)	0.00764	0.04768	0.04848	0.06717	0.09213
B128 (analysis)	0.00623	0.04478	0.04478	0.06093	0.08171
Gain captured by B64	**73.8%**	**86.3%**	**85.2%**	**85.0%**	**72.9%**
Extra median-Q gain at B128	18.5%	6.1%	7.6%	9.3%	11.3%
Tail increase at B128	N/A	N/A	N/A	+139%	+328%

difference remains unresolved at $n = 3$. We therefore use this baseline to show competitive search behavior, especially in the constrained aggressive regime, rather than claim universal superiority over NSGA-II.

D.6 Search-Budget Scaling

We report offline budget scaling separately from between-method comparisons. For one frozen trajectory, its tier readout at prefix B is

$$Q@\geq s \times (B) = \min_{\substack{z \in \mathcal{V}_B \\ \rho^{\mathrm{cert}}(z) \leq s^{-1}}} Q(z). \tag{51}$$

with unattained tiers reported explicitly. The historical $\alpha = 0.875$ prefixes B16, B32, B64, and B128 use fixed recorded objective values and a common final readout basis. Since eligible prefixes are nested, their best feasible Q is non-increasing by set inclusion. This identity is not evidence that one search method is superior, and it need not hold after changing the eligibility or measurement basis between prefixes.

B64 remains the preregistered deployment budget. Descriptively, it captures about 85–86% of the observed B16-to-B128 median-error reduction in the $2.5\times$–$3.5\times$ tiers; B128 adds 6.1–9.3%. The median-selected B128 winners have worse worst-object/CVaR90 values in the aggressive tiers (+139% at $\geq 3.5\times$ and +328% at $\geq 4\times$). This is a limitation of median-only selection, not evidence that more search intrinsically worsens quality: the B64 policies remain available in the larger pool. We preserve the frozen B64 portfolio and use B128 to describe extended-budget behavior without retroactively adding a tail-aware selector.

The frozen B64 search captures 86.3%, 85.2%, and 85.0% of the observed B16→B128 reduction in median discrepancy at the $2.5\times$, $3\times$, and $3.5\times$ tiers. Increasing the budget to B128 continues to improve the median-selected objective, but the additional gain is comparatively modest.

The B128 aggressive winners show worse worst-object/CVaR90 readouts. This does not mean that additional search damages previously discovered policies: B64 policies remain in the larger pool. Rather, it exposes a limitation of the frozen median-only deployment selector. B64 remains the reported deployment budget, while B128 is retained as a search-scaling analysis.

D.7 Exploration–Refinement Sensitivity

The historical mixture parameter α denotes the fraction of continuation proposals assigned to global exploration, with the remainder assigned to archive-guided refinement. Table 24 records the LVIS sweep that produced the frozen $\alpha = 0.875$ B64 schedules later transferred to SA-3DAO. We retain it for provenance and sensitivity analysis, but not as confirmatory evidence that $\alpha = 0.875$ is superior to pure global exploration: the clean paired control summarized under Global exploration and refinement in Section 4.4 supersedes that cross-arm interpretation.

Table 24 reports the B64 sensitivity study over three search seeds. Mean Q is the unweighted arithmetic mean of the five displayed tier-wise mean Q values. The last column is a five-tier normalized portfolio score: within each seed and tier, Q is divided by the best value among the tested α settings, the five tier ratios are averaged, and we finally report mean±sample standard deviation

Table 24: Historical three-seed development sensitivity to the exploration and refinement mixture α at B64. Values are mean±sample standard deviation over search seeds; lower is better. The table documents the configuration-selection process and is not used as evidence that $\alpha = 0.875$ outperforms pure global exploration; the clean paired control summarized in Section 4.4 supersedes that interpretation.

α	$Q@\geq 2\times$	$Q@\geq 2.5\times$	$Q@\geq 3\times$	$Q@\geq 3.5\times$	$Q@\geq 4\times$	Mean $Q\downarrow$	Portfolio $\downarrow$
0.25	0.00644 ± 0.00152	0.04843 ± 0.00542	0.04990 ± 0.00791	0.07772 ± 0.00009	0.09997 ± 0.02678	0.056492	1.0785 ± 0.0949
0.50	0.01529 ± 0.00384	0.04855 ± 0.00584	0.05243 ± 0.01071	0.08285 ± 0.01487	$\mathbf{0.08645 \pm 0.00874}$	0.057114	1.3705 ± 0.2144
0.625	0.00877 ± 0.00301	0.04817 ± 0.00738	0.05039 ± 0.01123	$\mathbf{0.07570 \pm 0.01218}$	0.09497 ± 0.00720	0.055600	1.1297 ± 0.1106
0.75	0.01046 ± 0.00125	0.04812 ± 0.00722	0.04880 ± 0.00674	0.08237 ± 0.01435	0.09508 ± 0.00700	0.056966	1.2027 ± 0.0349
0.875	0.00674 ± 0.00168	$\mathbf{0.04766 \pm 0.00653}$	$\mathbf{0.04806 \pm 0.00722}$	0.07867 ± 0.01378	0.09289 ± 0.01152	**0.054804**	$\mathbf{1.0585 \pm 0.0452}$
1.00	0.01094 ± 0.00321	0.04822 ± 0.00741	0.04848 ± 0.00786	0.08290 ± 0.01489	0.09272 ± 0.00285	0.056652	1.2312 ± 0.1814

across seeds. Lower is better. Because the $2\times$ tier contains a known timing-boundary flip, we also repeat the aggregation over $\{2.5, 3, 3.5, 4\}\times$ only.

Within this historical sweep, $\alpha = 0.875$ has the lowest simple five-tier mean Q (0.054804), the lowest five-tier seed-wise portfolio score (1.0585 ± 0.0452), and the lowest four-tier aggregate after excluding the timing-sensitive $2\times$ tier. These differences are descriptive rather than confirmatory: several exploration-heavy settings are nearly tied, the $\alpha = 0.25$ row was added later as a refinement-heavy addendum, and the campaign spans provenance-separated planner snapshots. Deterministic replay confirms equivalent candidate-generation logic on overlapping arms, but the clean paired control is the appropriate test of $\alpha = 0.875$ versus pure global exploration. That control finds an exact tie at B64 and an unresolved B128 difference, so we use $\alpha = 0.875$ here only to document how the frozen deployment schedules were obtained, not to claim a refinement advantage.

The historical α sweep documents how the frozen B64 deployment trajectories were obtained. Although $\alpha = 0.875$ has the best descriptive aggregate in that sweep, clean paired controls do not establish a universal advantage over pure global exploration. We therefore use the sweep for configuration provenance and sensitivity, not as an independent performance claim for refinement.

E Finding III: PSWA and Backend-Dependent Deployment

The third finding is that infrastructure-level execution cost can change which complete policy is feasible—and therefore optimal—under a fixed deployment budget. We first establish the operator identity of PSWA, then measure its fixed-policy speedup, and finally isolate its effect on budget-conditioned policy selection.

E.1 Packed Execution Preserves Window-Attention Semantics

For sparse windows with valid projected blocks $(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i)$, PSWA concatenates valid tokens,

$$\mathbf{Q} = \mathrm{cat}_i\, \mathbf{Q}_i, \qquad \mathbf{K} = \mathrm{cat}_i\, \mathbf{K}_i, \qquad \mathbf{V} = \mathrm{cat}_i\, \mathbf{V}_i, \tag{52}$$

and records cumulative offsets

$$\mathbf{s} = [0, L_1, L_1 + L_2, \ldots, \sum_i L_i]. \tag{53}$$

Let M_{win} be zero within each original sparse window and $-\infty$ across different windows. Under identical projected queries, keys, values, attention scale, and window membership,

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}; M_{\mathrm{win}}) = \mathrm{cat}_{i=1}^{N}\, \mathrm{Attn}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i; 0). \tag{54}$$

Thus, packed execution changes the layout but not the mathematical window-attention operation. This is an exact-arithmetic identity rather than a bitwise floating-point guarantee for the complete iterative generator.

E.2 Fixed-Policy PSWA Acceleration

Figure 4 visualizes the within-schedule PSWA speedups across the dense and compressed operating points. Table 25 reports the schedule and execution timing decomposition underlying the summary in the main text. All PSWA comparisons hold the schedule and model-evaluation count fixed; the schedule-only contrast changes the frozen schedule under the masked dense reference.

Table 25: Schedule and system latency decomposition on SA-3DAO.

Comparison	Paired median gain	Speedup	Repeat envelope
PSWA @ dense ($A \to B$)	11.78%	1.1699×	1.32%
Schedule only ($A_2 \to C$)	35.10%	1.5773×	1.12%
PSWA @ sparse ($C \to D$)	14.59%	1.1958×	1.48%
Full ($A_2 \to D$)	44.44%	1.8860×	1.48%

A separate PSWA measurement in the component-ablation session, taken at SS13/SLaT3 rather than at any of the four schedules above, gives a warmed $1.1757\times$ with all 100 objects faster. It is reported distinctly because it comes from a different session and schedule placement and is not pooled with Figure 4 or Table 25.

PSWA quality checks compare formal repeat-1 outputs on SA-3DAO against the masked dense reference while holding the schedule and evaluation count fixed. For dense execution, CD changes by +0.77%, F@0.01 by -0.89%, layout 3D-IoU by +0.01%, ADD-S by +0.01%, and ADD-S@0.1 by 0.00%. For the sparse schedule, the corresponding changes are +0.14%, -0.35%, +0.02%, +0.01%, and 0.00%. ICP rotation is reported separately because its variation is larger than the other metrics.

E.3 Fixed-Policy Fidelity

Because PSWA leaves valid tokens, window membership, and sampling policy unchanged, it should not introduce a systematic approximation error. Observed end-to-end metric changes remain small.

These measurements complement rather than replace the operator identity: the proof concerns exact arithmetic within attention, whereas the empirical check captures finite precision and error propagation through the full iterative generator.

E.4 Execution Backend Changes Policy Feasibility

We additionally freeze 15 complete policies from the historical LVIS-8 search pool and remeasure only their execution cost under native sparse execution versus the deployed PSWA configuration. Schedules, seeds, candidate quality values, object set, and model-evaluation counts are fixed. Each GPU card uses its own same-session Official pre/post anchors, and every lane is normalized only by its card-matched anchors. The maximum anchor drift is 2.45%, below the preregistered 5% gate.

Table 26: Budget-conditioned selection under native sparse execution and the deployed PSWA configuration on the same 15 frozen LVIS-8 policies. "Eligible" is the number of candidates satisfying the target speed tier. The preregistered timing-envelope rule resolves the $3\times$ and $3.5\times$ changes; the $4\times$ crossing is retained as unresolved.

Tier	Native eligible	PSWA eligible	Native selected	PSWA selected	Readout
2.0×	11	11	cand05	cand05	same
2.5×	9	11	cand05	cand05	same
3.0×	3	8	cand01	cand05	resolved change
3.5×	0	4	–	cand01	resolved change
4.0×	0	2	–	cand01	unresolved

At $3\times$, cand05 changes from $2.749\times$ under native execution to $3.294\times$ under PSWA and therefore becomes feasible; the selected frozen reference discrepancy changes from cand01's 0.07566 to cand05's 0.04126. This example is same-card (both candidates are on the same measurement card), so it does not rely on cross-card ranking. At $3.5\times$, native execution has no eligible candidate while PSWA admits four. The non-dominated candidate membership is identical under both backends (cand01, cand05, cand12, cand13, cand14), so the supported conclusion is specifically that execution cost changes budget feasibility and budget-conditioned selection, not that the Pareto membership is reordered. This experiment does not rerun search, does not modify the frozen SA-3DAO policies, and does not claim joint optimization over multiple execution backends.

E.5 A Sufficient Condition for Backend-Induced Selection Change

For backend b, define

$$\mathcal{F}_b(\bar{\rho}) = \{z \in \mathcal{V} : \rho_b(z) \le \bar{\rho}\}. \tag{55}$$

Suppose the native selector returns z_0, while a lower-discrepancy policy z_1 satisfies

$$\rho_{\mathrm{PSWA}}(z_1) \le \bar{\rho} < \rho_{\mathrm{native}}(z_1). \tag{56}$$

Then z_1 is excluded under the native backend but admitted under PSWA, and therefore

$$\mathcal{E}(\hat{z}_{\mathrm{PSWA}}(\bar{\rho})) \le \mathcal{E}(z_1) < \mathcal{E}(z_0). \tag{57}$$

This result formalizes why operator-level acceleration can improve selected deployment quality without changing the quality of any individual policy.

E.6 Invariant Execution Reuse

Beyond packed sparse-window attention, the deployment backend reuses execution-invariant quantities whenever valid: cumulative sequence metadata for recurring layouts, masks that are redundant under explicit sequence boundaries, and fixed conditioning projections whose inputs and transformations remain unchanged. These optimizations never reuse evolving denoising features and therefore remain conceptually distinct from diffusion feature caching.

F Additional End-to-End Evaluation

F.1 Additional Quality–Speed Metrics

Figure 7 completes the eight-metric view of Figure 5, adding V-IoU, normal consistency, auxiliary EMD, and ADD-S for the same reported sweep. Individual metrics need not vary monotonically across different schedules: the Orch3D EMD estimates are 0.3259, 0.3233, and 0.3285 for HQ, Balanced, and Aggressive, respectively. Connecting segments do not denote an optimized continuous frontier or evaluations at intermediate speeds.

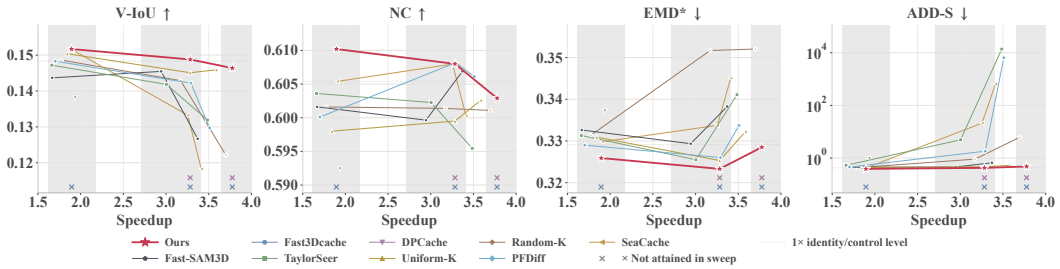

Figure 7: Additional SA-3DAO quality–speed metrics for the same sweep as Figure 5: V-IoU, NC, EMD*, and ADD-S. EMD and ADD-S use inverted axes; ADD-S additionally uses a logarithmic scale to retain the full error range without clipping. TaylorSeer I16 (3.4856×) records ADD-S 13,870, while Orch3D-Aggressive (3.7763×) records 0.4774. These are distinct operating points, not an exact matched-speed pair. Colors, markers, shaded speed regions, and control lines follow the main figure. Crosses denote speed regimes not attained in the tested parameter sweep, not measured speed–quality points.

The unthresholded ADD-S panel reveals error magnitudes that bounded overlap and success scores cannot convey. TaylorSeer I16 has 3D-IoU 0.02233 and ADD-S@0.1 0.02151 alongside its large ADD-S value, complementing the main figure's layout evidence. The sweep also exposes metric-specific trade-offs; it is not used to claim that one Orch3D policy is best in every metric at every speed.

Plotting and source conventions. The figures retain the supplied sweep records and reported speedups; they are not a replot of every representative row in Table 29. The sweep's Uniform-K point near 3.28× uses SS7/SLaT2. Dotted lines use the sweep's identity/control summaries: CD 0.09803, F@0.01 0.14107, V-IoU 0.14833, NC 0.60542, EMD* 0.32913, 3D-IoU 0.32163, ADD-S 0.42434, and ADD-S@0.1 0.48387. These differ from the Official25 row in the representative table.

The panels show point estimates without per-point uncertainty intervals; a noise scale measured for another configuration is not used as a significance test.

Non-attainment in the parameter sweep. Crosses ("Not attained in sweep") mean that no tested configuration of the corresponding baseline reached the marked speed regime in the completed parameter sweep. This is empirical non-attainment under the tested parameter ranges, implementation, and timing protocol, not an unreported measurement or a claim of impossibility outside that scope. Crosses are placed near the panel bottom at target-regime centres solely as status markers; neither coordinate represents a measured operating point. Speed non-attainment is distinct from failing the seven-metric quality-retention gate.

The additional panels report V-IoU, NC, auxiliary EMD, and ADD-S for the same quality–speed sweep shown in the main paper. Individual metrics need not vary monotonically with speed because different schedules trade shape and layout fidelity differently. The connected lines are therefore visual guides rather than a claim of a continuous optimized frontier.

F.2 Extended Qualitative Results

Figures 8–11 extend the main-text qualitative comparison to 16 objects and 10 methods, split into two consecutive eight-object batches for readability. The true-color views show whether accelerated reconstructions remain visually faithful under the common rendering setup, while the matched line views suppress appearance and lighting so that local geometric disagreement is easier to inspect. All four figures use the same object–method grid, camera, and per-object crop windows; each line rendering is therefore a diagnostic re-encoding of its corresponding color rendering rather than a separately reframed visualization. In the line views, black edges are shared with the SAM3D reference, blue edges occur only in the reference, and red edges occur only in the compared method. Isolated blue structure therefore exposes missing reference geometry, whereas isolated red structure exposes extra or displaced geometry. The SAM3D self-reference column is black by construction.

Across the displayed objects, Orch3D remains visually closest to the reference geometry: its line renderings are dominated by shared black contours with comparatively few isolated red or blue structures. The difference is particularly visible on thin or articulated geometry, where aggressive acceleration can easily remove a support, break a rail, or shift a narrow contour. Examples include the sled, bell-lyre, balustrade, shopping cart, forklift, and the wire legs of the battery. Orch3D preserves both the global silhouette and these small structural elements, whereas the accelerated baselines more frequently exhibit missing thin parts, fragmented supports, or locally displaced contours. The red callouts in the color figure mark representative regions where such differences are easiest to see. Together, the two renderings explain the aggregate SA-3DAO trend at the object level: the benefit is not only a globally plausible shape, but better preservation of fine geometry under aggressive sampling reduction.

The 16-object qualitative study uses identical cameras, crop windows, lighting, and rendering protocols across methods. The matched edge-disagreement views make local geometric errors easier to identify: black denotes shared edges, blue reference-only edges, and red method-only edges.

Across the displayed objects, Orch3D generally retains more shared contour structure, particularly for thin supports, rails, frames, and articulated parts that are easily lost under aggressive acceleration.

F.3 Quality-Retention Sensitivity

The main text separates two questions that are otherwise easy to conflate: deployment efficiency at comparable quality and the one-time cost of discovering the deployment policy. For reproducibility, the operating-point labels used in the main paper map to the original DPCache configurations as follows: DPCache-HQ denotes the 25/25 setting and DPCache-Fast denotes k8/8. We use the clean SAM 3D Objects Official25 row as a method-independent quality reference and exclude EMD^* from the primary gate because it is an auxiliary finite-iteration Sinkhorn estimate. The seven primary metrics are CD, F@0.01, V-IoU, NC, 3D-IoU, ADD-S, and ADD-S@0.1.

For a tolerance ϵ, a lower-is-better metric must satisfy

$$m(z) \leq (1+\epsilon) m_{\mathrm{ref}}, \tag{58}$$

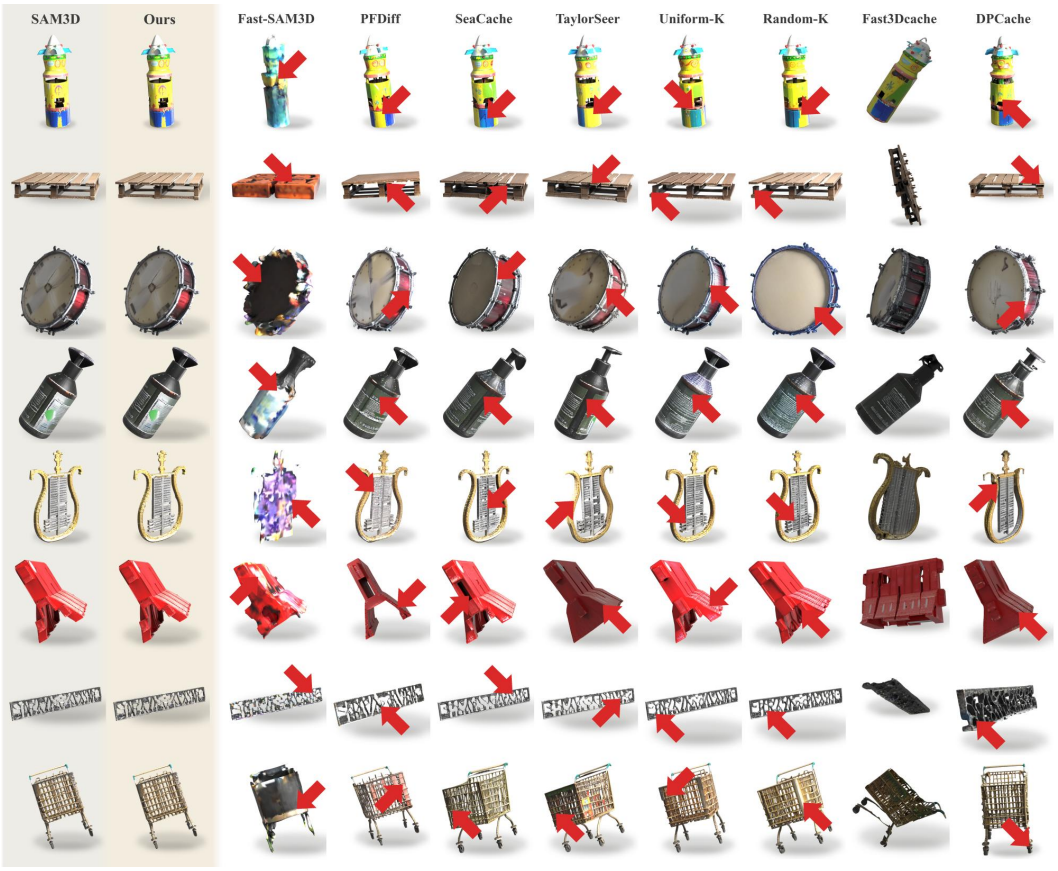

Figure 8: Extended true-color qualitative comparison, batch 1 of 2, across 10 methods. Rows from top to bottom are rocket, sled, snare drum, pushpump, bell-lyre, roadblock, balustrade, and shopping cart. All methods use the same camera, crop window, lighting, contact-shadow setup, and no per-object retuning, so visible differences reflect reconstruction rather than presentation changes. The softly highlighted SAM3D and Orch3D columns aid column tracking. Red circles mark representative regions where accelerated baselines lose, fragment, or displace geometry; the same rows and regions are re-encoded geometrically in Figure 10.



Figure 9: Extended true-color qualitative comparison, batch 2 of 2, continuing the same 10-method layout and rendering protocol as Figure 8. Rows from top to bottom are forklift, Soyuz descent module, battery, umbrella, palm tree, strawberry, mini house, and eagle statue. This second batch broadens the comparison to additional articulated, thin-structure, organic, and compact object geometries while preserving exactly the same crop and column layout. Corresponding edge disagreements are shown in Figure 11.

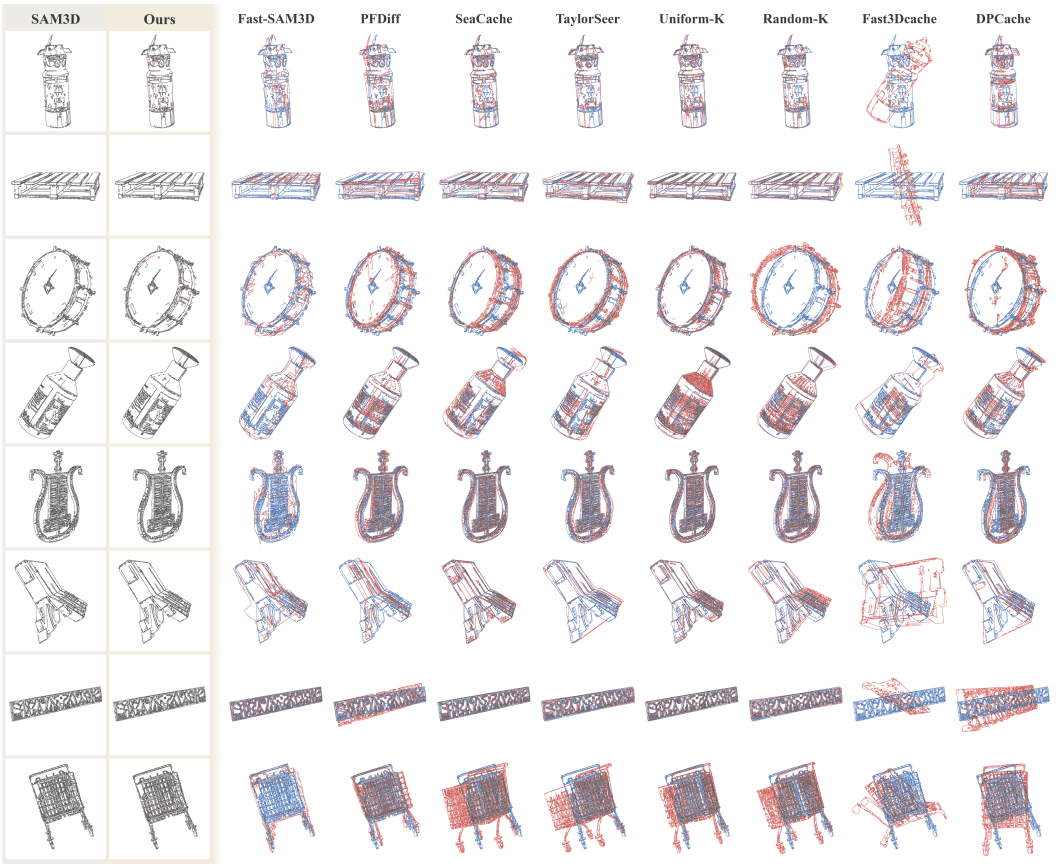

Figure 10: Matched edge-disagreement view for the same eight objects, 10 methods, camera, and crop windows as Figure 8. Black denotes edges shared with the SAM3D reference, blue denotes reference-only edges, and red denotes method-only edges; blue therefore exposes missing geometry and red exposes extra, shifted, or distorted geometry. Orch3D remains dominated by shared black contours across both global silhouettes and fine structures, while accelerated baselines more often show isolated red/blue fragments around rails, supports, frames, and narrow appendages. The SAM3D self-reference column is black by construction.

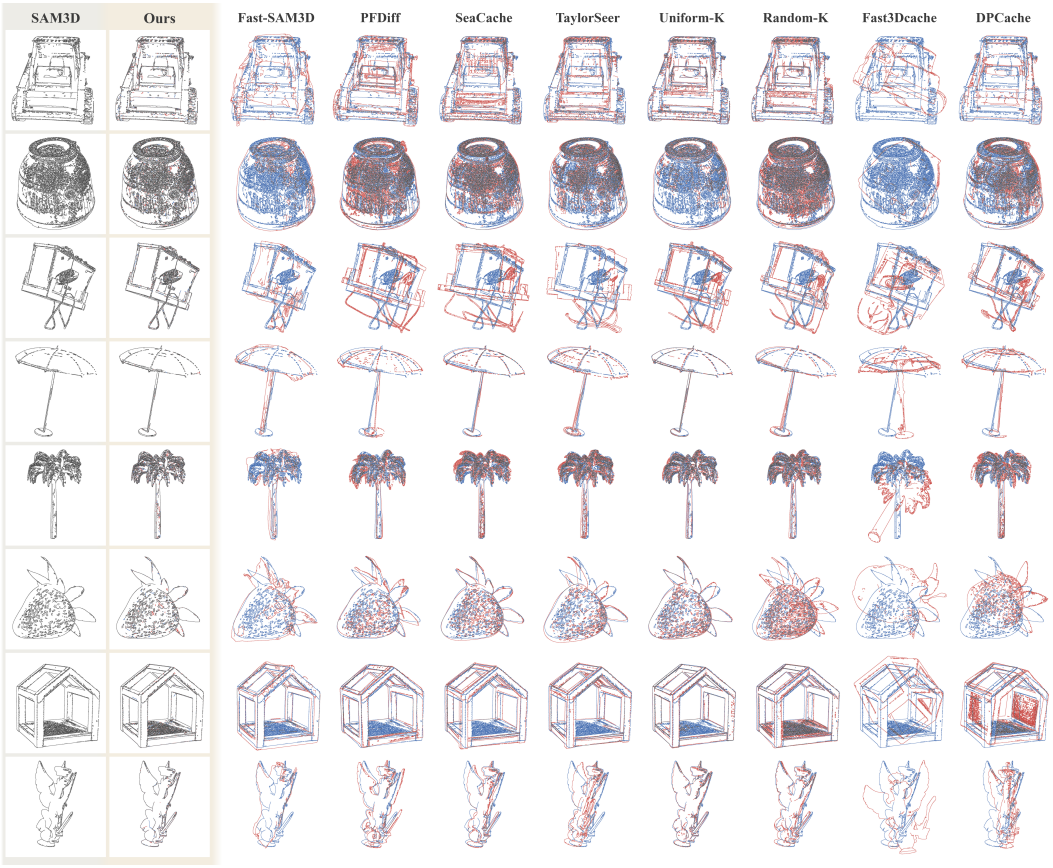

Figure 11: Matched edge-disagreement view for the same eight objects and row order as Figure 9. The color semantics are identical to Figure 10: black denotes shared edges, blue reference-only edges, and red method-only edges. The matched re-encoding makes local geometric disagreement visible even when the coarse object remains recognizable, and provides a direct cross-check of the true-color reconstruction differences in the second batch.

whereas a higher-is-better metric must satisfy

$$m(z) \geq (1-\epsilon)m_{\mathrm{ref}}. \tag{59}$$

A policy is quality-feasible only when all seven inequalities hold simultaneously. For the primary $\epsilon = 0.05$ gate, the corresponding limits are CD ≤ 0.10521, F@0.01 ≥ 0.13528, V-IoU ≥ 0.14079, NC ≥ 0.57057, 3D-IoU ≥ 0.29973, ADD-S ≤ 0.46473, and ADD-S@0.1 ≥ 0.45971.

Table 27: Five-percent quality-retention audit relative to clean SAM 3D Objects. All seven primary non-EMD metrics must pass. Speedup uses $8.729788/T$; cross-method timings are not paired on the same card.

Point	CD	F	V-IoU	NC	3D-IoU	ADD-S	ADD-S@.1	Pass	Time	Speedup
DPCache-HQ	✓	✓	✓	✓	✓	✓	✓	7/7	8.591	1.0162×
DPCache-Fast	✓	×	×	✓	×	×	×	2/7	4.506	1.9374×
Orch3D-HQ	✓	✓	✓	✓	✓	✓	✓	7/7	4.6068	1.8950×
Orch3D-Balanced	✓	✓	✓	✓	✓	✓	✓	7/7	2.6592	3.2829×
Orch3D-Aggressive	✓	✓	✓	✓	×	×	✓	5/7	2.3117	3.7763×

Thus, among the evaluated operating points, the fastest policies satisfying the primary gate are DPCache-HQ at 8.591 s/object ($1.0162\times$) and Orch3D-Balanced at 2.6592 s/object ($3.2829\times$), where both speedups use the same 8.729788 s/object SAM 3D Objects reference. This is deliberately stated as a comparison among evaluated points rather than a claim that either method's full policy space has been exhaustively optimized.

To account for the one-time search cost, let C_m^{search} denote offline GPU-seconds for method m, and let $T_m^*(\epsilon)$ be its fastest evaluated deployment point that passes the quality gate. For N deployments,

$$C_m(N;\epsilon) = C_m^{\mathrm{search}} + N\,T_m^*(\epsilon), \tag{60}$$

or, per object,

$$\bar{C}_m(N;\epsilon) = \frac{C_m^{\mathrm{search}}}{N} + T_m^*(\epsilon). \tag{61}$$

The corresponding break-even scale is

$$N^*(\epsilon) = \frac{C_{\mathrm{PaCO}}^{\mathrm{search}} - C_{\mathrm{DP}}^{\mathrm{search}}}{T_{\mathrm{DP}}^*(\epsilon) - T_{\mathrm{Orch3D}}^*(\epsilon)}. \tag{62}$$

For the B64 PaCO search trajectory, $C_{\mathrm{PaCO}}^{\mathrm{search}} = 5.3$ GPU-hours $= 19{,}080$ GPU-s. The source-faithful DPCache port records 483 GPU-s for dense LVIS calibration on one RTX 4090; because its dynamic-programming step is not separately instrumented, this is a lower bound on total DPCache offline cost. At $\epsilon = 0.05$, substituting 8.591 and 2.6592 s/object gives $N^* \approx 3{,}135$ objects. If DPCache is instead granted an optimistic zero offline cost, the crossing moves only to approximately $3{,}217$ objects.

Using all seven primary non-EMD metrics, the fastest evaluated Orch3D policy changes naturally with the allowed quality tolerance: HQ at 2%, Balanced at 5%, and Aggressive at 10%. This behavior is precisely what the PaCO repertoire is designed to support—one offline search provides policies for multiple deployment requirements.

G Offline Search Cost and Deployment Amortization

PaCO search is performed offline and its cost is amortized over future generation requests. Let C_m^{search} be the offline GPU cost of method m, and $T_m^*(\epsilon)$ its fastest evaluated policy satisfying quality tolerance ϵ. For N deployments,

$$C_m(N;\epsilon) = C_m^{\mathrm{search}} + NT_m^*(\epsilon). \tag{63}$$

The break-even scale between PaCO-based Orch3D and DPCache is

$$N^*(\epsilon) = \frac{C_{\mathrm{PaCO}}^{\mathrm{search}} - C_{\mathrm{DP}}^{\mathrm{search}}}{T_{\mathrm{DP}}^*(\epsilon) - T_{\mathrm{Orch3D}}^*(\epsilon)}. \tag{64}$$

The qualitative conclusion is therefore not tied to a single 3D-IoU threshold: over the tested 2–10% multi-metric retention range, the accelerated DPCache-Fast point remains infeasible, while Orch3D moves from HQ to Balanced to Aggressive as the allowed degradation is relaxed. The resulting break-even remains on the order of a few thousand deployments. This analysis uses the reported SA-3DAO

Table 28: Sensitivity of the fixed-quality accounting to the allowed per-metric degradation. Each row requires all seven primary metrics to satisfy the stated tolerance relative to Official25. Break-even uses the measured 19,080 GPU-s PaCO B64 search cost and the 483 GPU-s DPCache calibration cost.

Tolerance	DPCache fastest feasible	Orch3D fastest feasible	N^*
2%	DPCache-HQ, 8.591 s (1.0162×)	HQ, 4.6068 s (1.8950×)	4,668
5%	DPCache-HQ, 8.591 s (1.0162×)	Balanced, 2.6592 s (3.2829×)	3,135
10%	DPCache-HQ, 8.591 s (1.0162×)	Aggressive, 2.3117 s (3.7763×)	2,962

timings; DPCache and Orch3D were not measured as a paired same-card timing experiment, so the numerical crossing should be interpreted as a reported-timing amortization estimate rather than a hardware-normalized constant.

At the 5% seven-metric retention gate, the measured B64 PaCO search cost of 5.3 GPU-hours gives an estimated break-even point of approximately 3,135 deployed objects using the recorded DPCache calibration cost. Granting DPCache zero offline cost moves the estimate only modestly to about 3,217 objects.

These values are deployment-scale estimates rather than hardware-independent constants because the cross-method timings are not paired on the same GPU.

H Additional Search Diagnostics

This section retains secondary diagnostics that are useful for reproducibility and mechanism analysis but are not required for the main claims.

H.1 Evidence-Gated Exploration–Refinement Controller

The evidence-gated variant is an auxiliary controller diagnostic and is not the configuration used to produce the reported SA-3DAO operating points. We retain it to document alternative allocation of proposal budget between global exploration and refinement.

H.2 Generational Recombination Diagnostic

This study tests whether strong stage-wise components remain strong after cross-policy recombination. Because its realized mutation rate is affected substantially by duplicate and legality repair, and one timing basis fails the registered drift gate, it is used only as operator-level mechanism evidence. Fair NSGA-II remains the formal evolutionary-search baseline.

I Reproducibility and Provenance

I.1 Schedule Provenance

All reported frozen schedules are stored as explicit endpoint lists and canonical policy keys. Historical exploratory schedules are retained for provenance but are not substituted for the clean baselines reported in the paper.

I.2 Timing and Hardware Provenance

Each controlled timing study uses internally consistent GPU-specific reference measurements. Timing bases are never mixed across cards inside a paired comparison. Historical studies retain their original frozen timing basis rather than importing a later denominator.

I.3 Audit Reconstruction Notes

The fresh joint-versus-factorized campaign retains worker, checkpoint, timing, and observation receipts. The Independent-arm audit launcher was missing from the original runtime bundle; the

Table 29: Representative SA-3DAO operating points across different speed regimes. Speedup is computed uniformly as $8.729788/T$ using the common SAM 3D Objects reference. Orch3D schedules are selected on LVIS and transferred without SA-3DAO retuning. This table summarizes representative operating points rather than a strictly matched-speed ranking; quality–speed trade-offs across nearby speed regimes are shown in Figure 5.

Method	Shape					Pose			Efficiency	
	CD ↓	F@0.01 ↑	V-IoU ↑	NC ↑	EMD* ↓	3D-IoU ↑	ADD-S ↓	ADD-S@0.1 ↑	Time (s)↓	Speedup ↑
SAM 3D Objects	0.1002	0.1424	0.1482	0.6006	0.3334	0.3155	0.4426	0.4839	8.729788	1.0000×
Uniform-K	0.0992	0.1445	0.1504	0.5980	0.3309	0.3176	0.4327	0.4839	4.7347	1.8438×
Random-K	0.0992	0.1420	0.1486	0.6016	0.3314	0.3060	0.4274	0.4444	4.9157	1.7759×
Fast3Dcache	0.0974	0.1412	0.1488	0.6066	0.3269	0.3216	0.4243	0.4839	8.7110	1.0022×
TaylorSeer	0.1011	0.1417	0.1472	0.6036	0.3313	0.2704	0.5303	0.4301	5.2495	1.6630×
Fast-SAM3D	0.1015	0.1387	0.1437	0.6016	0.3326	0.3120	0.4488	0.4839	5.2331	1.6682×
PFDiff	0.0998	0.1431	0.1483	0.6001	0.3290	0.2786	0.4545	0.4194	5.1260	1.7030×
SeaCache	0.0984	0.1429	0.1514	0.6054	0.3298	0.3119	0.4356	0.4839	4.5740	1.9086×
DPCache	0.1039	0.1335	0.1381	0.5924	0.3372	0.1323	0.9471	0.1505	4.5060	1.9374×
Orch3D-HQ	**0.0967**	**0.1478**	**0.1517**	**0.6102**	0.3259	**0.3217**	**0.3871**	**0.4839**	4.6068	1.8950×
Orch3D-Balanced	0.0971	0.1438	0.1488	0.6080	**0.3233**	0.3176	0.4244	0.4731	2.6592	3.2829×
Orch3D-Aggressive	0.0980	0.1355	0.1464	0.6029	0.3285	0.2896	0.4774	0.4624	**2.3117**	**3.7763**×

*Auxiliary EMD estimate. **See Appendix** A: EMD definition.

byte-identical frozen audit script was subsequently rerun read-only over the retained receipts. No inference, timing, observation, or search artifact was regenerated or modified.

I.4 Reproducibility Summary

The released artifact records complete policy identities, proposal provenance, objective values, timing receipts, search histories, and frozen deployment selections. Together with the explicit PaCO proposal logic, metric definitions, and PSWA execution protocol above, these records allow the reported search and deployment decisions to be reconstructed without rerunning the optimization.

J Full Main-Text Results and Ablation Detail

This section carries the full quantitative material that the abbreviated main text reports only in summary form: the complete representative operating-point table, the preregistered ablations, and the per-metric sensitivity readouts.

J.1 Ablation Studies

At 4G+4R/B128, allocation conditioning expands refinement coverage from 3.83 to 18.67 parent cells, increases H_{norm} from 0.349 to 0.866, and reduces the four-tier portfolio score J from 1.0869 to 1.0054. **See Appendix** D.3, Tables 15 and 16: six-seed memory and parent-coverage diagnostics.

Refinement. Allocation-conditioned dual-anchor refinement preserves access to substantially more compute-allocation regimes than global parent pooling, providing structured local search around both fidelity- and latency-oriented anchors. **See Appendix** D.6, Table 23: budget scaling; Appendix D.7: G/R sensitivity.

PSWA. At fixed schedules, PSWA yields $1.1209\times$–$1.1616\times$ speedup. The same backend change shifts the $3\times$ frozen-pool selection from SS7/SLaT3 to SS12/SLaT3 and enables a $3.5\times$ feasible policy, while non-dominated membership is unchanged. Fixed-schedule quality changes remain small: for dense/sparse execution, CD changes by $+0.77\%/+0.14\%$, F@0.01 by $-0.89\%/-0.35\%$, and 3D-IoU by $+0.01\%/+0.02\%$, respectively. **See Appendix** E.2, Fig. 4, and Appendix E.4, Table 26.

Table 30: Deployment validation. (a) Seven-metric retention within 5% of clean SAM 3D Objects. (b) Relative PSWA quality changes with schedules and evaluation counts fixed.

(a) 5% retention audit

Point	Pass	Time (s) ↓	Speedup ↑
DPCache-HQ	**7/7**	8.5910	1.0162×
DPCache-Fast	2/7	4.5060	1.9374×
Orch3D-HQ	**7/7**	4.6068	1.8950×
Orch3D-Balanced	**7/7**	**2.6592**	**3.2829**×
Orch3D-Aggressive	5/7	2.3117	3.7763×

(b) PSWA fidelity check

Metric	Dense	Sparse
CD	+0.77%	+0.14%
F@0.01	−0.89%	−0.35%
3D-IoU	+0.01%	+0.02%
ADD-S	+0.01%	+0.01%
ADD-S@0.1	0.00%	0.00%

Execution cost changes deployment choice. To isolate this link, we hold 15 LVIS-8 policies and their recorded quality values fixed and remeasure only execution cost (Appendix E.4). At the $3\times$ target, a lower-discrepancy policy moves from $2.749\times$ under native execution to $3.294\times$ with PSWA; the selected frozen Q consequently changes from 0.07566 to 0.04126. This is a same-card threshold crossing, not a quality improvement of an unchanged schedule. Non-dominated membership remains identical: PSWA makes a previously over-budget policy eligible, allowing execution savings to support a more faithful schedule. No search is rerun in this diagnostic, so it isolates budget-conditioned selection from candidate discovery.

Quality-retaining deployment. Under a 5% retention gate on seven primary SA-3DAO metrics (all except auxiliary EMD*), Orch3D-Balanced is the fastest evaluated point passing all seven metrics at $3.2829\times$. The accelerated DPCache-Fast point reaches $1.9374\times$ but passes only 2/7 metrics. Table 6b jointly reports the retention audit and the fixed-schedule PSWA fidelity check; Appendix G gives the per-metric thresholds, 2–10% sensitivity, and offline-cost accounting.

Budget and deployment sensitivity. The frozen B64 budget captures 85–86% of the observed B16-to-B128 median-Q reduction at $2.5\times$–$3.5\times$ (Table 31). B128 improves this objective further, but its median-selected winners have worse worst-object/CVaR90 readouts in the aggressive tiers. Additional search does not inherently harm quality: the B64 policies remain in the larger pool, while median-only selection can choose different winners. B64 remains the preregistered deployment budget; B128 describes extended-budget behavior rather than a retuned deployment rule. Nested-prefix gains do not establish superiority over other search algorithms; Appendix D.6 reports the complete prefix and tail diagnostics.

Table 31: Budget and retention sensitivity. (a) Three-seed mean best-observed $Q \downarrow$ and relative budget gains. (b) Fastest evaluated policies passing all seven metrics at each tolerance; N^* gives estimated break-even deployments.

(a) Search-budget scaling

Budget	$Q_{\geq 2.5\times}$	$Q_{\geq 3.0\times}$	$Q_{\geq 3.5\times}$	$Q_{\geq 4.0\times}$
B16	0.0660	0.0698	0.1026	0.1202
B32	0.0601	0.0609	0.0672	0.0953
B64	0.0477	0.0485	0.0672	0.0921
B128	**0.0448**	**0.0448**	**0.0609**	**0.0817**
B64 captured	**86.3%**	**85.2%**	**85.0%**	**72.9%**
B128 extra	6.1%	7.6%	9.3%	11.3%

(b) Retention sensitivity

Tol.	Point	Speedup	N^*
2%	DPCache-HQ	1.016×	4,668
	Orch3D-HQ	1.895×	
5%	DPCache-HQ	1.016×	3,135
	Orch3D-Bal.	3.283×	
10%	DPCache-HQ	1.016×	2,962
	Orch3D-Aggr.	3.776×	

When offline search pays off. Search is a one-time cost rather than part of each object's inference. At the 5% seven-metric retention gate, the recorded 5.3 GPU-hour B64 search and 483 GPU-s DPCache calibration give an estimated break-even point of 3,135 objects; granting DPCache zero offline cost changes it to 3,217. The 483 GPU-s measurement excludes separately uninstrumented dynamic programming and is a lower bound, not the complete offline cost. These crossings use reported, non-paired cross-method timings and evaluated quality-feasible policies; they are deployment-scale estimates, not hardware-independent constants (Appendix G). Across 2–10% tolerances, the fastest feasible Orch3D point changes from HQ to Balanced to Aggressive, whereas DPCache-HQ remains selected.

Mechanism decomposition. The refinement-active B128 study separates allocation-conditioned memory from the latency-anchor factor, while the PSWA decomposition separates schedule compression from packed execution. The memory effects have the preregistered favorable direction in 5/6 seeds but confidence intervals cross zero; they are therefore mechanism evidence rather than standalone significance claims. Table 32 summarizes the compact readouts, with full protocols in Appendix D.3 and Appendix E.2.

Table 32: Mechanism decomposition. Left: B128 factor effects (six seeds) and B64 refinement-pool stress (three seeds); all 95% CIs cross zero and negative ΔJ is favorable. Right: schedule and PSWA timings from a separate SA-3DAO session.

PaCO memory / refinement				**PSWA / schedule decomposition**			
Factor	Mean ΔJ	95% CI	Fav.	Comparison	Median gain	Speedup	Envelope
Alloc. memory	**-0.0574**	[-0.1173, 0.0025]	**5/6**	PSWA, dense	11.78%	1.1699×	1.32%
Latency anchor	**-0.0352**	[-0.0786, 0.0083]	**5/6**	Schedule only	35.10%	1.5773×	1.12%
Interaction	-0.0483	[-0.1248, 0.0283]	–	PSWA, sparse	14.59%	1.1958×	1.48%
Refine-pool stress	-0.0880	[-0.2035, 0.0275]	**3/3**	Full stack	**44.44%**	**1.8860**×	1.48%

Parent coverage and portfolio quality answer different questions. Because local mutation preserves allocation, refinement can revisit a compute regime only when its parent pool retains that allocation. Global proposals introduce new cells; cell-conditioned anchors keep existing cells available even when their parents are not globally competitive. This does not guarantee that any particular offspring improves the deployment portfolio. In the six-seed factorial, only 1/101 latency-parent offspring in the full arm becomes a new online tier incumbent (Appendix D.3), so the anchor effect should not be attributed solely to direct replacements. These observations distinguish search support from realized gains: memory keeps allocation regimes accessible, while complete-rollout feedback determines whether their descendants improve the deployment portfolio.

K Discussion and Limitations

Our results expose two forms of coupling. On SAM 3D Objects, upstream schedules change downstream quality preferences, motivating complete-policy search. On TRELLIS, equal realized evaluation counts can yield different end-to-end latency when compute is redistributed across stages. PaCO addresses policy coupling, while PSWA captures sparse-structure-dependent cost. The allocation-memory evidence is stronger as a mechanism result than as a universal performance claim: allocation conditioning substantially broadens parent-cell support and gives favorable mean ΔJ, but the six-seed confidence intervals still cross zero. We therefore interpret allocation-conditioned memory as preserving structured refinement support rather than claiming superiority to global search at every budget. Likewise, PSWA has an exact-arithmetic operator identity for the fixed window partition, while finite-precision and whole-pipeline fidelity remain empirical and are checked separately. Our evidence is limited to two 3D generation systems and fixed model weights; broader architectures, model families, and hardware remain future work.

L Evidence-gated allocation: implementation details

The evidence-gated variant starts from the shared 16-candidate bootstrap and then processes batches of eight candidates. Its target speedup tiers are $\mathcal{S} = \{2.5, 3, 3.5, 4\}$. Initially, every proposal comes from E_G. After two consecutive exploration batches without a strict improvement in any tier incumbent, the next batch assigns one evaluation to E_R and the remaining evaluations to E_G. This probe batch uses the same 7:1 global-to-refinement allocation as the historical fixed $\alpha = 0.875$ configuration. All proposals in a batch use the history available before that batch begins.

After the batch completes, refinement is retained for the next batch only if its candidate is the selected tier winner and that tier's quality strictly improves over the pre-batch incumbent. First attainment of an otherwise infeasible tier also counts as progress. Winner selection uses $(Q, \rho_{\mathrm{gate}}, \mathrm{id})$, where ρ_{gate} is the normalized timing field available to this controller under its frozen protocol. It includes the same batch's global candidates and does not import a future denominator. A tie in error with the previous incumbent is not an improvement. If only global proposals produce progress, or if no

tier improves, the next batch contains only global proposals. The stagnation counter resets after an exploration-batch improvement and on transitions into or out of refinement. The fixed-mixture variant instead reserves the same refinement quota in every continuation batch.

Algorithm 3 Evidence-gated emitter allocation used by the implemented controller.

state ← EXPLORE; stall ← 0; tiers $\mathcal{S} \leftarrow \{2.5, 3, 3.5, 4\}$
while search budget remains **do**
  if state = EXPLORE **then**
    $(m_G, m_R) \leftarrow (8, 0)$
  else
    $(m_G, m_R) \leftarrow (7, 1)$
  end if
  generate and fully evaluate one batch using Algorithm 1
  for each $s \in \mathcal{S}$, recompute the feasible incumbent by $(Q, \rho_{\mathrm{gate}}, \mathrm{id})$
  improved ← true iff at least one tier is newly attained or has a strict Q decrease
  if state = EXPLORE **then**
    stall ← 0 if improved, otherwise stall $+1$
    if stall ≥ 2 **then**
      state ← PROBE; stall ← 0
    end if
  else
    useful ← true iff the batch's single refinement is itself the new incumbent of an improved tier
    state ← PROBE if useful, otherwise EXPLORE; stall ← 0
  end if
end while

L.1 GENERATIONAL MULTI-OBJECTIVE RECOMBINATION: PROTOCOL AND OPERATOR REALIZATION

We additionally study whether strong stage-wise components remain strong after being recombined across complete policies. This configuration deliberately uses several standard NSGA-II ingredients, including nondominated sorting, crowding distance, binary tournament parent selection, and elitist replacement over parents and offspring, but it is used here as a controlled mechanism diagnostic rather than as a standard NSGA-II baseline. All candidates are scored after a complete SS, SLaT, and decoder rollout with the same terminal-discrepancy and normalized-latency evaluator and the same B64 total candidate budget used by the paired search analysis.

Table 33: Protocol for the controlled generational multi-objective recombination diagnostic. The table distinguishes requested operators from the realized operators after duplicate/legality repair.

Component	Frozen setting
Initial measured set	Same B16 bootstrap as PaCO (16 complete policies)
Population size	$\mu = 8$ after environmental selection
Offspring per generation	$\lambda = 8$
Generations after B16	6 (48 post-bootstrap offspring; B64 total)
Objectives	Terminal discrepancy Q and online normalized latency ρ^{on}
Parent selection	Rank + crowding binary tournament
Environmental selection	NSGA-II-style nondominated sorting + crowding on parent∪offspring
Crossover	Cross-stage SS/SLaT swap, requested $p_{\mathrm{cross}} = 1.0$
Mutation	Requested $p_{\mathrm{mut}} = 0.10$, one allocation-preserving local edit
Random immigrants	0
Duplicate / invalid child	Deterministic repair; if needed, mutation is forced to obtain a legal unseen policy
Evaluation unit	Complete SS, SLaT, and decoder policy; no stage-local proxy score
Deployment readout	Same tier selector as PaCO, applied to all 64 evaluated policies

The initial B16 archive is identical to PaCO. After one $\mu = 8$ environmental truncation, the search produces 48 post-bootstrap offspring over six generations. Because duplicate or invalid recombinations

are repaired rather than discarded, the requested weak-mutation rate and the realized mutation rate are not the same; we therefore report both and interpret mutation-strength effects only descriptively.

Table 34: Operator realization and mechanism readout on LVIS. ΔQ is child quality minus the mean quality of its two parents, so positive values are worse. "Improve both" is the fraction of pure recombination children whose quality is better than both parents. Population and pool cells are counted at B64; the last column shows requested versus realized mutation rates after repair.

Seed	Pure recomb.	Improve both	Mean ΔQ	Pop. cells	PaCO pool cells	Mut. req./real.
2026091621	22	13.6%	+0.0126	5	12	0.188 / 0.542
2026091622	29	17.2%	+0.0090	5	13	0.083 / 0.396
2026091623	22	9.1%	+0.0108	5	10	0.125 / 0.542

The child-versus-parent error pattern appears in all three seeds of this diagnostic. It does not alone prove cross-stage incompatibility: selected multi-objective parents need not be jointly improvable in error. The B16 bootstrap spans eight cells and the B64 population retains five; PaCO's parent pool spans 10–13 cells. Because the generational population has only eight slots, this is not a capacity-matched diversity test. The operator trace further shows that the requested weak-mutation rate is not the realized one: duplicate/invalid offspring trigger 17, 15, and 20 collision repairs, raising the realized mutation rates to 0.542, 0.396, and 0.542. We therefore interpret the experiment primarily through the recombination and diversity mechanisms, not as a clean mutation-strength comparison.

As a separate stress diagnostic, the tier-gated portfolio contrast is directionally consistent with the mechanism readout: defining $\Delta J = J_{\mathrm{PaCO}} - J_{\mathrm{gen}}$, the three B64 values are -0.3907, -0.1955, and -0.2027 (mean -0.2630, paired 95% CI $[-0.5380, 0.0120]$). The third seed is split across two GPUs and one of those timing bases fails the preregistered 5% Official25 drift gate (maximum post/pre drift 21.0%, versus 4.25% on the companion GPU and 1.29% on the donor GPU). We therefore retain this experiment only as operator-level mechanism evidence. The certified Fair NSGA-II campaign in Appendix D.5 is the paper's performance comparison to evolutionary search.

M Frozen full-state gate policy

The full-state gate is applied after local mesh and topology checks. The current frozen thresholds are: at most 5% increase in mean local geometry risk; no increase in mesh-invalid, topology-failure, or component-failure rate; worst paired rotation regression at most 30°; worst paired log-scale regression at most $\log(1.10)$; worst normalized translation regression at most 0.10; and worst scene-state regression at most 0.10. We retain the raw constituent metrics even when the aggregated scene-state term is used for gating.

N Additional protocol notes

Transition-level full-state sensitivity is a proposal diagnostic rather than an additive model of path risk. A complete sparse path changes the state presented to later macro intervals, so an edge measured from a teacher prefix need not have the same effect inside another sparse path. These historical transition-level diagnostics are separate from PaCO's candidate objective, which is evaluated only through complete-policy rollouts.

Fresh-search audit provenance. The fresh Joint-vs-Factorized experiment preserved complete worker, timing, observation, and checkpoint receipts for all six arms. The Joint audit ran in-line and passed for all three seeds. For the Independent arm, the arm-specific audit script was accidentally omitted from the deployed runtime bundle, so the original in-line audit launcher exited before executing the audit; those exit receipts are preserved. We restored the byte-identical frozen audit script and reran it read-only over the retained receipts. The reconstructed audits pass for all three Independent seeds (956/919/907 rows checked) and independently reproduce the registered 30/60/90-minute UNATTAINED results. No inference, observation, timing, or search artifact was rerun or modified.

O STRUCTURAL ANALYSIS OF COMPLETE-POLICY SEARCH

This section derives the approximation, proposal-support, and deployment-selection properties used in Section 3. The arguments connect upstream-conditioned quality to complete-policy evaluation, allocation-preserving neighborhoods, and latency-constrained readout. All quality values belong to one fixed evaluation task and aggregation rule; constructed examples are distinguished from empirical observations.

O.1 PREFERENCE REVERSAL AND ADDITIVE APPROXIMATION

Proposition 1 (Additive approximation floor). Let $\Delta_a = Q(a,v) - Q(a,u) > 0$ and $\Delta_b = Q(b,u) - Q(b,v) > 0$ be the reversal margins in Eq. (2). For any additive score $\widetilde{Q}(s,t) = f(s) + g(t)$,

$$\max_{(s,t)\in\{a,b\}\times\{u,v\}} \left|Q(s,t) - \widetilde{Q}(s,t)\right| \geq \frac{\Delta_a + \Delta_b}{4}. \tag{65}$$

Any upstream-independent fixed choice between u and v also incurs regret of at least $\min(\Delta_a, \Delta_b)$ under one of the two upstream policies.

Proof of Proposition 1. Let $\mathcal{W} = \{a,b\} \times \{u,v\}$, and let the positive reversal margins be those in Proposition 1. For an arbitrary additive score $\widetilde{Q}(s,t) = f(s) + g(t)$, define $r(s,t) = Q(s,t) - \widetilde{Q}(s,t)$. The additive terms cancel in the four-point contrast, giving

$$r(a,v) - r(a,u) + r(b,u) - r(b,v) = \Delta_a + \Delta_b. \tag{66}$$

The triangle inequality therefore yields

$$\Delta_a + \Delta_b \leq 4 \max_{(s,t)\in\mathcal{W}} |r(s,t)|, \tag{67}$$

which proves Eq. (65). The bound is tight on this four-entry domain when f and g are unrestricted real-valued functions. Set $d = (\Delta_a + \Delta_b)/4$, choose residuals $r(a,v) = r(b,u) = d$ and $r(a,u) = r(b,v) = -d$, and subtract them from Q. The resulting matrix has zero four-point contrast and hence an additive representation. This tightness statement concerns only $\mathcal{W}$, not approximation on an entire schedule space.

Consequence for a fixed downstream choice. For $t \in \{u,v\}$, define conditional regret relative to the better of the same two downstream schedules as

$$R(s,t) = Q(s,t) - \min_{t'\in\{u,v\}} Q(s,t'). \tag{68}$$

Choosing u incurs regret Δ_b under b, while choosing v incurs regret Δ_a under a. Thus,

$$\min_{t\in\{u,v\}} \max_{s\in\{a,b\}} R(s,t) = \min(\Delta_a, \Delta_b) > 0. \tag{69}$$

This statement assumes a deterministic downstream choice independent of the upstream policy. A planner that conditions on upstream state need not obey that restriction. Complete-policy search resolves the dependence during offline selection; it does not require changing the deployed schedule for each input. Neither this regret identity nor the approximation floor establishes that one particular complete-policy optimizer is optimal.

Measurement-error conditions. The preceding identities apply to fixed objective values. If observed values $\widehat{Q}$ satisfy a deterministic bound $|\widehat{Q}(s,t) - Q(s,t)| \leq \varepsilon$ on all four entries, then each observed reversal margin greater than 2ε certifies its corresponding sign. For positive observed margins $\widehat{\Delta}_a, \widehat{\Delta}_b$, every additive score also obeys

$$\max_{(s,t)\in\mathcal{W}} |Q(s,t) - \widetilde{Q}(s,t)| \geq \left[\frac{\widehat{\Delta}_a + \widehat{\Delta}_b}{4} - \varepsilon\right]_+. \tag{70}$$

This is a conditional perturbation bound, not a claim that empirical repeat ranges are deterministic error certificates. We do not assign a numerical ε here or convert the existing repeat-envelope tests into confidence guarantees.

O.2 Allocation subspaces and refinement support

Fix stage m to K_m intervals with endpoint indices 0 and T_m. A legal ordered path is determined by choosing $K_m - 1$ distinct internal indices from $T_m - 1$ possibilities. Thus an allocation cell c has placement-domain size

$$|\mathcal{Z}_c| = \prod_{m=1}^{M} \binom{T_m - 1}{K_m - 1}. \tag{71}$$

This counts the legal placement domain before deduplication or proposal restrictions; it is not an evaluation-complexity guarantee for the search.

For a parent z, let $\mathcal{N}_R(z)$ contain the legal children obtainable by the frozen allocation-preserving placement mutation. Then

$$\mathcal{N}_R(z) \subseteq \mathcal{Z}_{\phi(z)}. \tag{72}$$

For any available parent pool $\mathcal{P}$, it follows that

$$\bigcup_{z \in \mathcal{P}} \mathcal{N}_R(z) \subseteq \bigcup_{c \in \phi(\mathcal{P})} \mathcal{Z}_c. \tag{73}$$

Repeated refinement cannot create a new allocation if neither global proposals nor fallback parents introduce it. The recorded legal-neighbor fallback may switch the requested stage, edit count, or parent; Eq. (73) uses the full pool of permissible realized parents, not only the initially requested parent. Deduplication and exhausted neighborhoods can remove available children, so the inclusion need not be an equality.

Allocation-conditioned anchors retain one or two distinct representatives per occupied cell before legal-neighbor filtering. Let $\mathcal{C}_{\text{obs}} = \{c : \mathcal{H}_c \neq \varnothing\}$ and let $\mathcal{P}_{\text{anchor}}$ be the union of the deduplicated extrema. Since each observed cell contributes at least one and at most two distinct identities,

$$|\mathcal{C}_{\text{obs}}| \leq |\mathcal{P}_{\text{anchor}}| \leq 2|\mathcal{C}_{\text{obs}}|. \tag{74}$$

This bounds only the parent pool, not the full recorded history, Pareto storage, or rollout cost. The implementation shuffles anchor identities, so cells with two distinct anchors need not have the same selection probability as cells with one. A global-extrema pool can represent fewer cells, restricting the subspaces available for refinement. This is a statement about proposal support, not uniform cell sampling or an improvement guarantee: a retained cell can still contain only poor unseen policies. Likewise, dominance of a parent constrains its recorded objective vector, not the unknown objective vectors of its descendants. MAP-Elites/MOME provide the established descriptor-indexed archive principle (Mouret & Clune, 2015; Pierrot et al., 2022); PaCO applies it to the allocation-preserving neighborhoods of complete sampling policies.

O.3 Finite-pool sufficiency of the Pareto repertoire

Fix a finite protocol-eligible pool $\mathcal{V}$ with one objective vector per canonical policy and a total order on keys $k(z)$. Recompute the cell fronts using exactly this pool and its audited (Q, ρ) values, and let

$$\mathcal{A}^{\mathcal{V}} = \bigcup_{c} \text{ND}_{(Q,\rho)}(\mathcal{V} \cap \mathcal{Z}_c). \tag{75}$$

Let $\text{Sel}(\mathcal{U}, \bar{\rho})$ return the feasible lexicographic minimum of (Q, ρ, k) in $\mathcal{U}$, or $\perp$ if none exists. For every budget,

$$\text{Sel}(\mathcal{V}, \bar{\rho}) = \text{Sel}(\mathcal{A}^{\mathcal{V}}, \bar{\rho}). \tag{76}$$

To prove this, suppose the winner in $\mathcal{V}$ were removed from its cell. A within-cell dominator would have no larger latency, remain feasible, and have either lower Q or equal Q with lower ρ, contradicting the winner's lexicographic minimality. Thus every nonempty query has its original winner in $\mathcal{A}^{\mathcal{V}}$. If the original feasible set is empty, its subset in $\mathcal{A}^{\mathcal{V}}$ is empty as well. Exact objective ties may be retained, or reduced to the smallest canonical key consistently with the selector.

The identity concerns budget readout, whereas the two cell extrema govern refinement. Two extrema alone need not preserve all budget queries. For example, three illustrative policies in one cell with (Q, ρ) values $(1, 0.3)$, $(2, 0.2)$, and $(3, 0.1)$ are all non-dominated; budget $\bar{\rho} = 0.2$ selects the middle point, which is neither extreme. These are constructed values, not experimental observations. The full observed pool or its correctly recomputed Pareto repertoire preserves that query. The identity

does not apply to a stale online front under a different timing denominator or eligibility filter, and does not bound the quality of unobserved policies.

O.4 A SUFFICIENT CONDITION FOR BACKEND-INDUCED POLICY SELECTION

For backend b, define the budget-feasible set $\mathcal{F}_b(\bar{\rho}) = \{z \in \mathcal{V} : \rho_b(z) \le \bar{\rho}\}$. Fix the same finite policy pool $\mathcal{V}$, frozen objective Q, and budget $\bar{\rho}$ for two backends. Suppose the native feasible set is nonempty and its selected policy is z_0. If a lower-discrepancy policy z_1 satisfies $Q(z_1) < Q(z_0)$ and

$$\rho_{\mathrm{PSWA}}(z_1) \le \bar{\rho} < \rho_{\mathrm{native}}(z_1), \tag{77}$$

then z_1 is excluded by the native budget but admitted by PSWA. The PSWA selector minimizes Q over a set containing z_1, so

$$Q(\hat{z}_{\mathrm{PSWA}}(\bar{\rho})) \le Q(z_1) < Q(z_0). \tag{78}$$

Its selected identity must therefore differ from z_0. If the native feasible set is empty, admitting any policy instead changes attainment; this is distinct from improving the quality of an existing native incumbent.

The condition concerns threshold crossing, not a necessary reversal of relative policy costs. In particular, a uniform acceleration factor $\gamma > 1$ with $\rho_{\mathrm{PSWA}}(z) = \rho_{\mathrm{native}}(z)/\gamma$ gives

$$\mathcal{F}_{\mathrm{PSWA}}(\bar{\rho}) = \mathcal{F}_{\mathrm{native}}(\gamma\bar{\rho}), \tag{79}$$

which can already change the selected policy while preserving non-dominated candidate membership. Thus the frozen-pool study tests budget-conditioned selection rather than non-uniform synergy or joint optimization over backends. Its Q values are borrowed reference-fidelity observations; the argument makes no claim about unmeasured backend-specific or ground-truth quality. Empirical timing margins and the preregistered repeat-envelope readout are reported separately in **Appendix E.4: backend-conditioned selection**.

O.5 PACKED EXECUTION PRESERVES WINDOW-ATTENTION SEMANTICS

For windows with valid Q/K/V blocks (Q_i, K_i, V_i), PSWA concatenates the blocks and records cumulative offsets

$$Q = \mathrm{cat}_i\, Q_i, \quad K = \mathrm{cat}_i\, K_i, \quad V = \mathrm{cat}_i\, V_i, \qquad C = [0, L_1, L_1 + L_2, \ldots, \sum_i L_i]. \tag{80}$$

A variable-length kernel uses C to keep windows isolated, reducing attention work from $\mathcal{O}((\sum_i L_i)^2)$ to $\mathcal{O}(\sum_i L_i^2)$ before scattering outputs back to the original sparse order.

Let M_{win} be zero within a window and $-\infty$ across windows. With dropout disabled and the same attention scale,

$$\mathrm{Attn}(Q, K, V; M_{\mathrm{win}}) = \mathrm{cat}_{i=1}^{N}\, \mathrm{Attn}(Q_i, K_i, V_i; 0). \tag{81}$$

Consider one head of dimension d and let I_i be the packed token indices of window i. For query index $r \in I_i$, the block-diagonal mask sets all logits outside I_i to $-\infty$. Its attention weight for key s is therefore

$$a_{rs} = \begin{cases} \dfrac{\exp(\langle q_r, k_s\rangle/\sqrt{d})}{\sum_{t\in I_i} \exp(\langle q_r, k_t\rangle/\sqrt{d})}, & s \in I_i, \\ 0, & s \notin I_i. \end{cases} \tag{82}$$

These are exactly the weights obtained by evaluating scaled dot-product attention on (Q_i, K_i, V_i) alone. Multiplying by the values and concatenating the per-window outputs proves Eq. (81). An inverse packing permutation restores the original sparse order. The argument holds independently for every head. Nonempty windows are executed; empty windows have no query rows. Any within-window bias or further mask must be identical in both computations and admit at least one key per executed query.

The identity assumes unchanged projected queries, keys, values, scale, and window membership. It is an exact-arithmetic statement about attention layout, not a bitwise guarantee for finite-precision kernels or a numerical guarantee for the whole iterative generator. The implementation's additional execution switches and empirical quality comparisons require their own recorded checks; the identity alone does not apportion the measured end-to-end speedup among those switches. See **Appendix E.2: PSWA and schedule decomposition** for the associated measurements.